

# CONSUL 2717



## **PRÁCE S POČÍTAČEM DRUHÝ KURS**

**Úvod:**  
**=====**

Druhý kurs práce s počítačem CONSUL 2717 volně navazuje na Úvodní kurs 10 lekcí obsluhy počítače a používání základních příkazů a funkcí jazyka BASIC-G. Proto i tento kurs je členěn do lekcí, číslovaných jako 11-20. Lekce jsou voleny tak, aby pro vysvětlení a pochopení problematiky postačovala asi hodina, což je závislé i na rychlosti zápisu jednotlivých příkladů, osvětlujících probíranou tematiku. Většina lekcí má na závěr uvedena jednoduchá cvičení, jejichž možná řešení jsou v příloze.

Druhý kurs si rovněž neklade za cíl naučit programovat, ale pracovat s počítačem, využívat jeho základní programové vybavení. Zvýšená pozornost byla věnována dialogu mezi programem a uživatelem, který se často nerespektuje, a tím uživatele počítače zavádí do nejistot.

Dvě lekce jsou věnovány práci v grafickém režimu počítače, problému češtiny (háčky a čárky) na obrazovce je věnována samostatná kapitola, v níž je využito nápadu RNDr. Roberta Gamby z Liberce, který jednoduše doplňuje diakritická znaménka do textu.

Lekce o datových polích a jejich ukládání na kazetu klade důraz na správnou manipulaci s daty v paměti při jejich vytváření, ladění programů a další práce s těmito daty.

Problematické vstup/výstupních obvodů je věnována pouze jediná lekce, i když je to téma velmi rozsáhlé a vyžaduje studium další literatury o vstup/výstupních obvodech. Lekci doplňují 3 samostatné přílohy programování a analýzy stavů v/v obvodů.

Poslední kapitola je věnována nejn nutnějším informacím pro práci v režimu MONITOR počítače CONSUL 2717.

Přílohy ERROR a ASCII obsahují chybová hlášení, včetně jejich kódů, a tabulku kódů ASCII.

Pro další studium jazyka BASIC je uvedena v poslední příloze doporučená literatura, která byla u nás vydána.

Přes veškerou péči, která byla sestavení Úvodního i tohoto kursu věnována, se mohou vyskytnout chyby nebo jiné (metodické, stylistické) nedostatky. Za jejich písemnou kritiku stejně jako za připomínky a náměty Vám předem děkuje

autor.

TEMA: Dialog mezi programem a uživatelem  
=====

Lekce: 11  
=====

NOVE POJMY: Instrukce zvukového výstupu, funkční klíče,  
----- řízená přestávka v programu, programový přepínač;  
volba různých variant v programu (MENU),  
náповéda pro uživatele programu (HELP).

NOVE PŘIKAZY: BEEP -akustické znamení (pípnutí);  
----- PAUSE N -pozastavení programu na zadanou dobu;  
AND,OR,NOT -logické operátory;  
INKEY -načtení obsahu funkčních klíčů F1;  
ON ... GOTO-programový přepínač.

Některé příkazy jazyka BASIC-G se mohou zdát na první pohled samostatné, ve skutečnosti jsou užitečnými pomocníky uživatele.

Zvukový výstup počítače je užitečný pro upozornění obsluhy, že se něco zajímavého nebo důležitého stalo. Pokud nahráváme program, pípnutí po přečtení jeho klavičky oznamuje, že nesouhlasí číslo nahrávky; pokud nahráváme dobře, oznámí pípnutí konec nahrávání bez chyby (OK) nebo s chybou (++File error++). Pokud je při vstupu dat (INPUT) požadováno více hodnot, pípnutí značí, že jsme ještě nevyčerpali všechny; každé chybové hlášení se ohlašuje tímto způsobem. Pro upozornění uživatele programu na podobné okolnosti je možno v jazyku BASIC-G použít příkaz BEEP. Např.

IF X>100 THEN BEEP:IF X>200 THEN BEEP:BEEP

Pozor! Napsat dva podmínkové výrazy za sebou lze jen tehdy, je-li výraz druhé podmínky splněn i v podmínce první (200>100). Pokud by tomu tak nebylo, nikdy by se druhá podmínka netestovala (zbytečné příkazy). Jistější je mít na řádku pouze jedinou podmínku. BEEP nemá žádné parametry a proto jej není možno modifikovat.

V některých aplikacích je nutno pozastavit průběh programu na přesně stanovenou dobu. To umožňuje příkaz

PAUSE N

jehož parametr určuje násobky 0.1sec a může být v rozsahu 1-255 (tj. 0.1 - 25.5 sec); není-li parametr uveden nebo je 0 nastaví se 255. Parametr N může být i proměnná nebo výraz. Zrušit čekání lze stiskem mezerníku. PAUSE by se měla užívat tam, kde je to účelné (zobrazení informace, otázky, mezivýsledku), neměla by ale omezovat uživatele programu např. tím, že musí dlouho čekat, nebo naopak tím, že je informace zobrazena velmi krátce (např. když je nutno přečíst celou stránku textu). V takových případech je vhodné ponechat na uživateli, kdy chce pokračovat -např. stiskem libovolné klávesy při zprávě v dialogovém řádku, zadané příkazem

55 ?"...pro pokračování stlačte klávesu"

V podmínkových příkazech se často používají výrazy s logickými operátory:

- NOT - logická negace;
- AND - logický součin (konjunkce);
- OR - logický součet (disjunkce).

Operátory AND a OR jsou tzv. binární, neboť se vztahují ke dvěma operandům (proměnným, konstantám a pod.) mezi nimiž se operátor nachází. Operátor NOT je unární, protože se vztahuje k jedinému operandu, který za ním následuje. Vyhodnocením logické operace vznikne pravdivostní hodnota pravda (true,1) nebo nepravda (false,0), tj. podmínka definovaná touto operací je nebo není splněna. Pravdivostní tabulky logických operací mají tento tvar:

A	NOT A
1	0
0	1

A	B	A AND B
0	0	0
0	1	0
1	0	0
1	1	1

A	B	A OR B
0	0	0
0	1	1
1	0	1
1	1	1

V tabulkách značí 1 pravdu (platnost) a 0 nepravdu (neplatnost). Negaci lze použít například místo neekvivalence, neboť stejnou hodnotu mají:  $A \neq B$  (A různě od B) jako: NOT  $A=B$  (není pravda, že A je rovno B). Častěji je využíván logický součin k testování, zda jsou současně splněny zadané podmínky (nebo jinak: platí-li první podmínka A SOUČASNĚ platí i druhá podmínka). Například:

```
IF (P*7+S*5+T*1=100) AND (P+S+T=50) THEN 70
```

Jestliže si ještě vzpomínáte na složitý úkol nákupu v 6. lekci, kdy jsme měli za 100 Kčs nakoupit přesně 50 kancelářských potřeb v různé ceně.

Pokud postačuje splnění jedné z několika podmínek, potom využijeme operátor logického součtu - OR, jak jsme to použili např. v lekci 9 na rozhodování o platných rocích, měsících a dnech při sestavení rodného čísla:

```
IF R<1900 OR R>1988 THEN PRINT"To snad ne?":...
IF M=0 OR M>12 THEN ...
IF D=0 OR D>31 THEN ...
```

Slovně to znamená: platí-li první podmínka NEBO druhá podmínka, potom vykonaj následující příkaz.

Logické operace budeme často používat v této i dalších lekcích, kde je bude možno podrobněji poznat v různých situacích.

Funkce INKEY umožňuje načtení hodnoty z funkčních klíčů F0-F11, protože po stlačení klíče se jeho číslo přečte do zvolené proměnné. Pokud není stlačen žádný klíč, načte se hodnota 255, je-li současně stlačeno více klíčů, platí hodnota nejvyššího. Provádění programu se ovšem nepozastavuje, proto je nutno vstup informace z klíčů programově ošetřit např. takto:

```
50 DISP"Dalsi postup vyber klicem F2-F7"
55 A=INKEY: IF A=255 THEN 55      :REM Cekani na klic
60 IF A<2 OR A>7 THEN 50         :REM Povoleny klice F2-F7
65 IF A=2 THEN PRINT"Stlacen F2": GOTO 400
70 IF A=3 THEN...
```

Pokud požadujeme větvení programu podle hodnot stlačených klíčů, lze postupovat pomocí podmíněných příkazů typu řádku 65, nebo elegantněji pomocí tzv. programového přepínače, který má obecnější použití, a ušetří řadu samostatných programových řádků typu IF ... THEN ... Programový přepínač má obecný tvar

ON výraz GOTO seznam čísel řádků

Výraz uvedený v příkazu se vyhodnotí a jeho celočíselná hodnota (INT) určuje pořadové číslo řádku v seznamu řádků, na němž bude program pokračovat. Hodnota výrazu musí být z intervalu 1-255, jinak se ohlásí chyba ( ++Fnc.param++ ). Pořadí čísel řádků je poziční zleva doprava, jak názorně ukazuje tento příklad:

```
10 PRINT"Zadej znamku (1-5), stlac EOL":INPUT Z
15 IF Z=0 OR Z>5 THEN PRINT"Takova neexistuje !":GOTO 10
20 PRINT Z;"=":"ON Z GOTO 21,22,23,24,25
21 PRINT"Vyborne":GOTO 10
22 PRINT"Chvalitebne":GOTO 10
23 PRINT"Dobre":GOTO 10
24 PRINT"Dostatecne":GOTO 10
25 PRINT"Nedostatecne":GOTO 10
```

Pomocí hodnot z klíčů můžeme vytvořit nabídku (menu) uživateli:

```
10 DATA0,Prikaz,15,Evol klicem,,,,:REM Nadpis+prazdny radek
12 DATA0,GCLEAR,25,F1,0,SCALE,25,F2,0,AXES,25,F3
14 DATA0,MOVE,25,F4,0,PLOT,25,F5,0,FILL,25,F6
16 DATA0,BMOVE,25,F7,0,BPLOT,25,F8,0,KONEC,25,F9
18 RESTORE10:FOR R=5 TO 15:READ S,Tx,S1,Fx
20 PRINT AT R,S,Tx:AT R,S1,Fx: NEXT R
22 A=INKEY:IF A=0 OR A>9 THEN 22: REM A=255 patri mezi A>9
24 ON A GOTO 100,200,300,400,500,600,700,800,900
100 GCLEAR:PRINT AT0,0:"Popis prikazu: GCLEAR":.....
900 GCLEAR:PRINT AT 26,0"koniec popisu grafickych prikazu."
```

Místo nabídky pro 'KONEC' lze uvést 'DALSI PRIKAZY' a na řádku 900 (nebo libovolném jiném pro klíč F9) pokračovat v nabídce se stejnými klíči, ale jinými jim odpovídajícími řádky programu. Pokud by se pořadí programových řádků v přepínači nevešlo na jeden řádek, lze pokračovat podobným příkazem na dalším řádku, např.

```
26 ON A GOTO 1000,1100 :REM Pro A=10,11
```

Je ovšem nutno změnit i podmínku platnosti A (obdobně řádku 22). Pokud je nabídka krátká, lze ji uvést v dialogovém řádku, např.

```
60 DISP"F1-dalsi cast F2-znovu F3-tisk F4-konec"
62 F=INKEY:IF F=0 OR F>4 THEN 62
64 ON F GOTO 71,72,73,79
71 K=K+1:GOTO...
72 RUN :REM Znovuspusteni
73 ?"Je-li zapnuta tiskarna, stlacte klavesu"
74 CONTROL 4,3,132,5:PRINT #404;"Tabulka dilcich vysledku"
75 FOR I=0 TO K:PRINT #404;"Pro K=";I;" je ....":NEXT I
76 GOTO 60 :REM Navrat na nabidku
79 END
```

Podobnou nabídku využijeme tehdy, nechceme-li přepsat výsledky práce programu v pracovní oblasti stínítka. Opět lze místo KONEC zařadit DALSI a pokračovat další nabídkou s jinými klíči a pod. Takové nabídky dalšího postupu by se měly v programu objevit po každé, jakmile vyžadujeme od uživatele programu nějaké rozhodnutí. Není dobrou vizitkou pro autora programu, když nabídne jako varianty dalšího postupu několik příkazů GOTO... A co když se uživatel v roztržitosti splete v čísle řádku a zničí si předchozí výsledky práce programu. Proto se v dobrých programech objevuje obvykle další 'pojišťující' informace, např. po F6-smazat se do dialogového řádku napíše

```
? "Doppravdy smazat? (AND/NE a EOL)"
```

a definitivně se rozhoduje až podle napsaného slova. Obdobnou funkci má řádek 73 našeho příkladu, protože při stlačení klíče F3-tisk zmizí obsah dialogového řádku a tiskne se, pokud je zapnuta tiskárna (a připojena ON LINE,READY). Není-li zapnuta, neděje se nic, klávesa STOP je neúčinná, a co má uživatel dělat ?? (Selže-li všechno, přečti si návod.)

Cvičení: 1. Napište programový přepínač pro volbu napsáním prvního písmene z nabídky: D=Dalsi, Z=Znovu, T=Tisk, K=Konec. Srovnajte vstup pomocí INPUT A\$ v cyklu zkoumání pořadí znaků v řetězci B\$="DZTK" pomocí MID\$(B\$,I,1).  
2. Ověřte, jaké hodnoty dávají klíče Fi se SHIFT.

TEMA: Podprogramy a jejich využívání  
=====

Lekce: 12  
=====

NOVE POJMY: Co je to podprogram, jeho volání a ukončení,  
----- podprogramový přepínač, způsob ošetření chyb.

NOVE PŘIKAZY: GOSUB - volání podprogramu;  
----- RETURN - návrat z podprogramu;  
ON výraz GOSUB- podprogramový přepínač;  
ON ERR GOTO - skok na ošetření chyby.

Při tvorbě programů se často stává, že je nutné opakovat stejnou skupinu příkazů na různých místech programu. Aby nebylo nutné opakovat zápis stejného algoritmu, je účelné zapsat jej pouze jedenkrát jako tzv. podprogram, z hlavního programu si do něj 'odskočit' vždy, když to bude potřeba, a potom se nezapomenout vrátit. Pro skok do podprogramu je používán příkaz

GOSUB číslo řádku

kde GOSUB je zkratkou z anglického 'GO to SUBrutine' (jdi na podprogram), a číslo řádku určuje, kde tento podprogram začíná (to je tzv. vstup do podprogramu). Pro návrat (výstup z podprogramu) do hlavního programu slouží příkaz

RETURN

který může být v podprogramu i vícekrát (nedoporučuje se, podprogramy by měly mít jeden vstup i výstup), ale vždy má stejnou úlohu: vrátí se vždy za příkaz GOSUB, který jej volal, byť je to na další programový řádek, nebo i na řádek, kde je několik volání různých podprogramů. (Je to proto, že přetěnění GOSUB si BASIC zapamatuje adresu dalšího příkazu v pořadí jejich vykonávání, a na tuto adresu se vrátí po RETURN.)

Jako příklad si uvedeme podprogram na podtrhávání textu na obrazovce různými symboly, které podprogramu předáme pomocí proměnné P (parametru); tu musíme definovat před voláním podprogramu, v němž je proměnná využita jako hodnota kódu ASCII znaku (CHR\$(P))

```
10 GCLEAR:TX="Tento text chceme podtrhnout.":PRINT TX
15 P=45:GOSUB50:REM Podtrhnutí pomocí '=' (kód 45)
20 P=34:GOSUB50:P=42:GOSUB50:P=127:GOSUB50
25 END
50 FOR I=1 TO LEN(TX):PRINT CHR$(P);:NEXT I:PRINT:RETURN
```

Nezapomněli jste na středník za PRINT v cyklu? Proto asi víte, z jakého důvodu byl zařazen do podprogramu osamocený příkaz PRINT.

Doporučuje se podprogramy umísťovat na konec hlavního programu, aby je nebylo nutno přeskakovat. Protože však každé vícemístné číslo řádku 'ukrajuje' paměť o to více, kolikrát je použito, je nutné někdy zařadit podprogramy před hlavní program (první příkazový řádek obsahuje příkaz přeskočení podprogramů: GOTO... nebo programový přepínač ON...GOTO...).

Jako parametr v GOSUB můžeme použít i proměnnou nebo výraz, který nesmí začínat číslicí, např.

GOSUB N\*10

Podobně jako programový přepínač pracuje přepínač podprogramů, vytvořený z příkazů

ON výraz GOSUB seznam čísel řádků

Volané podprogramy musí být ukončeny příkazy RETURN, které zajišťují návrat do hlavního programu na řádek, následující za příkazem ON...GOSUB. Počet podprogramů není omezen, podprogramy mohou obsahovat volání dalších podprogramů (vněšování podprogramů).

Při práci s programy mohou vznikat různé chyby, které počítač ohlásí anglickou zkratkou (viz přílohu) a další práci na programu zastaví. To je dobré při ladění programů pro programátora, ale uživateli programu to může znepříjemnit práci, znervózni a začne se dopouštět dalších chyb. Tomu lze předejít jednak tím, že uživatele včas na všechny 'základnosti' připravíme, kontrolujeme jeho práci při ovládání programu, zadávání dat, obsluhu periférií. Jestliže přesto dojde k chybě, je vhodné ji srozumitelně vypsát (česky, nikoli anglicky) a nabídnout další postup - vždyť například použití příkazu RUN ke znovuspouštění programu zničí všechny proměnné, dimenzování polí lze použít jen jednou apod. Příkaz

ON ERR příkaz

umožňuje potlačit hlášení o chybě a umožňuje její ošetření samostatným programovým modulem. Obvykle se ON ERR... zadává na začátku programu - pokud se v programu nevyskytne chyba, počítač tento příkaz ignoruje. Pokud se chyba objeví, počítač nedokončí zpracování na chybném řádku, ale skočí na 'příkaz' za ON ERR.... Je vhodné, aby tímto příkazem byl skok na první řádek ošetřovacího programového modulu. Příkazem ON ERR... se vyprázdní zásobník návratových adres podprogramů ('provedou' se všechny RETURN) i zásobník rozpracovaných cyklů (FOR...TO...NEXT). Také se zruší režim nezastavování na chybě a při další chybě by se provádění programu zastavilo (pokud mu novým příkazem ON ERR... nezabráníme). Příkaz ON ERR GOTO... musí být umístěn na řádku jako poslední (jako GOTO...); za ním lze umístit jen komentář.



Známost chybu "dělení nulou" (++Dv by zero++) bychom mohli ošetřit takto:

```
10 GCLEAR:PRINT"Vypocet prevracenych hodnot"
20 ON ERR GOTO 80
30 PRINT,"Zadej cislo: (a stlac EOL)";:INPUT A
40 PRINT"1/";A;"=";1/A
50 DISP"Pokracovat - P, Konec - K (a EOL)";:INPUT Ax
60 IF Ax="P" THEN 30
70 END
80 REM Osetreni chyby
85 PRINT"Pozor! Nulou nelze delit !": GOTO20
```

Namítnete, že jsme v tomto příkladu předem věděli k jaké chybě dojde při zadání čísla 0. BASIC-G si číslo chyby ukládá na adrese '26=38, z níž si je můžeme přečíst příkazem E=PEEK('26) a podle velikosti proměnné E a znalosti čísel chyb (jsou v příloze) můžeme takový program sestavit. Protože nás bude zajímat i řádek na kterém vznikla chyba, můžeme si jeho 'dvoubytovou' hodnotu přečíst příkazem CR=APEEK('5E5D). Zkusme si upravit předcházející program takto:

```
80 PRINT"Chyba cislo:";PEEK('26);" na radku:";APEEK('5E5D)
```

Podrobněji se s příkazy PEEK a APEEK seznámíme později.

Podprogramy je vhodné využívat i při opakovaném kreslení obrázků

```
10 GCLEAR:SCALE0,255,0,242:X=90:Y=90:GOTO 30
15 MOVE X+13,Y+2:PLOT X+18,Y+5;X+20,Y+10;X+18,Y+15
20 PLOT X+13,Y+18;X+8,Y+15;X+6,Y+10;X+8,Y+5;X+13,Y+2:RETURN
30 FOR A=2 TO 14:FOR X=0 TO 230 STEP A
35 Y=10*A:GOSUB 15:NEXT X:A:END
```

Cvičení: 1/ Převedte program 'známky' v minulé lekci z typu

----- ON..GOTO.. na ON..GOSUB..

2/ Vytvořte program na ošetření některých chyb BASIC-G (využijte podmínku OR pro různé chyby a skok na řádek E+10, pokuste se otestovat vytvořený program).

E=8 - Overflow

E=12- Dv by zero

E=15- No str.spc

E=17- Type conv.

E=12- Dv by zero

E=19- Input err

E=23- File error

TEMA: Práce v grafickém režimu počítače

Lekce: 13

NOVE POJMY: Soustava souřadnic a její měřítko;  
----- kreslení os souřadnic, pohyb pera  
v rovině, kreslení spojnice dvou bodů.

NOVE PŘIKAZY: SCALE  $X_1, X_p, Y_d, Y_h$  -měřítko souřadnicové soustavy;  
----- AXES  $X, Y$  -kreslení os v bodu  $X, Y$ ;  
MOVE  $X, Y$  -přesun pera do bodu  $X, Y$ ;  
PLOT  $X, Y; P$  -spojnice s novým bodem  $X, Y$ .

V grafickém režimu počítače lze pomocí několika příkazů kreslit úsečky, křivky a podobné útvary do pracovního prostoru obrazovky.

Kreslení všech útvarů probíhá vždy v nějakém souřadnicovém systému, který je nutno nejprve na pracovní ploše obrazovky zvolit. Vodorovně můžeme vykreslit úsečku o délce až 256 bodů (tzv. horizontální rozlišovací schopnost stínítka), zatímco svisle můžeme rozlišit pouze 243 bodů-pixelů (vertikální rozlišení). Vodorovná osa bývá označována  $X$  a svislá  $Y$ , hodnoty souřadnic na těchto osách rostou zleva doprava a zdola nahoru. Nejmenší souřadnici  $X$  můžeme označit  $X_1$  (levá) a největší  $X_p$  (pravá). Obdobně bude  $Y_d$  (dolní) nejmenší a  $Y_h$  (horní) největší souřadnici na svislé ose.

Volbu určité části z nekonečné roviny na obrazovce provádíme pomocí příkazu: SCALE  $X_1, X_p, Y_d, Y_h$ . Tento příkaz nastaví měřítko pro veličiny  $X$  a  $Y$ , které chceme používat ke kreslení na stínítku. Budeme-li využívat všechny body pracovní plochy, můžeme použít příkaz

SCALE 0,255,0,242

který určí počátek souřadnicového systému vlevo dole a na stínítku umožní využívat část prvního kvadrantu (obě souřadnice  $X, Y$  jsou kladné). Pro zobrazení ve všech kvadrantech je nutno umístit počátek souřadnic do středu obrazovky "symetrickým" příkazem např.

SCALE -1,1,-1,1

který musí provést transformaci mezi počtem zobrazovaných bodů a hodnotami zobrazovaných veličin (mezi počátkem 0 a  $X=1$  je 127 bodů, ale svisle mezi 0 a  $Y=1$  je jen 120 bodů). Počet bodů, které připadnou na jednotku hodnoty zobrazované veličiny je celé číslo:  $\text{INT}(255/(X_p - X_1))$  -vodorovně a  $\text{INT}(242/(Y_h - Y_d))$  -svisle.

Parametry příkazu SCALE mohou být jak konstanty, tak proměnné či výrazy, a platí až do nového zadání. Příkaz SCALE musí předcházet ostatním grafickým příkazům, jinak by byly neúčinné.

Pro vykreslení vodorovné a svislé čáry, procházející bodem se souřadnicemi X,Y slouží příkaz

#### AXES X,Y

kde souřadnice mohou být zadány opět přímo (konstantou), nebo nepřímo (proměnnou nebo výrazem). Při zpracování tohoto příkazu se nejdříve vykreslí vodorovná čára zleva doprava (od  $X_l$  po  $X_p$ ) a potom svislá zdola ( $Y_d$ ) nahoru ( $Y_h$ ). Pero (kursor) zůstane nastaveno na dolním konci svislé čáry (zkuste si to, a uvidíte)

Orámování pracovní plochy stínítka dosáhneme pomocí příkazů AXES s parametry, odpovídajícími krajním souřadnicím příkazu SCALE:

```
SCALE -4,4,-3,3:AXES -4,-3:AXES 4,3
```

Ke zrušení rámečku lze použít stejných příkazů, protože kreslení se provádí negací (inverzí) předcházejícího stavu na obrazovce. Samozřejmě lze použít příkazu GCLEAR. Pomocí AXES lze nakreslit celostránkovou tabulku:

```
10 GCLEAR:SCALE 0,10,0,10
20 FOR K=0 TO 10:AXES K,K:NEXT:END
```

Pro kreslení v grafickém režimu slouží pomyslné pero, jež je nutno zadat výchozí souřadnice pomocí příkazu

#### MOVE X,Y

(a nesmíme před tím zapomenout na příkaz SCALE). Proto se pomocí MOVE 0,0 můžeme nastavit do počátku souřadnicového systému, polohu pera můžeme zadat i nepřímo pomocí proměnných nebo výrazů, po dokončení grafických příkazů zůstane pero na nových souřadnicích, jak bude u těchto příkazů uvedeno.

Pro kreslení bodů nebo úseček má BASIC-G příkaz

```
PLOT X,Y,1   nebo   PLOT X,Y
```

Výchozí bod pro kreslení musí být nastaven pomocí MOVE A,B, nebo je dán pozicí pera z předcházejícího příkazu PLOT X,Y (nebo příkazu AXES X,Y). Zkuste si nakreslit trojúhelník pomocí programu:

```
5 GCLEAR:SCALE-2,2,-2,2:MOVE0,0
10 PLOT1,1:PLOT-1,1:PLOT0,0:END
```

Následuje-li za sebou více stejných příkazů PLOT, lze vynechat jejich psaní, a jednotlivé souřadnice oddělit středníkem, např.:

```
10 PLOT-1,-1;2,0;0,0:END
```

Pokud "třetí" parametr příkazu PLOT je nula, nebo není uveden, tak pero kreslí z výchozí pozice do zadané cílové pozice. Pokud je tento parametr nenulový, přesune se pero bez kreslení úsečky, ale v nové pozici se "ohlásí" vykreslením tečky.

Zkusme si doplnit předcházející program takto:

```
8 PLOT 1,1,1,2,-2,1,-3,-3,1,-4,4,1
```

Po spuštění programu se vykreslily i rohy čtyřúhelníka.

Pokud je některá ze souřadnic v příkazu PLOT X,Y mimo rozsah daný měřítkem SCALE, odpovídající úsečka se nevykreslí.

Zkusme si vykreslit pomocí cyklu "prostorový" obrázek

```
10 GCLEAR:SCALE0,255,0,242
20 FOR I=0TO80STEP4:PLOT100-I,100-I,1
30 PLOT 100-I,I/2,I/2,I/2,I/2,100-I,100-I,100-I
40 NEXT I:END
```

Další příklad vykreslí tabulku v horní polovině obrazovky se zadaným počtem sloupců a řádků:

```
10 GCLEAR:SCALE 0,255,-120,120
15 PRINT AT24,0,"Zadej počet sloupcu tabulky <25",
20 INPUT SM:ST=255/SM
25 PRINT AT25,0,"Zadej počet radku tabulky <40",
30 INPUT RM:RT=255/RM
35 FOR R=120 TO 0 STEP -RT:MOVE 0,R:PLOT 255,R:NEXT R
40 FOR S=0 TO 255:MOVE S,0:PLOT S,120:NEXT S:END
```

Pro zobrazení funkce lze využít také kreslení v cyklu.

```
10 GCLEAR:PI=3.14159:SCALE -PI/18,4*PI,-1,1,1 1
15 AXES 0,0:MOVE 0,0
20 FOR X=PI/18 TO 4*PI STEP PI/18
25 PLOT X,SIN(X),1:NEXT X:END
```

Protože je v PLOT využit třetí parametr (1), průběh obou period funkce SIN(X) je vytečkován; jeho vynecháním získáme plnou křivku. A ještě jeden příklad složitějšího grafického obrazce:

```
10 GCLEAR:SCALE 0,255,0,242:PI=3.14159
15 FOR F=0 TO 2*PI STEP PI/4
20 FOR G=F-PI/4 TO 2*PI STEP PI/4
25 MOVE120+40*SIN(F),100+50*COS(F)
30 PLOT 120+40*SIN(F+G),100+50*COS(F+G)
35 NEXT G,F:END
```

Cvičení: 1. Nakreslete vánoční stromek s 5 páry větví a kmenem  
----- 2. Nakreslete kolo vypletené 60 dráty, pro kružnice použijte cyklus: FOR I=0 TO 2\*PI STEP 11  
PLOT SIN(I),COS(I):NEXT I

TEMA: Zobrazení v grafickém režimu

lekcce- 14

=====

=====

NOVE POJMY: Varianty způsobu vykreslování;  
----- psaní textových informací v grafice;  
vykreslování plných ploch, maska kreslení.

NOVE PRIKAZY: PEN -volba způsobu kreslení perem;  
----- LABEL -zobrazení textu v různých měřících;  
FILL -vykreslení plné plochy.

Naučili jsme se pohybovat perem a kreslit úsečky. Další grafický příkaz umožňuje změnou "pera", tj. parametru K v příkazu PEN K zobrazovat některým z následujících způsobů

K=0 -kreslení plným jasnem (nastavováním);  
1 -kreslení polojasem;  
2 -blikání;  
4 -inverse (tmavé na bílém);  
8 -negování předchozího zobrazení - základní režim kreslení;  
16 -mazání dříve zobrazených bodů

Způsob vykreslování je dán hodnotou parametru K, který může být i součtem výše uvedených hodnot. Např. PEN 3 značí blikání polojasem, není-li v součtu hodnota 4, bude kresba světlá na tmavém pozadí a naopak. Po zapnutí počítače (nebo RESET) je nastavena vždy hodnota PEN 8. Zkuste si uvedené možnosti tímto programem:

```
10 GCLear:SCALE -2,2,-2,2:AXES 0,0      :REM Měřitko a souř. osy
20 DATA 8,1,2,3,4,6,0,10,12,16,20      :REM Parametry pera
30 FOR J=1 TO 10:READ K:PEN K:MOVE 0,1  :REM 10* nastavit pero
40 DISP"PEN";K
50 FOR I=1 TO 12.56 STEP 0.11            :REM Cyklus pro kresbu
60 PLOT SIN (I),COS (I)                  :REM obvodu kružnice
70 NEXT I:PAUSE10:NEXT J:END
```

Příkaz PLOT můžeme modifikovat pro "bodové" kreslení na konci úsečky PLOT změnou řádku 60 takto

```
60 PLOT SIN (I),COS (I),1
```

a aby to netrvalo tak dlouho, můžeme zvětšit STEP na 0.4 i více. Z příkladu je vidět, že ne všechny možnosti pera jsou vhodné a navíc, že se nastavení pera vztahuje na šestici bodů v každém grafickém řádku (proto byly vykresleny obdelníky místo úseček).

Zatím jsme měřitko jenom definovali, ale vykreslené osy nepopsali -ani by to pro různá měřítka nešlo použitím příkazu PRINT AT ...

Pro tyto případy je v grafickém režimu určen příkaz

### LABEL H,V/výrazy

kde H určuje horizontální zvětšení (konstanta, proměnná či výraz)  
V určuje vertikální zvětšení  
výrazy tvoří je libovolné znaky přípustné v BASIC-G, od konstant až po řetězce (ty pochopitelně v uvozovkách).

Příkaz LABEL se nejčastěji používá na psaní zvětšených textů a popisování souřadnic. Začátek kreslení (levý dolní roh prvního znaku) je určen předcházejícím příkazem (MOVE, LABEL, PLOT, FILL). Výrazy lze oddělovat podobně jako v PRINT ( ; , SPC). Po LABEL zůstane kurzor vpravo dole za posledním zobrazeným znakem.

Ukážeme si to v následujícím příkladu:

```
10 GCLEAR:SCALE -1,11,-1,11:AXES 0,0:PEN 8
20 FOR I=0 TO 9:MOVE I-.2,-.5:LABEL 1,1:I
30 NEXT I:LABEL 2,2;" x"
40 MOVE 1,9:LABEL 3,3;"Popis osy x"
50 MOVE 1,1,9,1:LABEL *;"Popis osy x"
60 END
```

V příkazu LABEL na řádce 50 jsme místo stejných parametrů napsali pouze znak \*, což jazyk BASIC-G připouští. Posunutí výchozích souřadnic v MOVE mělo za následek vykreslení ozdobného textu.

Pohyb znaku (textu) po obrazovce lze realizovat cyklem proměnné, která je použita v příkazu MOVE. Vykreslený znak je nutno před pohybem (změnou MOVE) smazat na stejné pozici pomocí téhož LABEL jinak by zůstal a rozmazával se ve směru pohybu.

```
70 GCLEAR:SCALE 0,266,0,242:AX=">"
80 FOR R=0 TO 200 STEP 4:MOVE R,R:LABEL 2,2;A*
90 PAUSE 1:MOVE R,R:LABEL 2,2;A*:NEXT R:END
```

Poslední grafický příkaz

### FILL S,V/maska

umožňuje vykreslit na obrazovce vpravo a nahoru od nastaveného pera obdelník o šířce S (0-255 vodorovně) a výšce V (0-242). Po zpracování příkazu zůstane kurzor vlevo nahoře. Tento příkaz můžeme použít pro vyznačení přesné pozice na ose, zařadíme-li jej do programu popisu os v řádce

```
25 MOVE I+1,0:FILL 1,5;1
```

Jistě jste si povšimli, že pro S a V neplatí měřítko SCALE, ale skutečné počty zobrazených bodů (pixelů); v našem případě to byla šířka 1 bod a výška 5 bodů. Navíc jsme v příkazu MOVE zadali souřadnici X o +1 větší, aby nám FILL neumazal část osy Y v počátečních souřadnicích. A protože S a V mohou být i proměnnými nebo výrazy (nabývající pouze povolené hodnoty), můžeme kombinací MOVE ... a FILL ... kreslit nejen obdelníky a čtverce.

A nyní nám zbývá odhalit tajemství masky. Je to tzv. bitová maska, sestávající ze šesti bitů. Z této masky se zobrazí pouze ty body, v jejichž bitech je jednotka - pozor na opačné pořadí bitů

pozice bitu	0	1	2	3	4	5	
"váha" bitu	1	2	4	8	16	32	
maska = 1	1	0	0	0	0	0	(1+0)
maska = 7	1	1	1	0	0	0	(1+2+4)
maska = 21	1	0	1	0	1	0	(1+4+16)
maska = 63	1	1	1	1	1	1	(1+2+4+8+16+32)

Názorněji si význam masky předvedeme následujícím programem:

```

10 GCLEAR:SCALE 0,255,0,242:MOVE 10,10
20 FOR I=1 TO 63: FILL 10,2:I: NEXT I
30 LABEL 1,1:"Rostoucí hodnota masky ve FILL 10,2:I"
40 END

```

Pomocí FILL lze kreslit různé vodorovné nebo svislé grafy. Např.

```

10 GCLEAR:SCALE 0,255,-242,0:AXES 40,-10
15 PRINT AT0,12"Fond pracovních dnu v měsících:"
20 FOR R=2 TO 24 STEP 2: READ M$:PRINT ATR,0,M$::GOSUB40:NEXT:END
25 DATA Leder,Unor,Brezen,Duben,Kveten,Cerven
30 DATA Cervenec,Srpen,Zari,Rijen,Listopad,Prosinec
40 DISP,"Zadej počet pracovních dnu měsíce:":INPUT P
45 PRINT ATR,9:P:MOVE 60,-(9+R*9):FILL 5*P,8;1:RETURN

```

- Cvičení:
1. Nakreslete šachovnici pomocí masek příkazu FILL.
  2. Vánoční stromek z lekce 13 vykreslete plně také pomocí příkazu FILL s proměnnými parametry.
  3. Nakreslete 10 kruhů s různým poloměrem, náhodně rozmístěných na obrazovce, a do jejich středů vepište jejich pořadová čísla.

TEMA: Jak vyzrát na háčky a čárky  
=====

Lekce: 15  
=====

NOVE POJMY: Znakově orientovaná plocha obrazovky,  
----- nastavení pera na této ploše,  
vykreslení zadaného obrázku,  
převod čísla na znak z ASCII kódu.

NOVE PŘIKAZY: BMOVE -pohyb pera ve znakově orientovaném prostoru,  
-----BPLOT -vykreslení obrázku tvořeného šesticemi body;  
CHR\*(K)-převod čísla K na znak z ASCII kódu.

Znakově orientovaná plocha obrazovky není nic jiného než to, že můžeme místo znaků ASCII zobrazit i znaky jiné (například české, slovenské, řecké, azbuku) nebo různé jiné malé obrázky. Když se podíváte zblízka na písmena textu na obrazovce, jsou vytvořena z teček, kterých je až 5 vedle sebe a 7 pod sebou. Protože je za písmenem a mezi řádky nutno mít mezeru pro lepší čitelnost, říkáme, že je znak zobrazován v matici 6\*8 bodů. například:

XXXXX.	1. mikrořádek
X.....	2.
X.....	3.
XXX...	4.
X.....	5.
X.....	6.
XXXXX.	7.
.....	8.

Na řádek obrazovky se vejde celkem 48 znaků vedle sebe, to je 48 šestic s pořadovými čísly 0-47. Na výšku pracovní plochy obrazovky je to 243 bodů, tvořících tzv. mikrořádky s čísly od 0 do 242. Ve skutečnosti se pro oddělení řádků používají 2 mikrořádky a tak je možno na obrazovku umístit  $\text{INT}(243/9)=27$  řádků. Poslední nelze využít, dostupných je jenom 26, s pořadovými čísly 0-25 (PRINT AT ...). Protože se text píše shora dolů, je počátek této znakově orientované plochy vlevo nahoře. Takto jsme si objasnili možnou velikost parametrů nového příkazu

BMOVE X,Y

kde X = 0-47 znamená pozici šestic bodů na řádku;  
Y = 0-243 značí pořadové číslo mikrořádku.

Proto můžeme pomocí BMOVE nastavit pero/kursor i tam, kam to nedokáže příkaz PRINT, i když vodorovně jenom po šesticích bodech. Rozdělení na šestic bodů je pevné, proto není použitelný příkaz SCALE. Parametry BMOVE mohou být jak konstanty, tak i proměnné nebo výrazy.



Pro rychlé vykreslení zadaného znaku, motivu či obrázku slouží příkaz

BPLOT AX,N

který zobrazuje N šestic bodů vedle sebe a pokračuje na dalších mikrořádcích pod sebou, dokud má v řetězci AX nějaké "znaky". Tu-šíte asi, že zakopaný pes v tomto příkazu je v řetězci AX. Jeho obsah lze určit tak, že si nakreslíme zobrazovaný motiv (např. písmeno č) do bitové masky podobné té, kterou jsme si vysvětlili u příkazu FILL (nejméně významný bit-0 je vlevo, nejvyšší bit-5 je vpravo). Pro jednoduchost vynecháme nuly v bitové masce, aby byl zřetelnější tvar budoucího znaku

pozice bitu	0	1	2	3	4	5	
váha bitu	1	2	4	8	16	32	součet vah
1. mikrořádek		1		1			10
2.			1				4
3.		1	1	1			14
4.	1				1		17
5.	1						1
6.	1						1
7.	1				1		17
8.		1	1	1			14

Nyní už stačí převést váhy jednotlivých mikrořádků do řetězce, a jsme hotovi

AX=CHR\$(10)+CHR\$(4)+CHR\$(14)+CHR\$(17)+CHR\$(1)+CHR\$(1)+CHR\$(17)+CHR\$(14)

Příkaz CHR\$(K) jsme si doposud také nevysvětlili - převádí konstantu K (nebo hodnotu proměnné či výrazu) na znak v ASCII kódu (např. "výsledkem" CHR\$(65) je písmeno A). Protože součty vah v našem příkladu neodpovídají zobrazitelným znakům, musíme řetězec sestavit nikoli ze znaků, ale ze součtu 'charakterů'. Protože už umíme pracovat s daty, bude snazší vytvořit AX takto:

```
10 GCLEAR:AX="":DATA10,4,14,17,1,1,17,14
20 FOR I=1 TO 8:READ K:AX=AX+CHR(K):NEXT I
30 BMOVE 5,125:BPLOT AX,1
40 END
```

Spuštěním programu se přesvědčíme, zda jsme nový znak zakódovali správně. A protože jsme zvědaví, co dělá konstanta N v příkazu BPLOT AX,N zadáme řádek:

```
35 BMOVE 7,125:BPLOT AX,8:BMOVE 7,130:BPLOT AX,4
```

Druhý parametr určuje, kolik 'charakterů' z řetězce se vykreslí vedle sebe. Je to proto, aby bylo možno vykreslovat i větší obrazce. Pokud se setkáme s požadavkem na vertikální pohyb takto definovaného obrazce, musíme jej rozsvítit, zhasnout a posunout. A tuto činnost opakovat v cyklu, což ovšem vede k blikání nepříjemnému pro oči. Vhodnější je vytvořit rozšířený obrazec ve směru pohybu, který 'zachová' svět dosud svítících bodů, které mají svítit i po posunu obrazce, rozsvěcí nové body (vpředu) a zhasíná ty, které již svítit nemají (vzadu). Ukážeme si to na dalším programu, který bude pohybovat malým trojúhelníkem:

```
10 GCLEAR X*=CHR*(12)+CHR*(30)+CHR*(63) :REM Obrazec
20 Y*=CHR*(12)+CHR*(18)+CHR*(33)+CHR*(63):REM Jeho obalka
30 BMOVE 10,240:BPL0T X*,1 :REM Vychodi pozice
40 FOR Y=239 TO 0 STEP -1
50 BMOVE 10,Y:BPL0T Y*,1:NEXT Y:END :REM Pohyb obalky
```

Když už je nám jasná funkce obou příkazů, pokusíme se je aplikovat na psaní českých textů na obrazovce. Nebojte se, že budeme pro každou samohlásku vytvářet řetězec pro její podobu s čárkou nebo háčkem či kroužkem, a obdobně pro 'háčkované' souhlásky. Uděláme to jednodušeji, ale za toto zjednodušení musíme něčím zaplatit - každý kompromis něco stojí. V textu, který budeme chtít napsat 'česky', musíme za každé čárkované písmeno (malé či velké) napsat apostrof ', a za každé háčkované nebo kroužkované obrácený apostrof '. Výpis na obrazovku budeme provádět znakově ('charakterově'). Když bude znakem apostrof (kód 39) nebo obrácený apostrof (kód 96), vrátí se výpis o znak zpět (vlevo) a dokreslí nad stávající znak odpovídající znaménko - místo háčku nad velkými písmeny to budou dvě tečky, místo čárky jedna tečka. Pak pokračuje výpis dále. Je samozřejmě pomalejší než pomocí jiných příkazů, ale to jsou ty kompromisy. A nyní vlastní program:

```
10 GCLEAR:REM Demoprogram psani ceskych textu
15 FORI=1TO5:FORM=0TO2:READA:X*(I)=X*(I)+CHR*(A):NEXTM:NEXTI
20 DATA16,8,0,16,0,0,20,8,0,20,0,0,8,20,8
25 DATA3,3,"Ha'c'ky, C'a'rky i krouz'ky nad U' i u' "
30 DATA3,8,"C'a'rka, tec'ka, ha'c'ek, dvojtec'ka, krouz'ek
35 GOSUB55:GOSUB55:END
39 REM Carka se vytvori ze znaku ' (ASCII 39) za pismenem
40 IFY<850RY=105THENBPL0TX*(2),1:RETURN:REMtecka misto carky
45 BPL0TX*(1),1:RETURN:REMcarka
55 READS,R,A*:L=LEN(A*):FORN=1TOL:X=ASC(MID*(A*,N,1))
60 IFX=39ORX=96THENBMOVEK+S,R*9+1:GOSUBX:NEXTN
65 K=K+1:Y=X:PRINTATR,K+S;CHR*(X):NEXTN:K=0:RETURN
95 REM Hacek a krouzek je ze znaku ` (ASCII 96) za pismenem
96 IFY<850RY=117THENBPL0TX*(5),1:RETURN:REM hacek
97 IFY>91THENBPL0TX*(3),1:RETURN:REM hacek
98 BPL0TX*(4),1:RETURN:REM dvojtecka misto hacku
```

TEMA: Datová pole a práce s nimi

Lekce: 16

=====

=====

NOVE POJMY: Deklarace (dimenzování) polí,  
-----  
uchování polí na kazetě,  
načtení datových polí z kazety

NOVE PŘIKAZY: DIM -deklarace datového pole;  
-----  
DSAVE -uchování dat na kazetu,  
DLOAD -načtení dat z kazety;  
CLEAR -vynulování proměnných.

Počítat můžeme využívat jako kalkulačku pro zpracování zadávaných údajů, můžeme jej používat pro různé kombinace a variace s daty uloženými v programových řádcích DATA - čísla nebo texty (řetězci ze znaků). Pro načtení takových dat do programu postačují příkazy INPUT... nebo READ... Rada programů však vyžaduje velmi rozsáhlá pole dat, která jsou vytvořena pomocí jiných programů, jež je nutno aktualizovat, nebo která si mohou programy mezi sebou předávat.

Představme si program 'SLOVNIK', který má data česká v jednom poli a anglická v poli druhém tak, aby si slovíčka pod stejnými pořadovými čísly odpovídala. Práci se slovíčky zajistí odpovídající program. Podle místa v paměti může být ve slovníku jen omezený počet (pole) slovíček. Co se stane, až budeme všechna slovíčka umět. Musíme opsat program a umístit do něho jiné pole slovíček? Ne. Postačuje mít pole organizováno tak, aby je bylo možno samostatně vytvořit, uchovat na kazetě a kdykoli nahrát zpět do paměti. A zdaleka nejlepší je, když bude pole dat rozděleno na dvě - české a anglické, neboť to druhé lze kdykoli nahradit polem slovíček v jiném jazyce. Pokusíme se takovému programu přiblížit, i když ve značném zjednodušení.

Pokud počet proměnných polí nepřekročí v BASIC-G počet 11 (0-10) rezervuje si pro ně místo v paměti program. Pro rozsáhlejší pole indexovaných proměnných je nutno paměť předem vymezit pomocí dimenzování (deklarace) pole

DIM proměnná (max. indexy)

Proměnné mohou být jednorozměrné  $A(i)$ , dvojrozměrné  $Bx(j,k)$  nebo i trojrozměrné  $C(i,j,k)$ . Indexy se uvádějí od 0 do maximální hodnoty, uvedené v příkazu DIM  $Bx(Jmax,Kmax)$ . Každé pole může být deklarováno jen jednou, chybná deklarace způsobí chybu typu (++) Arr.allog ++). V jediném příkazu můžeme deklarovat i více polí:

Například příkaz: DIM A(40),B\$(19,9),C(5,5,5) vymezi:

DIM A(40) -jednorozměrné pole číselné proměnné o 41 prvcích;  
DIM B\$(19,9)-dvojrozměrné textové pole o 20 \* 10 prvcích.  
DIM C(5,5,5)-trojrozměrné číselné pole o 6 \* 6 \* 6 prvcích

Uvedeme si jednoduchý program pro zkoušení anglických slovíček:

```
10 GCLEAR: DIM A$(50),C$(50) :REM Dve pole pro slovíčka
15 DATA 9,CAT,DOG,TOWN,BEER,HOUSE,SUN,RETURN,CALL,SAVE
20 DATA KOCKA,PES,MESTO,PIVO,DUM,SLUNCE,NAVRAT,VOLANI,UCHOVAT
25 READ N:PRINT"Slovník má"/N,"+"N;" slovíček"
30 FOR I=0 TO N-1:READ A$(I):NEXT I:REM Naplnění anglického
35 FOR I=0 TO N-1:READ C$(I):NEXT I
40 PRINT"Zapamatujte si:"FOR I=0TO N-1:PRINTC$(I),A$(I):NEXT I
45 ?"Pokud je umíte, stlačte EOL":GCLEAR
50 PRINT"Zkouska: napiste anglicky ekvivalent":SP=0
55 FOR I=0TO N-1:PRINT C$(I),"=":"REM Nabídka českého slova
60 INPUT S$:IF S$=A$(I) THEN SP=SP+1:GOTO 70:REM Správně
65 PRINT,,"Ma být:"A$(I) REM Chyba
70 NEXT I:PRINT,"Je to ":"IF SP>7 THEN PRINT"vyborne":GOTO90
75 IF SP>5 THEN PRINT"dobre":GOTO90
80 IF SP>3 THEN PRINT"vyhovujici":GOTO90
85 PRINT"nevyhovujici"
90 END
```

Program je jednoduchý a proto snad i srozumitelný, vyzkoušejte si jej a potom si jej upravte tak, aby bylo možno naplnit pole dvojicemi slovíček:

```
30 FOR I=0 TO 50: GOSUB 95:NEXT I
35 PRINT"Slovník je naplněn":N=51

95 DISP"Anglicky...":INPUT A$(I):DISP"Cesky...":INPUT C$(I)
97 DISP"Je to spravne? A/N (EOL)":INPUT D$:IF D$="N" THEN 95
99 RETURN
```

Pomocí podprogramu pro dialogový řádek jsme vytvořili jednoduchý návod pro obsluhu programu i opravu zadávaných slovíček. Správné napsání netestujeme (jen "N"),proto není nutné psát A a EOL,stačí jen stisk EOL. Potlačíte-li pomocí SHIFT+PRINT vypisování na obrazovku, musíte před RETURN zařadit opis obou slovíček.

Naplnili jsme si malou jednoduchou databázi slovíčky, i tak to bylo pracné. Proto bychom se měli naučit uchovávat pole dat na kazetu. BASIC-G má pro tento případ užitečný příkaz

DSAVE N:A\$(min) "Nazev

umožňující uložení prvků všech typů polí na vnější magnetické medium - kazetu. Pořadové číslo záznamu je N<99 (identifikátor na pásce). A\*(min) pro textové proměnné, nebo např. B(Imin,Jmin) pro dvojrozměrné pole číselných proměnných, musí být první prvek pole v pořadí (s nejnižším indexem). Název záznamu nemusí být uveden, ale je-li, musí být za uvozovkami! (++Syntax error++).

Pole, které chceme pomocí DSAVE zaznamenat, musí být v programu vymezeno příkazem DIM... Záznam je typově odlišen od jiných písmenem '/D' (Data). Pro náš program bychom mohli záznam provést takto:

```
100 ?"Zapni magnetofon pro záznam a stlač EOL."
105 DSAVE 2;A*(0) "Anglicky
110 DSAVE 3;C*(0) "Česky
115 ?"Zastav magnetofon a stlač EOL."
```

Na obal kazety je dobré si poznačit vedle údaje počítadla otáček a pořadového čísla nahrávky také její jméno a především velikost vymezené paměti i první prvek pole, např.

```
035 02/D Anglicky -A*(50),A*(min)
```

Opakem záznamu dat je jejich načtení do paměti počítače pomocí příkazu

```
DLOAD N;A*(min)
```

kde: N je pořadové číslo záznamu na kazetě;  
A\*(min) je první prvek pole (s nejmenším indexem).

Pokud chceme pole načíst do programu, musí být jeho prvky předem vymezeny příkazem DIM..., neboť počítač srovnává rozměr pole na kazetě s rozměrem pole v programu (++File small++). To je proto, aby nedošlo k nežádoucímu přemazání ostatních proměnných v paměti. Příslušná část programu by měla přesně informovat uživatele:

```
200 DIM A*(50),C*(50): ?"Pust magnetofon a stlač EOL"
205 DLOAD 2;A*(0)
210 DLOAD 3;C*(0)
215 ?"Zastav magnetofon a stlač EOL"
220 FOR I=0 TO 50:PRINT A*(I);:NEXT I:REM Kontrolní opis dat
225 END
```

Pokud se nám nahrávka nezdaří, je nutno ji zopakovat. Protože není možné dvakrát vymezovat stejné pole, je nutné použít pouze příkazu DLOAD... (například spuštěním programu od řádku 205 pomocí GOTO 205).

Pokud nemáme jiná pole v programu, lze vymezení zrušit příkazem CLEAR (nulování proměnných) a po něm použít příkaz GOTO 200 nebo RUN 200 (ten již bez CLEAR, neboť RUN také nuluje proměnné a ruší deklarace).

POZOR.  
=====

Při práci s datovými poli je nutná velká opatrnost, aby se tato pole nepoškodila. Nezbytností je, aby jakákoliv práce programem nebo jeho ukončení končily zprávou 'OK' v dialogovém řádku. Pokud to nezajistíme, přiděláme si řadu starostí jako autoři programu (o problémech uživatelů raději pomlčet).

Proto doporučujeme tento postup:

- a/ každý program musí být ukončen příkazem END;
- b/ po chybě v programu a jeho zastavení s výpisem chyby je nutné zadat přímým příkazem také END, aby se vypsalo 'OK', před libovolnou manipulací s programem;
- c/ při zastavení programu klávesou STOP je nutné také standardní ukončení pomocí END, pokud nepokračujeme příkazem CONT;
- d/ příkaz pro uchování dat DSAVE musí být zadán správně syntakticky, jinak dojde také ke zničení pole dat;

Při nedodržení těchto doporučení nelze pak po nestandardním zakončení programu ani pomocí DSAVE data uchovat, neboť již nejsou uložena v paměti.

- Cvičení: 1/ Zkuste náhodný výběr slovíček pro zkoušení aplikací  
----- funkce RND.  
2/ Jak vynechat slovíčka, která byla správně zodpovězena?

TEMA: Uživatelské funkce a jejich využití  
=====

Lekce 17  
=====

NOVE POJMY. Definování uživatelské funkce,  
----- používání těchto funkcí,  
přeměna řetězce a výrazu na číslo,  
formátování informací na obrazovce.

NOVE PRIKAZY DEF FNC A(X) - definice uživatelské funkce;  
----- FNC A(X) - použití funkce v programu;  
VAL (Tx) - přeměna číselného řetězce na číslo;  
VAL(výraz) - přeměna výrazu na číslo;  
PRINT INK(K) - změna způsobu zobrazení informace.

Pokud při tvorbě programu nevystačíme se standardními funkcemi, umožňuje BASIC-G vytvářet si další funkce podle požadavků zadání programu. Tyto tzv. uživatelské funkce (definované programátorem pro uživatele) mají formát

FNC A(X)

kde A-jméno funkce, tvořené jedním z 26 abecedních znaků (A-Z);  
X-formální parametr funkce zadáný jednoznakovou proměnnou.

Hodnota skutečného parametru se do funkce zadá během zpracování programu. Definování uživatelské funkce se provádí příkazem DEF, například:

DEF FNC B(Z) = Z + 1/Z +Z^2

Tento příkaz je sám o sobě nevykonný a není-li volán svým jménem není vyhodnocen a proto může být uveden kdekoliv v programu. Ale proč jej tam dlouho hledat, když může být hned na počátku, definován společně s výčtem používaných proměnných (ty se také definují v programu až při prvním použití) a dimenzováním datových polí. V pravé části definice funkce je výraz, který se vyhodnotí pro konkrétní hodnotu proměnné; např.

K=FNC B(2); proměnná bude mít hodnotu K=6.5  
L=FNC B(3); L=12.3333

Funkce je definována pomocí jediné proměnné, ale její použití je obecné pro libovolnou proměnnou, jak dokumentuje tento program, který hledá řešení rovnice  $E(X)=0$  metodou přilehlí intervalu A,B za předpokladu, že má rovnice v tomto intervalu řešení.

```

10 DEF FNC E(X)=2 + (EXP(X)-EXP(-X))/2
15 PRINT"Zadej interval (od) A":INPUT A
20 PRINT"Zadej interval (do) B":INPUT B
25 PRINT"Zadej presnost reseni C":INPUT C
30 S=(A+B)/2
35 IF ABS(FNC E(S))<C THEN 50
40 IF FNC E(S)*FNC E(A) <0 THEN B=S:GOTO30
45 IF FNC E(S)*FNC E(B) <0 THEN A=S:GOTO30
50 PRINT"Reseni: X=";S:END

```

Uživatelská funkce je vhodná i tam, kde se proměnné nemění pravidelně a není možné řešit úlohu cyklem, nebo tam, kde další hodnoty jsou počítány z předcházejících výsledků a pod. Především ale šetří místo v paměti, neboť se definuje jen jednou (obdobu podprogramu).

Další vhodnou funkcí je přeměna řetězce čísel nebo výrazu na číslo pomocí

VAL (X)     nebo     VAL (výraz)

S výsledkem lze pracovat pomocí všech aritmetických operátorů a funkcí. V řetězci se nejprve přeskočí všechny mezery zleva a číslo se sestavuje tak dlouho, pokud má v BASIC-G smysl. Zbytek řetězce se zanedbává. Proto je např. VAL("")=0.

Zde je na místě uvést, že přesné součty více jak šestimístných čísel lze v BASIC-G provádět pouze pomocí řetězců a funkce VAL. Zkontrolujte, jestli bude součet  $9\,999\,999 + 555 = 10\,000\,554$ .

```

10 GCLEAR:AX="9999999":BX="555":P=0
12 A=LEN(AX):PRINT"AX="TAB(20-A)AX;TAB(40-A)AX
14 B=LEN(BX):PRINT"BX="TAB(20-B)BX;TAB(40-B)BX
16 PRINT TAB(20-A);:GOSUB50:PRINT TAB(40-A);:GOSUB50
18 IF A=B THEN 24:REM Jsou-li delky čísel stejné
20 IF A>B THEN N=A-B:FOR I=1 TO N:BX="0"+BX:NEXT
22 N=B-A:FOR I=1 TO N:AX="0"+AX:NEXT:REM Doplnění nul zleva
24 FOR I=LEN(AX) TO 1 STEP-1:REM Počítáme zprava doleva
26 S=VAL(MID$(AX,I,1))+VAL(MID$(BX,I,1))+P: P=0
28 IF S>9 THEN P=1:REM Nastavení přenosu P, vznikl-li
30 SX=STR$(S):DX=RIGHT$(SX,1)+DX:NEXT:REM Převod na text
32 IF P=1 THEN DX="1"+DX:PRINT:REM Přenos v nejvyšším řádu
34 PRINT"Suma="TAB(18-A)VAL(AX)+VAL(BX); " ? ?";
36 PRINT TAB(40-LEN(DX))DX:PRINT"Srovnej výsledky"
38 END
50 FOR I=1 TO A:PRINT"-";:NEXT:RETURN:REM Podtrhnutí

```

Zarovnávaní celých čísel vpravo je možné pomocí převodu čísel na texty (řetězce) funkcí STR\$, určení délky textu funkcí LEN(AX) a jeho vypsání pomocí tabelátoru: PRINT TAB(20-LEN(AX))AX. Tak je to i v uvedeném programu.



Aby bylo možno sečítat čísla v textech, je nutno zajistit stejnou délku čísel doplněním nevýznamných nul zleva (řádky 20,22). V cyklu (24-30) se provádí součet jednotlivých pozic čísel zprava, připočte se přenos P, vynuluje se, aby se mohl obnovit při  $S > 9$ . Součet pozic S se převede na text S\*, z něhož se do výsledného součtu použije jen pravá číslice, ne přenos, který se v nejvyšším řádu připočte až po skončení cyklu.

Protože funkce VAL(F\*) může přeměnit na číslo také výraz zadáný jako text (např.  $F* = X + 1/X + X*X$ ), uvedeme si program, který umožní vytvořit tabulku hodnot pomocí funkcí, zadaných "vstupem" INPUT (Program 'neví', podle jakých funkcí bude počítat a vypisovat.)

```
10 GOCLEAR:PRINT"Vypocet tabulek pro zadane operace"
12 PRINT"Zadej 3 matematicke operace s promennou I"
14 GOSUB60:F*=A*:GOSUB60:G*=A*:GOSUB60:H*=A*
16 PRINT AT3,0:GOSUB50
18 PRINTTAB(10)"F*=";TAB(24)"G*=";TAB(38)"H*="
20 PRINTTAB(2)"I=";TAB(10)F*;TAB(24)G*;TAB(38)H*:GOSUB 50
22 FOR I=1TO15:I*=STR*(I):J=LEN(I*):PRINTTAB(4-J);I;
24 A*=STR*(VAL(F*)):A=LEN(A*):PRINT TAB(12-A);A*;
26 A*=STR*(VAL(G*)):A=LEN(A*):PRINT TAB(26-A);A*;
28 A*=STR*(VAL(H*)):A=LEN(A*):PRINT TAB(40-A);A*;
30 NEXT:GOSUB50:END

50 FORI=1TO48:PRINT"--";NEXT:PRINT:RETURN
60 PRINT AT24,0"Zadej funkci promenne I pomocí operatoru:"
62 PRINT AT25,0"Napr.: I+1/I nebo I+EXP(I) a pod."
64 INPUT A*:PRINT A*:RETURN
```

V programu je využit podprogram pro opakované zadání funkcí proměnné I jako A\*, funkce jsou po volání podprogramu předány proměnným F\*, G\* a H\*. Podprogramem je také podtrháváno. Opakované jsou využity proměnné A\* a A pro výpis výsledků výpočtu na řádku obrazovky, aby se zbytečně počet proměnných nerozšiřoval. Kdo je na takové finty připraven, není pak překvapen při nevhodném chování programu.

- Cvičení: 1/ Upravte program součtu vícemístných čísel na součet  
----- čísel zadaných pomocí příkazu INPUT.  
2/ Vypočtete programem součet a součin všech číslic vícemístného čísla zadaného pomocí INPUT.

TEMA: Co jsme dosud neprobrali  
=====

Lekce: 18  
=====

NOVE POJMY: Umístění proměnných v paměti počítače,  
----- čtení obsahu adres, zápis nového obsahu,  
převod desítkového čísla na šestnáctkové.

NOVE PRIKAZY: ADR(X),ADR(X\*)-hodnota adresy uložení proměnné;  
----- PEEK, APEEK -čtení obsahu jedné/dvou adres;  
POKE, APOKE -zápis bytu/dvou bytů na adresu;  
HEX\* -převod desítkové šestnáctkový;  
FRE(X),FRE(X\*)-volné místo v paměti pro proměnné.

Pro informaci o programu a jeho rozmístění v paměti počítače je někdy užitečné znát nejen hodnoty proměnných, ale i jejich adresy v paměti počítače (v zóně programu nebo mimo ni). K tomuto účelu jsou v BASIC-G příkazy/funkce.

ADR (X) a ADR (Y\*)

které umožňují zjištění adres uložení proměnných. Například:  
A1=ADR(X) - umožňuje přečíst do A1 adresu uložení proměnné X;  
A2=ADR(C\*)- načte do A2 adresu uložení proměnné C\* tak, že v jednotlivých bytech můžeme zjistit hodnoty:  
1.byte -délka textu v řetězci (0-255);  
2.byte -nepoužit, má hodnotu 255;  
3-4.byte-adresa začátku textu.

Protože pro pochopení je nejlepší příklad, tady je:

```
10 GCLEAR:A=5:DIM B(9):C=-5:D=.1
15 PRINT"ADR(A)=";ADR(A)
20 FOR I=0TO9:PRINT"ADR(B(I))=";ADR(B(I)):NEXT I
25 PRINT"ADR(C)=";ADR(C):END
```

Kde jsou uloženy proměnné už víme, ale jak jsou uloženy neznáme.  
Funkce

PEEK (ADR)

nám pomůže obsah adres, na nichž je proměnná uložena, přečíst. Protože je adres pro číselnou proměnnou celkem 6, zkusíme zjistit, co je na jednotlivých adresách uloženo. Zvolíme nejprve dvě proměnné stejné hodnoty, ale různých znamének:

```
10 GCLEAR:A=1:C3=-1
15 M=ADR(A):PRINT"ADR(A)=";M
20 FOR K=M TO M+5:J=PEEK(K):PRINT K;HEX$(K);J;HEX$(J):NEXT K
25 M=ADR(C):PRINT"ADR(C)=";M
30 FOR K=M TO M+5:J=PEEK(J):PRINT K;HEX$(K);J;HEX$(J):NEXT K
35 END
```

Výpis programu je jasný, ale poněkud 'zamlžen' desítkovými čísly která se těžko analyzují. Proto jsme použili také funkci převodu desítkového čísla na šestnáctkové:

HEX\*(K),

kde K=0-32767. Tato funkce převádí celou část parametru -INT(K). zkuste si to přímým příkazem

PI=3.14159:PRINT PI/HEX\*(PI)

Použitím tabulky ASCII kódu v příloze zjistíme:

- název proměnné je v zóně hodnot proměnných na adrese ADR(A)-2 (druhý znak názvu proměnné) a na ADR(A)-1 (první znak jejího názvu), a jsou pro něj vyhrazeny pouze 2 byty; z toho vyplývá, proč sice můžeme mít název proměnné jakýkoliv, ale platné jsou jen dva prvé znaky;
- hodnota proměnné je na 4 bytech ve speciálním binárním kódu.

Hodnotu dvou po sobě jdoucích bytů paměti můžeme zjistit pomocí funkce APEEK(ADR), která nahrazuje použití dvou funkcí PEEK(ADR). Podobným způsobem jako u číselných proměnných můžeme zjistit adresu uložení textových proměnných ADR(T\*) a samozřejmě i způsob uložení textu. Zkuste si následující program

```
10 GCLEAR:A1*="STO":L=PEEK(ADR(A1*)):REM Delka textu= LEN(A1*)
15 A=APEEK(ADR(A1*)+2):PRINT L,ADR(A1*)
20 FOR I=A-5 TO A+2:PRINT I,CHR*(PEEK(I)):NEXT:END
```

Pro řetězcovou proměnnou platí, že její název je uložen na adrese ADR(A\*)-5 v zóně programu (tj. nad adresou 2400), a její obsah je (bez uvozovek) na adrese ADR(A\*). Známe jak číst obsahy adres a tak nezbývá než se seznámit s příkazem zápisu/modifikovaného obsahu na danou adresu:

POKE ADR,seznam

Je-li v seznamu více hodnot (0-255), zapisují se na uvedenou a následující adresy (takto lze například v programu zapsat jeho část ve strojovém kódu, jak bude uvedeno později). Pomocí POKE můžeme vhodně modifikovat systém, ale můžeme také zničit program nebo vyvolat havarii počítače. To je nutné mít na paměti vždy, když chceme tento příkaz použít a především tehdy, když zapisujeme jeho parametry. Podobným příkazem je

APOKE ADR,seznam

lišící se tím, že zapisuje na adresu a adresu+1 hodnotu v rozsahu dvou bytů (0-65535) ze seznamu. Přehlednější je zadávat adresy i data šestnáctkově, i když to pro začínající uživatele bude komplikovanější. Zkusme si něco zapsat a přečíst těmito příkazy:

```

10 GCLEAR: DATA 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15
15 FOR I='7000 TO '700F:READ D:POKE I,D:NEXT
20 PRINT "A=ADR";"PEEK(A)","HEX*(PEEK(A))","HEX*(APEEK(A))"
25 FOR I='7000 TO '700F:PRINT I;PEEK(I),HEX*(PEEK(I));
30 PRINT HEX*(APEEK(I)):NEXT:STOP
35 RESTORE 10:FOR I='7010 TO '702F STEP 2:READ D:APOKE I,D:NEXT
40 PRINT "A=ADR";"PEEK(A)","HEX*(PEEK(A))","HEX*(APEEK(A))"
45 FOR I='7000 TO '700F:PRINT I;PEEK(I),HEX*(PEEK(I));
50 PRINT HEX*(APEEK(I)):NEXT:END

```

Po zastavení programu výpisem ++Stop at ln.30++ a prostudování výsledků práce programem, můžeme pokračovat přímým příkazem CONT

Některé užitečné adresy a hodnoty pro POKE:

POKE '2E,80	změní počet vypisovaných znaků na řádku na 80,
POKE 'C137,255	zamezí opisování dialog.řádku na obrazovku;
POKE 'C1FA,'A8	zobrazování bodu negací předchozího stavu; je to základní režim zobrazení;
,'B0	zobrazení bodu nastavením; bod bude svítit;
,'AF	zobrazení bodu mazáním; bod nebude svítit;
POKE 'C03A, 00	základní zobrazení: bílé znaky, tmavé pozadí;
,'40	zobrazení polojasem;
,'80	zobrazení blikáním při plném jasu;
,'C0	zobrazení blikáním s polojasem.

Funkce FRE(X) umožňuje zjistit zbývající volnou paměťovou oblast pro program a číselné proměnné. Je používána pro kontrolu, zda máme ještě volnou paměť při psaní dlouhého programu, nebo při dimenzování rozsáhlých datových polí (předejdeme tak hlášení ++Pg too big++). Největší hodnoty volné paměti jsou (před zápisem programu)

PRINT FRE(X)	Výsledek: 14830
PRINT FRE(X*)	3841

a pozor, nejmenší nejsou nulové, ale nad 100 bytů, závisí na použitých instrukcích v programu (cykly a podprogramy), což se neprojeví při zápisu programu, ale až po jeho spuštění. A co potom dělat?

- zhustit zápis příkazů vynecháním mezer mezi příkazy a parametry
- zapisovat více příkazů na řádek, oddělených dvojtečkou;
- nepoužívat čísla řádků nad 10000, především sem neumísťovat volané podprogramy (za GOSUB je číslo řádku znakově, ne binárně);
- více využívat podprogramy - už při dvojnásobném opakování stejných příkazů v některých programových řádcích;
- zamyslet se, zda použitý algoritmus nelze zjednodušit;
- prověřit množství proměnných, nejsou-li platné po celou dobu činnosti programu, používat je vícenásobně.

TEMA: Vstupní a výstupní obvody počítače.

Lekce: 19

=====

=====

NOVE POJMY: Vstupní a výstupní brána (kanál),  
----- řídicí slovo a stavové slovo brány,  
příkazy pro vstup a výstup dat.

NOVE PŘIKAZY: CONTROL - zápis řídicího slova brány;  
----- STATUS - čtení stavu brány;  
OUTPUT - výstup dat;  
OUT - výstup jednoho znaku;  
ENTER - vstup dat z brány do proměnné;  
INPUT#K; - vstup dat z kanálu do proměnných;  
INP - vstup jednoho znaku;  
BIT - určení hodnoty bitu v bytu;  
WAIT - čekání na splnění podmínky....

V lekci 10 úvodního kursu jsme se seznámili s prvními příkazy pro práci s vnější kazetopáskovou pamětí (SAVE, LOAD, CHECK), lekce 16 tohoto kursu nám přiblížila příkazy pro zápis a čtení dat (DSAVE, DLOAD). Spolupráce počítače s přídatnými zařízeními, která jsou k němu připojena přes některý interface (obvod mezistyku), se provádí přenosem dat přes určenou vstupně výstupní (v/v) bránu (kanál). Tyto brány jsou osazeny standardními v/v integrovanými obvody typu:

- 8251 - univerzální seriový přijímač a vysílač dat (USART);
- 8253 - trojnásobný čítač nebo časovač (CTC);
- 8255 - tříkanálový paralelní v/v obvod (PIO).

Tyto obvody zčásti nebo úplně řídí přenos dat mezi počítačem a připojenými zařízeními (tiskárna, zapisovač, síť, joystick, modem), proto jsou programovatelné - lze definovat jejich vlastnosti a funkce. Programování se provede zápisem jednoho nebo více řídicích slov do jejich řídicího registru. Po jeho naprogramování je možno komunikovat přes v/v bránu s přídatným zařízením, zapisovat nebo číst data, zjistit stav (STATUS) v němž se obvod nachází a pod.

Ovládání v/v bran je možné ve dvou základních úrovních:

- logické (vyšší) pomocí příkazů a funkcí BASIC-G, kdy se místo skutečných adres v/v obvodů používají symbolická čísla kanálů;
- fyzické, kdy se ovládání v/v obvodu musí zabezpečit programem na úrovni strojového kódu procesoru a proto se používají skutečné adresy obvodů.

V následující tabulce je uveden způsob adresace v/v obvodů.

Kanál	Reg.	Adresa	Brána	Obvod	Funkce
k=1	r=0	'10-28	DATA	8251	Magnetofon, sériový přenos
	=1	'1D-29	CWR, STATUS		Rídící a stavový registr
k=5	r=0	'5C-92	CT0	8253	Časovač pro magnetofon
	1	'5D-93	CT1		Univerzální čítač/časovač
	2	'5E-94	CT2		Univerzální
	3	'5F-95	CWR		Rídící registr
<hr/>					
Systém:		4	PA	8255	Rízení klávesnice
		5	PB		Čtení stavu klávesnice
		6	PC		Konfigurace, akust. výstup
		7	CWR		Rídící registr
<hr/>					
k=4*)	r=0	'4C-76	PA	8255	Univerzální v/v
	1	'4D-77	PB		Paralelní tiskárna
	2	'4E-78	PC		Bitové vstupy/výstupy
	3	'4F-79	CWR		Rídící registr
<hr/>					
		'48-72	**) )	8255	Čtení EPROM intel. kabelu
		'49-73		----	Adresa term. intel. kabelu
<hr/>					
2. kabel		8	**) )	8255	Čtení EPROM
		9		----	Čtení adresy terminálu
		'0C-12	PA	8255	
		'0D-13	PB		
		'0E-14	PC		
		'0F-15	CWR		

\*) Tento kanál není v C2717 standardně zapojen, dodává se v tzv. inteligentním kabelu pro připojení paralelních tiskáren atd.

\*\*) Adresa EPROM je zapisována rozdělena přes PA (dolní byte) a PB (horní byte adresy).

Z této tabulky zjistíme adresy obvodů, ovšem co na tyto adresy budeme posílat, to je mnohem komplikovanější, protože pro každý obvod platí jiná řídící slova i jiný algoritmus ovládání (viz např. AKTUALITY C 2717-1,2,3). Stručné tabulky pro programování obvodů jsou uvedeny v příloze. Proto se seznámíme jenom s nejdůležitějšími a často používanými příkazy a řídícími slovy:

Pro zápis řídících údajů do registrů v/v se používá:

CONTROL k,r/n1,n2 - kde: hodnoty k a r jsou v uvedené tabulce;  
hodnoty 'ni' jsou požadované funkce v/v  
CONTROL 1,1/64,78,55 64-vnitřní nulování 8251;  
78-mod. asynchronní režim, taktování 16\*,  
8 bitů dat bez parity, 1 stop bit;  
55-povel zahájení přenosu: DTR, RTS, TxEN,  
RxEN a nulování chyb PE, OE, FE.

```
CONTROL 4,3;132,5      132-nastavení 8255 pro strobovaný výstup
                        dat přes PB (HANDSHAKE);
                        5-nastavení PC2=1 pro přerušení INTEB
                        (podmínka pro programové ošetření)
                        v režimu strobování
CONTROL 5,3;52          52-nastaví mod 2 čítače CT0 v 8253;
CONTROL 5,0;0,1         0-dolní byte '00 konstanty '0100;
                        1-horní byte '01 konstanty '0100.
```

Pro zápis dat do datových registrů v/v se používá příkazů, které mají zabudovaný testy připravenosti pro výstup dat:

OUTPUT 1;5	výstup 'znaku' 5 (ENQ) seriovým kanálem;
OUTPUT 1;A×	výstup textu z proměnné A× přes 8251;
OUTPUT 404;B×	tisk obsahu B× přes PB 8255 na tiskárnu;
OUTPUT 403;seznam	tisk hodnot ze seznamu přes PA 8255;
PRINT #404;B×	je obdobou OUTPUT 404;B× (stejná 'rutina');
LIST #404,	vytiskne 'listing' programu přes PB 8255.

Pro zápis jednotlivých znaků (bytů) do v/v registrů lze použít:

```

OUT 29,55      zahájení přenosu přes 8251;
OUT 28,6       výstup 'znaku' 6 (ACK);
OUT 79,132     nastavení PB v 8255 pro výstup dat;
OUT 77,13      výstup 'znaku' CR (návrat tiskací hlavy);
OUT 7,11       změna matice zobrazovaných znaků na 8*9;
OUT 7,10       návrat do původního stavu-konfigurace systému.

```

Pro čtení stavové informace kanálu (chyby přenosu, stav signálů) do proměnné lze využít příkazy:

```
S=STATUS 1,1      načtení stavu sériového přenosu přes 8251;
PC=STATUS 4,2     načtení stavu brány PC obvodu 8255;
```

Ekvivalentními příkazy k uvedeným příkladům STATUS jsou:

S=INP(29)            vstup znaku z brány o adrese 29 (8251);  
PC=INP(78)           vstup znaku z portu PC obvodu 8255.

Pro čtení dat z v/v obvodů, se zajištěným testem připravenosti těchto dat, je možno využít příkazů:

```

ENTER 1;P      vstup znaku z 8251 do proměnné P;
ENTER 402;Zx   vstup jednoho znaku z PC obvodu 8255 do Zx;
ENTER 403;Px   načtení dat v modu 1 (HANDSHAKE) PA do proměnné
                Px (data musí končit 'znaky' CR+LF = '0D0A'); je
                to obdobou zpracování příkazu INPUT Px;

```

INPUT #403;seznam      vstup dat v modu 1 přes PA do seznamu proměnných (číselných nebo textových);  
INPUT #404;K            vstup dat v modu 1 přes PB do proměnné.

Pro testování pravdivostní hodnoty jednotlivého bitu v bytu má BASIC-G příkaz-funkci:

BIT S,P                P- pozice bitu zprava (0-7);  
                         S- jednoznačová proměnná- např. STATUS (0-255).

Tímto příkazem si zkuste vypsat dvojkové kód zvoleného znaku:

```
10 GCLEAR:PRINT"Zadej znak" INPUT A$  
20 A=ASC(A$):PRINT A$;"=" "/A;  
30 FOR I=7 TO 0 STEP-1:PRINT BIT A,I;  
40 NEXT I:END
```

Obdobou funkce BIT ve vztahu ke kanálu počítače je funkce WAIT, která pozastaví program až do doby nastavení nebo shození určitého bitu nebo skupiny bitů zadané brány/kanálu počítače. Definuje se jako

WAIT P,M;K

kde P - fyzická adresa portu/kanálu (0-255);  
M - maska vyjadřující pozice testovaných bitů tohoto portu (0-255), měnících se z log.0 do log.1;  
K - maska (0-255), udávající, které bity mají být testovány na změnu hodnoty z log.1 do log.0.

WAIT 29,2      - testuje přijetí dat (RxRDY) v MHB 8251A;  
WAIT 78,1      - testuje změnu bitu PC0 z log.0 na log.1 (INTRb);  
WAIT 78,4,4    - testuje změnu bitu PC2 z log.1 na log.0 (-ACK);

Cvičení: 1/ Zkuste si, co udělá příkaz: OUT 7,11 (systémový PC).  
-----      Návrat do původního stavu je příkazem OUT 7,10.



TEMA: Používání příkazů MONITORU počítače

Lekce: 20

NOVE POJMY: Základní programové vybavení počítače,  
----- module pro komunikaci s mikropočítačem,  
výpis a modifikace obsahu paměti,  
nahrávání programů ve strojovém kódu.

NOVE PŘIKAZY: DUMP - šestnáctkový a znakový výpis paměti;  
----- MEM - výpis paměti do dialogového řádku;  
SUB - modifikace obsahu paměti;  
JUMP - skok na začátek programu ve strojovém kódu;  
MGLD - čtení programu ve strojovém kódu z kazety;  
MGSV - záznam programu ve strojovém kódu na kazetu;  
MGEND - ověření správnosti záznamu kontrolním čtením

Základní programové vybavení, které umožňuje uživateli i vyšším programovacím jazykům komunikaci s technickými prostředky počítače, se nazývá OPERAČNÍ SYSTÉM. Jeho nejjednodušší varianta bývá nazývána MONITOR a je obvykle tvořena mnoha podprogramy ve strojovém kódu procesoru. Fyzicky je monitor umístěn v pamětech typu EPROM (Eraseble Programmable Read Only Memory - mazatelné programovatelné paměti určené pouze ke čtení) - v C2717 na adresách od 18000 výše v délce 4 kB.

Základní funkcí monitoru C2717 je inicializace systému po zapnutí počítače a předání řízení interpretu jazyka BASIC-G. Návrat do monitoru je umožněn současným stlačením kláves SHIFT+RCL, po němž se v dialogovém řádku vypíše zpráva

+ + Os ready + +

Příkazů pro práci v režimu monitor je pouze několik, pokud jsou zadány správně, provedou se, v opačném případě oznámí v dialogovém řádku některou z chyb, nebo systémových hlášení:

++Err. in adres++	Chyba zadání adresy (4-místná, šestnáctková)
++Err. in data++	Chyba v zadání dat (šestnáctková, sudý počet)
++File error++	Chyba čtení dat z magnetofonu
++Memory overflow++	Přeplněna paměť v oblasti klíčů F0-F11
++No command++	Neznámý příkaz v režimu monitor
++No key++	Stisknutý klíč F1 nemá zadán obsah
++Executive++	Spuštěn program ve strojovém kódu
++Mg stop !++	Ukončeno nahrávání, zastavte magnetofon

Nejčastěji používaným příkazem je výpis paměti: DUMP adresa, kde adresa musí být šestnáctková, čtyřmístná a v rozsahu 0000-FFFF (nepoužívá se apostrof). Provedením příkazu je výpis paměti od zadané adresy po řádcích ve tvaru: adresa-obsah HEX-obsah ASCII.

Například DUMP 8427 začne vypisovat:

```
8427 4F 73 20 72 65 61 64 79      Os ready
842F .....
8437 .....
```

Výpis lze pozastavit pomocí SHIFT nebo zastavit klávesou STOP. Od adresy 0000 se vypisuje interpret BASIC-G, od adresy 2400 se vypíše program v BASIC-G (pozor, příkazy jsou zakódovány), pokud byl do počítače zaveden, od 8000 se vypíše obsah monitoru a od C000 obsah videopaměti.

Pro změnu obsahu paměti se používá sice příkaz SUB adresa, ale je vhodnější použít příkaz MEM adresa, který do dialogového řádku vypíše stávající obsah 16 bytů od zadané adresy (SUB nevypíše nic), a máme možnost zkontrolovat, jaká data budeme modifikovat. Zápis nového obsahu je šestnáctkovými číslicemi (např. program ve strojovém kódu), číslic musí být sudý počet (dvě tvoří byte). Zkuste si zadat MEM 7000 a modifikovat výpis takto

```
SUB 7000 20 43 20 32 37 31 37 20      ( a EOL )
```

nabídnou se vám k modifikaci další adresy SUB 7008 (stlačte EOL) a vypíšete si, co jste modifikovali, pomocí příkazu DUMP 7000:

```
7000 20 43 20 32 37 31 37 20      C 2717
7008 .....
```

Pomocí příkazu JUMP adresa můžeme odstartovat program ve strojovém kódu, který jsme od zadané adresy zapsali (SUB) nebo nahráli z magnetofonu. Požadujeme-li návrat do monitoru, musí být tento program zakončen některou z návratových instrukcí. Můžeme si zkusit nejjednodušší program, tvořený pouze návratovou instrukcí C9 (RET). Zadáme MEM 7000 a po výpisu modifikujeme jediný byte

```
SUB 7000 C9      (EOL)
```

"Program" spustíme příkazem JUMP 7000, problikne ++Executive++ a vypíše se ++Os ready++. Podobným příkazem můžeme předat řízení interpretu BASIC-G zadáme-li JUMP 0000. Návrat do MONITORU je přes SHIFT+RCL.

Protože ve strojovém kódu je řada programů již na kazetách, můžeme takový program nahrát, známe-li číslo jeho záznamu nn pomocí příkazu

```
MGLD nn      (EOL)
```

Program se ohlásí hlavičkou, v níž je jeho číslo/? a název, například:

```
00/? C212.ASM.
```

Z režimu interpretu jazyka BASIC-G lze také přímo nahrávat programy ve strojovém kódu (není nutný přechod do režimu MONITOR). Je k tomu určen příkaz

LOAD CODE nn

kde nn je číslo nahrávky programu (nebo příkaz LOAD CODE 0, který zajistí nahrávku prvního programu nalezeného na kazetě).

Pomocí MGLD lze načíst jen programy pořízené příkazem MGSV nebo nahrané vhodným kopírovacím programem. Spustit takové programy je možno pomocí JUMP adresa (pokud ovšem adresu známe; nebývá tožná s počáteční adresou, od níž se program zavádí).

Neznáme-li číslo nahrávky nn, lze zadat MGLD 00; způsobí načtení prvního nalezeného programu ve strojovém kódu. Správněji bychom měli hovořit o bloku binárních dat. To proto, že když chceme uložit na kazetu obsah části paměti počítače, uděláme to pomocí příkazu

MGSV nn, od-do, název

kde nn - číslo nahrávky na kazetě;

od - počáteční adresa, od níž budeme nahrávat data;

do - koncová adresa, do které jsou data v paměti umístěna;

název - jméno souboru (programu, dat) na kazetě (max. 8 znaků)

Takto bychom mohli uchovat na kazetu i soubor dat a instrukcí jazyka BASIC-G:

MGSV, 0000-2400, Basic-G

Adresy musí být uváděny šestnáctkově.

Kontrolu nahrávky souboru můžeme provést příkazem MGEND nn (nebo MGEND 00 neznáme-li číslo nahrávky). Nahrávka je testována na správnost kontrolního součtu, neprovádí se srovnání s obsahem příslušné oblasti v paměti počítače. Je-li kontrolní součet bezchybný, vymaže se dialogový řádek, při chybě se vypíše

+ + File error + +

obdobně, jako při chybě nahrávání programu.

Přechod do BASIC-G lze provést i pomocí kláves SHIFT+DEL, přechod do režimu inverzního zobrazení pomocí SHIFT a šipka vlevo s dorazem.

Příklady řešení úloh

=====

Lekce/cvičení:

- 11/1 60 DISP"Dalsi cast    Znovu    Tisk    Konec"  
62 INPUT A\$:B\$="DZTK":FOR F =1 TO 4  
64 IF A\$=MID\$(B\$,F,1) THEN ON F GOTO 71,72,73,79  
66 NEXT F
- 11/2 Klíče se SHIFT nedávají žádné číselné hodnoty, nelze je využívat pro programový přepínač.
- 12/1 Začátek programu je shodný (řádky 10,15), další pokračování  
20 PRINT Z;"=":ON Z GOSUB 21,22,23,24,25:GOTO 10  
21 PRINT"Vyborne":RETURN  
22 PRINT"Chvalitebne":RETURN  
23 PRINT"Dobre":RETURN  
24 PRINT"Vyhovujici":RETURN  
25 PRINT"Nevyhovujici":RETURN
- 12/2 2 GCLEAR:E=0:CR=0:ON ERR GOTO 6  
4 GOTO 100  
6 E=PEEK('26):CR=APEEK('5E5D)  
8 PRINT AT 24,0;"Chyba";E;" na radku:";CR  
10 IF E=8 OR E=12 OR E=15 OR E=17 OR E=19 OR E=23 THEN E+10  
12 ON ERR GOTO 6  
14 GOTO 100  
18 PRINT"Prekrocen číselný rozsah (10<sup>7</sup>-38,10<sup>7</sup>38)":GOTO 12  
22 PRINT"Pokus o dělení nulou":GOTO 12  
25 PRINT"Plná paměť pro texty-retezce":GOTO 12  
27 PRINT"Text místo čísla nebo naopak":GOTO 12  
29 PRINT"Chyba INPUT: písmeno místo čísla":GOTO 12  
33 PRINT"Chyba nahravky; očisti hlavu magnetofonu":GOTO 12  
100 DISP"Test ERR 8; zadej číslo >38":INPUT K:J=K^K  
102 DISP"Test ERR 12; zadej číslo 0":INPUT K:J=K/K  
104 DISP"Test ERR 17; zadej znak":INPUT K\$:J=K\$  
106 DISP"Test ERR 19; zadej znak":INPUT K:END
- 13/1 10 GCLEAR:SCALE=127,127,0,242:MOVE=10,0  
20 PLOT=10,20;-100,20;-10,40;-80,40;-8,60;-60,60  
30 PLOT=6,80;-40,80;-4,100;-20,100;0,120  
40 PLOT20,100;4,100;40,80;6,80,60,60;8,60  
50 PLOT80,40;10,40;100,20;10,20;10,0  
60 END
- 13/2 70 GCLEAR:SCALE=1.2,1.2,-1,1:PI=3.14159:MOVE 0,1  
72 FOR I=0 TO 2\*PI STEP .11:PLOT SIN(I),COS(I):NEXT I  
74 FOR I=0 TO PI STEP PI/30:MOVE SIN(I),COS(I)  
76 PLOT SIN(PI+I),COS(PI+I):NEXT I:END

```
14/1 10 GCLEAR:SCALE0,255,0,242
      15 MOVE 0,0:PLOT 241,0;242,242;0,241;0,0
      20 MOVE 1,1:FOR I=1 TO 4:FILL30,30;5:FILL30,30;10:NEXT I
      25 MOVE 121,1:FOR I=1 TO 4:FILL30,30;5:FILL30,30;10:NEXT I
      30 END

14/2 50 GCLEAR:MOVE 120,0:FILL 5,20;1
      52 I=0:FOR X=100 TO 120 STEP 2:MOVE X,20+I
      54 FILL 45-4*I,1;1:I=I+1:NEXT X
      56 I=0:FOR X=105 TO 121 STEP 2:MOVE X,31+I
      58 FILL 35-4*I,1;1:I=I+1:NEXT X
      60 I=0:FOR X=110 TO 122 STEP 2:MOVE X,40+I
      62 FILL 25-4*I,1;1:I=I+1:NEXT X
      64 I=0:FOR X=115 TO 121 STEP 2:MOVE X,47+I
      66 FILL 15-4*I,1;1:I=I+1:NEXT X
      68 I=0:FOR X=120 TO 122 STEP 2:MOVE X,51+I
      70 FILL 5-2*I,1;1:I=I+1:NEXT X:END

16/1 Postačuje upravit řádky.
      55 I=INT(RND(1)*N):PRINT CX(I);"=";
      70 DISP"Pokracovat A/N?":INPUT PX:IF PX="A" THEN 55
      72 PRINT,"Je to ";:IF SP>7...

16/2 Protože je slovník uchován jako data na kazetě, můžeme zná-
      má slovíčka vyškrtnout z paměti úpravou původního programu:

      60 INPUT SX:IFSX=AX(I)THEN SP=SP+1:AX(I)="" :CX(I)="" :GOTO70

17/1 10 GCLEAR:PRINT AT24,0"Zadej 1.cislo":INPUT AX
      11 PRINT AT25,0"Zadej 2.cislo":INPUT BX:P=0

17/2 20 GCLEAR:PRINT"Zadej vicemistne cislo.":INPUT KX
      22 B=0:C=1:FOR I=1 TO LEN (KX)
      24 V=VAL(MIDX(AX,I,1):B=B+V:C=C*V:NEXT I
      26 PRINT"Soucet vseh cislic je:";B
      28 PRINT"Soucin vseh cislic je:";C:END
```

Příloha ERROR  
=====

Chybová hlášení BASIC G:  
=====

Chyba (ERROR) Kód Význam

Subscr. err	1	Hodnota indexu pole je mimo hranice, nebo bylo použito shodné jméno pro různé typy proměnných nebo pokus o použití nedefinované uživ. funkce.
Arr. alloc.	2	Nesprávné nebo opakované dimenzování pole, dimenzování překročilo velikost volné paměti.
Err. param.	3	Chybný argument funkce.
Only in pg	4	Příkaz lze použít jen v programovém režimu.
No for stm	5	Chyba v cyklu FOR ... TO ... NEXT, chyba vnoření.
Data exhaust	6	Chybí DATA pro příkaz READ.
Pg. too big	7	Program je větší než vymezená paměť.
Overflow	8	Překročení rozsahu čísla: $10^{-38}$ - $10^{38}$ nebo $10^{38}$ .
Syntax err	9	Chybně zadaný příkaz (překlep).
Return err	10	Program narazil na RETURN bez volání GOSUB.
Numb. nonex	11	Neexistující číslo řádku pro GOTO/GOSUB/THEN.
Div by zero	12	Dělení nulou nebo chybné parametry SCALE.
Can't cont	13	Příkaz CONT použit nevhodně (po chybě/změně).
String long	14	Text v řetězci je delší než 255 znaků.
No str. spc	15	Přeplnění paměťové oblasti pro texty (řetězce).
Str. algrth	16	Řetězcový výraz je dlouhý nebo složitý, je nutné jej rozdělit.
Type conv	17	Text místo čísla nebo naopak.
File small	18	Neodpovídají deklarované parametry při DLOAD, liší se DIM v nahrávce a programu.
Input err	19	Neodpovídá hodnota v INPUT (znak místo čísla).
Field lost	20	Překročen počet údajů pro INPUT.
File bound	21	Číslo souboru na kazetě je mimo 0-99.
Stop	22	Zastavení programu příkazem/klávesou STOP.
File error	23	Chyba při nahrávání z magnetofonu.

Znaky v kódu ASCII (šestnáctkové a desítkové):

00	0	NUL	20	32	mezera	40	64	@	60	96	`
01	1	SOH	21	33	!	41	65	A	61	97	a
02	2	STX	22	34	"	42	66	B	62	98	b
03	3	ETX	23	35	#	43	67	C	63	99	c
04	4	EOT	24	36	x	44	68	D	64	100	d
05	5	ENQ	25	37	%	45	69	E	65	101	e
06	6	ACK	26	38	&	46	70	F	66	102	f
07	7	BEL	27	39	'	47	71	G	67	103	g
08	8	BS	28	40	(	48	72	H	68	104	h
09	9	HT	29	41	)	49	73	I	69	105	i
0A	10	LF	2A	42	*	4A	74	J	6A	106	j
0B	11	VT	2B	43	+	4B	75	K	6B	107	k
0C	12	FF	2C	44	,	4C	76	L	6C	108	l
0D	13	CR	2D	45	-	4D	77	M	6D	109	m
0E	14	SO	2E	46	.	4E	78	N	6E	110	n
0F	15	SI	2F	47	/	4F	79	O	6F	111	o
10	16	DLE	30	48	0	50	80	P	70	112	p
11	17	DC1	31	49	1	51	81	Q	71	113	q
12	18	DC2	32	50	2	52	82	R	72	114	r
13	19	DC3	33	51	3	53	83	S	73	115	s
14	20	DC4	34	52	4	54	84	T	74	116	t
15	21	NAK	35	53	5	55	85	U	75	117	u
16	22	SYN	36	54	6	56	86	V	76	118	v
17	23	ETB	37	55	7	57	87	W	77	119	w
18	24	CAN	38	56	8	58	88	X	78	120	x
19	25	EM	39	57	9	59	89	Y	79	121	y
1A	26	SUB	3A	58	:	5A	90	Z	7A	122	z
1B	27	ESC	3B	59	;	5B	91	[	7B	123	{
1C	28	FS	3C	60	<	5C	92	\	7C	124	
1D	29	GS	3D	61	=	5D	93	]	7D	125	}
1E	30	RS	3E	62	>	5E	94	^	7E	126	~
1F	31	US	3F	63	?	5F	95	_	7F	127	■

MHB8255 - paralelní vstup/výstupní brány PA, PB, PC:

Profil řídícího slova pro různé režimy práce:

76543210 - bity výstupního (řídícího) bytu posílaného do CWR  
 0000bitX - režim nastavení bitu PC (bit) do hodnoty X (0/1);  
 0000000X = '00-PC0=0; = '01-PC0=1  
 0000001X = '02-PC1=0; = '03-PC1=1  
 0000010X = '04-PC2=0; = '05-PC2=1      např. 3E05      MVI A,5  
 0000011X = '06-PC3=0; = '07-PC3=1      D34F      OUT 4FH  
 0000100X = '08-PC4=0; = '09-PC4=1  
 0000101X = '0A-PC5=0; = '0B-PC5=1      např. OUT 79,11  
 0000110X = '0C-PC6=0; = '0D-PC6=1      CONTROL 4,3,13  
 0000111X = '0E-PC7=0; = '0F-PC7=1

1 - aktivní druh provozu (mody 0,1,2)  
 1XX - 00-Skupina A mod 0 (nebo: 01-mod 1; 10-mod 2)  
 1XX - 0-PA OUT/výstup (1-PA IN/vstup); OUT 29,144  
 1XX - 0-PCH OUT (1 PCH IN); PCH=PC4/7)  
 1XX - 0-Skupina B mod 0 (1-skupina B mod 1); CONTROL 4,3,132  
 1XX - 0-PB OUT/výstup (1-PB IN/vstup)  
 1XX - 0-PCL OUT/výstup (1-PCL IN); PCL=PC0/3)

Mod 0 ve všech branách (zápis nebo čtení dat)

76543210	K	PA	PB	PCH	PCL
10000000	= '80=128	out	out	out	out
10000001	= '81=129	out	out	out	in
10000010	= '82=130	out	in	out	out
10000011	= '83=131	out	in	out	in
10001000	= '88=136	out	out	in	out
10001001	= '89=137	out	out	in	in
10001010	= '8A=138	out	in	in	out
10001011	= '8B=139	out	in	in	in
10010000	= '90=144	in	out	out	out
10010001	= '91=145	in	out	out	in
10010010	= '92=146	in	in	out	out
10010011	= '93=147	in	in	out	in
10011000	= '98=152	in	out	in	out
10011001	= '99=153	in	out	in	in
10011010	= '9A=154	in	in	in	out
10011011	= '9B=155	in	in	in	in

Mod 1 (HANDSHAKE) pouze v PA (+PC3/7) a PB (+PC0/2):

76543210	K	PA	PB	PCi (např. 7=PC7...=-DBF)
10100000	= 'A0=160	out	...	PC7=-DBF, PC6=-ACK, PC3=INTRa
10110000	= 'B0=176	in	...	PC4=-STB, PC5=IBF, PC3=INTRa
10000100	= '84=132	...	out	PC1=-DBF, PC2=-ACK, PC0=INTRb
10000110	= '86=134	...	in	PC2=-STB, PC1=IBF, PC0=INTRb

a jejich kombinace, nebo kombinace s modem 0.

Mod 2 pouze v PA (+PC3/7); PB může mít mody 0 nebo 1:

76543210	K	PA	PB	(PC3/7 viz u PA mod 1)
11000000	= 'C0=192	i/o	out(mod 0)	OUTPUT 405;výraz
1100001X	= 'C2=194	i/o	in (mod 0)	ENTER 405;proměnná
1100010X	= 'C4=196	i/o	out(mod 1)	CONTROL 4,3,196
1100011X	= 'C6=198	i/o	in (mod 1)	OUT 79,198



MHB 8251A - seriový synchronní/asynchronní vysílač/přijímač:

Formát instrukce pro synchronní provoz:

76543210 = bity řídícího slova CWR (Control Word Register);  
 .....00 - synchronní provoz;  
 ....xx00 - počet bitů: xx=00-5bitů; 01-6bitů; 10-7bitů; 11-8bitů;  
 ...x...00 - PEN (Parity ENable)-kontrola parity: x=0-ne; x=1-ano;  
 ..x....00 - EP-druh parity: x=0-lichá; x=1-sudá;  
 .x.....00 - synchronizace: x=0-interní; x=1-externí;  
 x.....00 - počet synchronizačních znaků: x=0-2sz; x=1-1sz;  
 00001100 = '0C=12: CONTROL 1,1;12 -provoz:sync+8b-PEN-EP+int+2sz  
 zápis synchroznaků: CONTROL 1,1;170,85 -2sz: 170='AA, 85='55.  
 Dále musí následovat povel zahájení přenosu, tj.celkem 4 byty.

Formát instrukce pro asynchronní provoz:

76543210 =bity řídícího slova CWR;  
 .....xx -rychlost (baud): xx=01-1\*; 10-16\*; 11-64\*;  
 ....xx.. -počet bitů: xx=00-5bitů; 01-6bitů; 10-7bitů; 11-8bitů;  
 ...x.... -PEN-kontrola parity: x=0-ne; x=1-ano;  
 ..x..... -EP-druh parity: x=0-lichá; x=1-sudá;  
 xx..... -počet stop-bitů: xx=01-1sb; xx=10-1.5sb; xx=11-2sb;  
 11001110 = 'CE=206: CONTROL 1,1;206 -provoz:as+16\*+8b-PEN-EP+2sb;  
 a musí následovat povel zahájení přenosu.

Formát povelu pro zahájení nebo změnu přenosu:

76543210 =bity řídícího povelu vyslaného do CWR:  
 .....1 -TxEN-povoleno vysílání; (0-vysílání nepovoleno);  
 ....1. -DTR-Data Terminal Ready: 1-zapnout modem; (0-vypnout);  
 .....1.. -RxEN-povolen příjem dat; (0-nepovolen);  
 ....1... -SBREAK-přerušeni provozu; (0-normální provoz);  
 ...1.... -ER-(Error Reset) -vynulování příznaků chyb:PE,OE,FE  
 ..1..... -RTS-(Request To Send) -výzva k vysílání (pro modem);  
 .1..... -IR-(Internal Reset) -vnitřní nulování;  
 1..... -EH-(Enable Hunt)-povoleno vyhledávání synchron. znaků;  
 00110111 = '37=55: CONTROL 1,1;55 -povel zahájení přenosu.

\*Stavové slovo MHB8251A: (S=STATUS1,1: S=INP(31))

76543210 -status byty: (B=BIT S,I)  
 .....x -TxRDY: x=1-nemá data k vysílání; x=0-vysílá data;  
 ....x. -RxRDY: x=1-přijímač přijal data, x=0-čeká data;  
 ....x.. -TxE: x=1-vysílání ukončeno; x=0-neukončeno;  
 ....x... -PE-Parity Error: x=1-chyba parity; x=0-parita správná;  
 ...x.... -FE-Frame Error: chyba rámce: x=1-nepřišel STOP bit;  
 ..x..... -OE-Overrun Error: x=1-nepřevzata včas přijatá data;  
 .x..... -SYNDET-SYNchronizace DETekována -přijaty synchroznaky;  
 x..... --DSR: x=0-modem připraven (odpověď na signál DTR);  
 00001010 -S='0A=10 - byte dat přijat s chybou parity.

CTC 8253-16 bitové čítače/časovače: CT0,CT1,CT2.

Formát řídícího slova CW (Control Word) pro CWR: CONTROL 5,3;CW 76543210 - bity řídícího slova (bytu):

00..... - CT0-čítač 0; R=Read-čtení  
 01..... - CT1-čítač 1; W=Write-zápis  
 10..... - CT2-čítač 2; LB=Low Byte (nižší byte)  
 11..... - Neplatné; HB=High Byte (vyšší byte)  
 ..00.... - Čtení se vzorkováním;  
 ..01.... - R/W HB - čtení/zápis vyššího bytu  
 ..10.... - R/W LB - čtení/zápis nižšího bytu  
 ..11.... - R/W LB,HB - čtení/zápis obou bytů  
 ....000. - Mod 0: čítá dolů, nahodí OUT, čítá dál;  
 ....001. - Mod 1: čítá dolů, nahodí OUT; start po GATE=1  
 ....010. - Mod 2: dělička CLK/N, výstup: impuls -OUT  
 ....011. - Mod 3: dělička CLK/N, výstup: +OUT=N/2, -OUT=N/2  
 ....100. - Mod 4: čítá dolů, výstup: -OUT; start po zápisu dat  
 ....101. - Mod 5: čítá dolů, výstup: -OUT; start po GATE=1  
 .....X - druh čítání: 0-binární; 1-desítkové;  
 00110000 - '30=48: CONTROL5,3;48 OUT95,48

Tabulka kódů řídících slov čítačů pro mody:

CT\ mod:	0	1	2	3	4	5
0: vzorkuj	0	0	0	0	0	0
R/W LB	16	18	20	22	24	26
R/W HB	32	34	36	38	40	42
R/W LBHB	48	50	52	54	56	58
1: vzorkuj	64	64	64	64	64	64
R/W LB	80	82	84	86	88	90
R/W HB	96	98	100	102	104	106
R/W LBHB	112	114	116	118	120	122
2: vzorkuj	128	128	128	128	128	128
R/W LB	144	146	148	150	152	154
R/W HB	160	162	164	166	168	170
R/W LBHB	176	178	180	182	184	186

Konstanty nastavení čítačů při kmitočtu hodin CLK=2.048 MHz  
 Rychlost Dělicí poměr Šestnáctkové Desítkové:CONTROL 5,X;  
 Baud=bit/s HB LB LB,HB

150	13653	3555	85,53
300	6826	1AAA	170,26
600	3413	0D55	85,13
1200	1706	06AA	170,6
2401	853	0355	85,3
4807	426	01AA	170,1
9615	213	00D5	213,0
19320	106	006A	106,0
38641	53	0035	53,0
78769	26	001A	26,0
157538	13	000D	13,0
292571	7	0007	7,0

Literatura:  
=====

- Peter Gabčo: BASIC - slovník jazyka  
Učebné pomůcky Banská Bystrica, 1987
- Miloslav Feil: BASIC pro PMD 85-2  
Komenium Praha, 1988
- Michal Vejvoda: Programovací jazyk BASIC pro PMD 85  
Knihnice Svazarmu, Praha 1988
- Ivo Machačka, Jan Pavlí: Programování v jazyku BASIC  
SNTL Praha, 1985
- Emil Kollert: Programování počítače IQ151 v jazyku BASIC  
Komenium Praha, 1984
- Petr Kroha, Pavel Slavík: BASIC pro začátečníky  
SNTL Praha, 1988
- Zdeněk Jedlička: BASIC pro začátečníky-příručka k počítači IQ151  
Komenium Praha, 1986
- Josef Olehla, Miroslav Olehla: BASIC u mikropočítačů  
NADAS Praha, 1988
- Kolektiv: Úvod do programování (kurs číslicové techniky II)  
Knihnice ČSVTS, Praha 1986, dodává Tesla ELTOS
- Pavel Valášek: Mikroprocesor 8080 a základní obvody  
Knihnice ČSVTS, Praha 1986, dodává Tesla ELTOS
- Roman Kišš: Osobní mikropočítač PMD-85 -uživatelské příručky 1-5  
Dům techniky ČSVTS Ostrava 1985
- Vít Libovický, Jiří Olmer: Komentovaný výpis monitoru mikropočí-  
tače PMD 85-2, Zenitcentrum Beroun, Hostimská 703
- Aktuality C 2717, Incotex Brno, Hybešova 42, 65664 Brno

# Obsah:

=====

	Strana:
Lekce11: Dialog mezi programem a uživatelem BEEP, PAUSE, AND, OR, NOT, INKEY, ON, ... GOTO	3
Lekce12: Podprogramy a jejich využívání GOSUB, RETURN, ON, ... GOSUB, ON ERR GOTO	7
Lekce13: Práce v grafickém režimu počítače SCALE, AXES, MOVE, PLOT	10
Lekce14: Zobrazování v grafickém režimu PEN, LABEL, FILL	13
Lekce15: Jak vyzrát na háčky a čárky BMOVE, BPLOT, CHR*	16
Lekce16: Datová pole a práce s nimi DIM, DSAVE, DLOAD	19
Lekce17: Uživatelské funkce a jejich využití DEF FNC, FNC, VAL, PRINT INK	23
Lekce18: Co jsme dosud neprobrali ADR, PEEK, POKE, HEX*, FRE, APEEK, APOKE	26
Lekce19: Vstupní a výstupní obvody počítače CONTROL, STATUS, OUTPUT, OUT, ENTER, INP, BIT, WAIT	29
Lekce20: Používání příkazů MONITORU počítače DUMP, MEM, SUB, JUMP, MGLD, MGSV, MGEND	33
Příloha 1: Příklady řešení úloh	36
Příloha ERROR: Chybová hlášení BASIC-G	38
Příloha ASCII: Znaky v kódu ASCII –šestnáctkové a desítkové	39
Příloha 8255: Programování paralelních v/v obvodů	40
Příloha 8251: Programování seriového v/v obvodu	41
Příloha 8253: Programování čítače/časovače	42
Literatura:	43

---

Název: CONSUL 2717: Práce s počítačem - druhý kurs  
 Sestavil: Ing. Pavel Hlaváček  
 Vydal: Incotex, státní podnik,  
 Hybešova 42, 656 64 Brno

Cena: 8,- Kčs dohodnutá podle výměru FČJ číslo V-6/88, pol. 144