ÖNVÉDELEM

Látva az inkvizítorok lesújtó véleményét az AIRCOMP BASIC-ről, úgy gondoltam, fel kell emelnem szavamat mellette. Bár mint fejlesztő, talán nem tudok kellőképpen elfogulatlan lenni, mégis az a véleményem, ez az interpreter egy nagyon jól használható darab. Azok az extra újdonságok, amelyekkel a BASIC rendelkezik, mindenképp ezt támasztják alá. Sajnos a kínvallatás során kiderült, hogy még a rendszeres felhasználók sem eléggé ismerik ezeket. Remélem, ezen a problémán segít majd a gyártók új dokumentációja. Felhasználva az újság adta nyilvánosságot, hadd járuljak hozzá én is egy-két ötlettel a gép hatékonyabb felhasználásához.

Két speciális karakter.

A PRINT CHR $ (5) a CR változó „színére" törli a grafikus képet (ugyanez a CALL 7535-tel is elérhető).

A PRINT CHR $ (12) letörli az alfás képernyőt (ugyanez SHIFT/CR-rel is elérhető, illetve az is beépíthető egy stringbe).

Kiszámított GOTO, GOSUB, CALL

Az utasítások után aritmikai kifejezés is állhat, de ezek nem kezdődhetnek számmal. A számmal kezdődő kifejezést csak sorszámnak veszi, pl. GOTO A*10 és nem GOTO 10*A.

PEEK, POKE

A 8000 alatti címek nem memóriához fordulnak, hanem I/O címeket szólítanak meg. Így I/O eszközök olvashatók, illetve írhatók.

PRINT AT helyett

A CURSORPOINTER HEX 4014–4015 címen van. Ahova ez mutat, ott villog majd a cursor, illetve oda ír a gép. A következő programrészlet (praktikusan szubrutin) az Y sor X pozíciójába állítja a cursort: A = X+Y * 40+1:B = INT (A/256): POKE 16404, A – B * 256, B+192.

Gépi kódú tárolás – BASIC-ből

A következő gépi kódú program egy BASIC-CALL-lal meghívva kitárolja a megadott területet:

```
LD HL # Start
LD BC # End
LD DE (név eleje)
CALL 5E8
RET
```

(Tehát HL-ben a kezdet, BC-ben a vég, DE pedig a név első byte-jára mutat. A név végén 22 vagy 60 (HEX) karakter áll. Ha név nem kell, DE pl. egy 60-ra mutasson.)

Gépi kódú program elhelyezése a BASIC-ben

Legegyszerűbb, ha a programot így kezdjük:

```
1 GOTO 10
1. ".............."
10 ide jön a program.
```

A második sorba egy csupa spaceből álló stringet írunk be, olyan hosszút, amilyen hosszú a gépi program lesz. Monitorba átmenve 40A6-tól megtaláljuk a második sort, és a space (HEX 20) helyére beírhatjuk a gépi programot. Sajnos a program így nem tartalmazhat 22 vagy 60 (HEX) byte-ot, és BASIC-be visszatérve a 2. sor nem javítható – nem is szabad rámenni a cursorral.

A változó tábla kezdetére a 4018–19, a végére pedig a 401C–1D pointerek mutatnak. Ha ezt a területet kitároljuk (pl. a korábban említett módon), akkor ezeket az adatokat ugyanez a program beolvashatja. Ez így lényegesen gyorsabb, mint a PRINT# és az INPUT#.

A programok kicsit gyorsabbak lesznek, ha:

- A sűrűn használt változókat a program elején definiáljuk (DIM-mel vagy pl. A = A-val);
- Ne használjunk számkonstansokat – ezeket valahol tegyük bele változókba;
- A GOTO és GOSUB lehetőleg a program elején levő sorokat hívja.

Lukács József

Már régen készültünk a fenti bugyuta szóvicc elsütésére, de sohasem kínálkozott rá lehetőség. Így hát elhatároztuk, hogy az áprilisi BIT-LET-ben megjelenő CSM LOGO kiegészítésünket telis-tele rakjuk hibával, hogy így a májusi számban végre elsüthessük a LÓGÓ LOGO-t. Elhatározásunkat tett követte. Sikerrel. BIT-LET-ünk fennállása óta együttvéve nem sikerült annyi hibát elkövetni, mint abban a listában. Így hát nem valószínű, hogy a program valakinek is működött volna. Lógunk tehát némi hibaigazítással. E helyütt kérünk elnézést az olvasóktól, s reméljük, az alábbi listákban már semmi huba sincs, s ezeket beírva az áprilisi számban közöltek helyett és mellett – mert sajnos nem csak hibákat közöltünk, de ki is felejtettünk három programrészletet – most már talán nem lóg a LOGO, hanem rajzol!

```
-  330 IF LEN P$ < I+1 THEN RETURN
-  340 IF / P$/I+1/ <=":" AND P$/I+1/ > "/" / THEN stb.
-  342 IF PAR > 1 THEN stb.
-  360 IF PAR > 3 THEN stb.
-  380 IF / P$/I+1/ <":" AND P$/I+1/ > "/" / THEN stb.
-  385 IF C$ <> "" THEN stb.

-  706 IF W/RSP/ > 0 THEN stb.

-  810 IF P$/I+1/<> ":" THEN RETURN
-  814 IF P$/I/<> " " THEN stb.
-  820 LET B$=B$+"/7 szóköz!/": stb.

- A 1110. sorban üres string /""/ szerepel.
- 1133 IF SORSZ > 20 stb.
- 1138 IF B$/TO 3/<> "ELJ" AND B$/TO 2/<>"VE" stb.
- 1140 PRINT AT SORSZ,TAB; stb.
- 1141 IF B$/1/=" " THEN stb. / 1 szóköz!/
- 1144 IF A$/TO 3/<> ELJ THEN LET A$=A$ + "VEGE" : GO SUB
  PARANCS : LET A$="": GO TO 1100 :REM Üres string!

- 1750 IF A$/A+1/<> ":" THEN stb.
- A 1910. sor 3. utasításában 7 szóközt adunk a B$-hoz!
- 1920 IF VNCIM+6 > 29 THEN stb.
- A 1930. sor utolsó utasítása: LET A=VVEGE

- 2210 IF P$/I/<> " " THEN stb. / 1 szóköz!/
- A 2215.sor 3.utasításában 7 szóközt adunk a D$-hoz.

9090 DIM E$(20,29): REM  20* ELJARAS NEVE (MAX.7 BETUSEK!) +
 VALTOZOINAK SZAMA + HAROM VALTOZO NEVE
9092 FOR A=1 TO 20: LET E$(A,8)="0": NEXT A
9100 DIM E(20,4): REM A 20 ELJARAS CIME A P$-BAN + EGYENKENT
 3 VALTOZO ERTEKE
9110 LET ELJSZAMA=1: REM AZ 1.ELJARAS KOVETKEZIK
9120 DIM P(6): REM PARAMETEREK

-- 9305 LET VALTOZO = 800
-- 9310 LET ELJARAS = 1700 :REM Igy marad!
```

A „CÍMEK" és a „PROGRAM" nevű LOGO utasítások listáját, mely helyhiány miatt kimaradt, az alábbiakban közöljük:

```
2000 REM PROGRAM
2010 CLS
2015 LET B=1
2020 FOR A=1 TO LEN P$
2030 PRINT P$(A);
2040 IF B=30 THEN PRINT : PRINT "12345678901234567890123456
890": PRINT : LET B=0
2050 LET B=B+1
2060 NEXT A
2070 PRINT : PRINT "1234567890123456789012345678901":RETURN
```

```
2100 REM ELJ.CIMLISTAJA (CIMEK)
2110 CLS : PRINT "ELJ.CIME","NEVE": PRINT
2120 FOR A=1 TO 20
2130 PRINT E(A,1);" ";E$(A)
2135 PRINT E(A,2);" ";E(A,3);" ";E(A,4)
2140 NEXT A
2150 RETURN
```