Getting the PRINT USING keyword that's missing in SuperBASIC

5-Feb-89

F_FORM by Timo Salmi

I am currently (written in 1987) working on a linear programming and a linear goal programming code (linprep_exe and linsolve_exe), which I have developed and tested quite thoroughly over several years for the VAX/VMS. As you probably know linear programming codes are usually available for mainframe computers, but very seldom for micros. This results from the fact that LP is a rather spacialized application and not quite trivial to program without certain pitfalls.

An example of a tiny LP-problem:

Max z = 2X1 + 3X2
subject to
X1 + 4X2 < 12
3X1 - X2 < 8
X1,X2>0

Naturally I have had to develop a host of Super-BASIC procedures and functions. One function in particular should be of general interest. As we know, often to our chagrin, Super_BASIC lacks the PRINT USING statement. My function f_form$ makes up for this. (E.g. Donald Alcock has presendted a using$ function in his book, but it fails under several special circumstances). The syntax of f_form$ is adapted from the FORTRAN format statement Ff.d option. Thus e.g.

a = -3.456
a$ = f_form$(a,7,2) will return
-3.46

with the proper leading spaces. Large and small figures such as 9.9E10 and -1.6E-3 do not produce errors, but are handled by the function. Naturally, bugs can remain, and if anybody finds any, I would appreciate the information. In f_form$(a,7,0) the point is dropped following BASIC rather than FORTRAN conventions. Overflow is signalled by string ####.## (in the case of f_form$(10000000,7,2). The procedure is especially needed if using SUPERCHARGE, since otherwise supercharged programs output e.g. 0.1 as 9.99999999E-2. (The procedure in the supercharge manual to format output is not general enough.) I have saved this function on the cartridge, since it might be worthwhile for inclusion in the library.

Professor Timo Salmi

Home: Office:
Soukankaari 2 B 36 University of Vaasa
02360 Espoo P.O.Box 297
Finland SF-65101 Vaasa
Finland

InterNet addess: ts@chyde.uwasa.fi

```
26000 DEFine FuNction f_form$(luku,ff%,dd%)
26010 LOCal lukul$(mp),yl$(mp),kl%,el%,pl%,pitl%,wl%(2)
26020 kl%=ff%-dd%-1
26030 IF luku>=0 THEN
26040 lukul$=luku:negl%=0
26050 ELSE :lukul$=-luku:negl%=1:END IF
26060 IF "."INSTR lukul$ OR "e"INSTR lukul$ THEN lukul$=lukul$+.5*10^-dd%
26070 IF lukul$(1)="." THEN lukul$="0"&lukul$
26080 pitl%=LEN(lukul$):el%="e"INSTR lukul$:pl%="." INSTR lukul$
26090 IF NOT el% THEN
26100 yl$=lukul$
26110 IF NOT pl% THEN
26120 IF dd%>0 THEN yl$=yl$&"."&FILL$("0",dd%)
26130 ELSE
26140 yl$=lukul$(1 TO pl%-1)
26150 IF dd%>0 THEN
26160 IF pitl%-pl%>=dd% THEN
26170 yl$=yl$&"."&lukul$(pl%+1 TO pl%+dd%)
26180 ELSE :yl$=yl$&"."&lukul$(pl%+1 TO pitl%)&FILL$("0",dd%-(pitl%-pl%)):END IF
:END IF :END IF
26190 IF negl%:yl$="-"&yl$:END IF :wl%(1)=LEN(yl$)
26200 IF wl%(1)<=ff% THEN
26210 RETurn FILL$(" ",ff%-wl%(1))&yl$
26220 ELSE
26230 IF dd%>0 THEN
26240 RETurn FILL$("#",kl%)&"."&FILL$("#",dd%)
26250 ELSE :RETurn FILL$("#",ff%):END IF :END IF
26260 END IF :wl%(1)=lukul$(el%+1 TO)
26270 IF wl%(1)>=0 THEN
26280 IF pl% THEN
26290 yl$=lukul$(1TO pl%-1)&lukul$(pl%+1TO el%-1)
26300 ELSE :yl$=lukul$(1TO el%-1):END IF
26310 wl%(2)=LEN(yl$)
26320 IF wl%(2)<wl%(1)+1 THEN yl$=yl$&FILL$("0",wl%(1)-wl%(2)+1)
26330 ELSE
26340   IF pl% THEN
26350     yl$="."&FILL$("0",-wl%(1)-1)&lukul$(1TO pl%-1)&lukul$(pl%+1 TO el%-1)
26360   ELSE :yl$="."&FILL$("0",-wl%(1)-1)&lukul$(1TO el%-1):END IF
26370   END IF :lukul$=yl$:GO TO 26070
26380 END DEFine f_form$
26390 :
```

Resave and Date-mark Your Programs in SuperBASIC

25th February, 1987
Mr. Leon Heller
30 Baldslow Road

Hastings
East Sussex TN34 2EY
England

Dear Leon,

For publication in QUANTA:

Here is a handy little DSAVE procedure for resaving and datemarking SuperBASIC programs. Merge it into your program, substitute the name of your program on 32020 between quotes, and substitute your own name on line 32120. Do not use line numbers from 1 to 3 in your original program, since the lines are reserved for datemarking. Do not renumber the lines from 1 to 3 at any stage to avoid confusion. The procedure checks whether the current year on the clock is 1987. If not, DSAVE is not performed. In due time you have to update the year on line 32040 for obvious reasons. To resave and datemark on e.g. MDV1_ just give command DSAVE 1
If you have diskdrives other than FKDn_, or ramdisk, substitute the devicenames on lines 32090 to 32100 to suit your own peripherals.

```
1 REMark dsave (c) My Name
2 REMark Wed 1987 Feb 25 09:35:04
3 :
32000 DEFine PROCedure DSAVE(dr)
32010   REMark DSAVE by Timo Salmi
32020   LOCal a$,b$:a$="dsave"
32030   b$=DATE$:b$=b$(1 TO 4)
32040   IF b$<1987 OR b$>1987 THEN
32050     PRINT#0,DAY$!DATE$;", DSAVE abort, set date"
32060     BEEP 4000,12:STOP:END IF
32070   IF dr=1:b$="mdv1_"&a$
32080   IF dr=2:b$="mdv2_"&a$
32090   IF dr=3:b$="fdk1_"&a$
32100   IF dr=4:b$="fdk2_"&a$
32110   DELETE b$:OPEN_NEW#3,b$
32120   PRINT#3,"1 REMark ";a$!"(c) My Name"
32130   PRINT#3,"2 REMark ";DAY$!DATE$
32140   PRINT#3,"3 :"
32150   LIST#3,100 TO:CLOSE#3:PRINT#0,"DSAVEd"!b$
32160 END DEFine DSAVE
```

Prof. Timo Salmi
School of Business Studies
University of Vaasa
Raastuvankatu 31
SF-65100 Vaasa, Finland

Resetting QL windows in SuperBASIC

BACK TO BASICS
Below you have yet another addition to the abounding set of procedures resetting the windows. This one is especially intended to be useful in connection with writing SuperBASIC programs.

The alternatives are RESET 8, RESET 4, and RESET 0. All the three are variants of the television display, where windows #1 and #2 are on top of each other. RESET 8 gives the same standard TV display as you get from key F2 when switching the QL on. The other two are similar but with smaller character sizes.

```
31000 DEFine PROCedure RESET(i)
31010   LOCal il:MODE i:REMark by Timo Salmi
31020   WINDOW 513,256,0,0:PAPER 0:CLS
31030   OPEN#2,con_:WINDOW#2,448,200,32,16
31040   PAPER#2,1:INK#2,7
31050   WINDOW#0,448,40-(i=4),32,216+(i=4)
31060   WINDOW 448+8*(i=4),200+2*(i=4),32-4*(i=4),16-(i=4)
31070   PAPER 2*(i=8):BORDER (i=4),4*(i=4)
31080   FOR il=0,1,2:CSIZE#il,(i=4)+2*(i=8),0:END FOR il
31090   PAPER#0,0:INK#0,7-3*(i=0):INK 7:CLS
31100   SCALE 100,0,0:FILL 0
31110 END DEFine RESET
```

Useful POKEs in SuperBASIC

PEEKAPOKE

Although QL uses relative addressing peeks and pokes can be useful. Here a a selection of memory addresses collected by Timo Salmi. Some of the addresses may have different values depending on the QL ROM version and the peripherals attached. The screen starts from 131072. One way of saving the screen is writing SBYTES mdv1_screen,2^17,2^15

The free memory can be obtained e.g. from function

```
18000 DEFine PROCedure f_mem
18010   RETurn PEEK_L(163856)-PEEK_L(163852)
18020 END DEFine f_mem
```

Quite a number of similar peeks is provided on the files accompanying Digital Precision's TURBO compiler.

PEEK_L(163872)/1024-256 returns memory expansion in kilobytes. (Its value will be 0, 256 or 512).

POKE_W 163886,0

other commands ...

time=PEEK_W(163886):PRINT time/50

can under some circumstances be used to measure elapsed time. However, if the commands in between include e.g. INPUT the counter will be muddled.

Caps lock can be turned on by POKE 163976,255 and of by POKE 163976,0
POKE_W 163980,30 defines the delay before a key starts repeating 30 being the default.
POKE_W 163982,2 defines the rate at which the key is repeated 2 being the default.
POKE_W 163986,3 sets the multitasking toggle key in ACSII. The default 3 is CTRL C. E.g. after POKE_W 163986,9 the task which the input buffer is attached to is changed by pressing TABULATE instead of the familiar CTRL C.

A Multitasking Trace for SuperBASIC

TRACE IT

If you have Digital Precision's TURBO you can set up a trace for SuperBASIC. Compile the code first and then multitask it with EXEC.

```
1 REMark trace by Timo Salmi
2 REMark Wed 1987 Feb 25 19:45:25
3 :
100 IF COMPILED THEN SET_PRIORITY 8
110 OPEN#3,scr_:CSIZE#3,2,0:WINDOW#3,136,12,344,16
120 BORDER#3,1,2:INK#3,0:PAPER#3,7:time=DATE
130 REPeat loop
140   IF DATE-time>3 THEN
150     WINDOW#3,2,1,0,0:CSIZE#3,2,0
160     WINDOW#3,136,12,344,16:BORDER#3,1,2
170     time=DATE
180   END IF
190   bline%=BASIC_W%(104):AT#3,0,0
200   PRINT#3,"TRACE"!bline%;
210   p%=5-LEN(bline%):IF p% THEN PRINT#3,FILL$(" ",p%);
220 END REPeat loop
230 :
```

Conversion Between Number Bases in SuperBASIC

TOUCHING ALL BASES

Conversions between number bases are very easy to do with the following SuperBASIC functions. In order to convert a binary number to an ordinary decimal you would use

```
PRINT f_todec('1011',2)
```

For converting a decimal value to hexadicimal could be done with

```
PRINT f_fromdec$(131072,16)
```

Finally converting e.g. 34 from base 5 to octal would just need

```
PRINT f_fromdec$(f_todec('34',5),8)
```

```
18000 DEFine FuNction f_todec(number$,base%)
18010   LOCal i%,dl,kl,digit$(31),loop
18020   IF base%<2 OR base%>32 THEN RETurn -1
18030   digit$="123456789ABCDEFGHIJKLMNOPQRSTUV"
18040   kl=1:dl=0:i%=LEN(number$)+1
18050   REPeat loop
18060     i%=i%-1:IF i%<1 THEN EXIT loop
18070     j%=number$(i%) INSTR digit$
18080     IF j%>base%-1 THEN RETurn -1
18090     dl=dl+j%*kl:kl=base%*kl
18100   END REPeat loop
18110   RETurn dl
18120 END DEFine f_todec
18130 :
18140 DEFine FuNction f_mod(a,b)
18150   RETurn a-b*INT(a/b)
18160 END DEFine f_mod
18170 :
18180 DEFine FuNction f_div(a,b)
18190   RETurn INT(a/b)
18200 END DEFine f_div
18210 :
18220 DEFine FuNction f_fromdec$(number,base%)
18230   LOCal numberl,result$(36),digit$(32),loop
18240   IF base%<2 OR base%>32 OR number<0 THEN RETurn -1
18250   numberl=number:result$=""
18260   digit$="0123456789ABCDEFGHIJKLMNOPQRSTUV"
18270   REPeat loop
18280     result$=digit$(f_mod(numberl,base%)+1)&result$
18290     numberl=f_div(numberl,base%)
18300     IF numberl<=0 THEN RETurn result$
18310   END REPeat loop
18320 END DEFine f_fromdec$
18330 :
```

Prof. Timo Salmi

School of Business Studies
University of Vaasa
Raastuvankatu 31
SF-65100 Vaasa
Finland

Testing of File Existence, elapsed time, etc.

INSTRUCTIVE, ISN'T IT

Let us not forget the budding SuperBASIC programmers. Some honed FuNctions and PROCedures might be in order among the interesting, but abounding material related to the commercial QL software and hardware so much in evidence in Quanta nowadays.

I have nothing against this trend, since personally I mostly use my the QL as a serious tool. Nonetheless, we perhaps should have more material in Quanta reflecting the fact that QL also is a home computer (with features not available in PC's). Perhaps something in the vein of Sinclair QL World's Better Basic series. So, here we go.

The familiar INSTR operator finds the first location of a character (or a sub-string) in a string. The search is case-independent. In some applications case-dependent search is needed. Furthermore, the in the case the special characters, such as the scandinavian letters, INSTR does not function consistency. The result depends on whether the SuperBASIC program is interpreted or compiled, and there also may be differences between the different ROM versions, for all I know. So here is a case-independent function for a single-character search giving consistent results for the entire character set.

While "B" INSTR "abcdeABCDE" would return 2, ql_instr("B","abcdeABCDE") will return 7.

```
18600 DEFine FuNction ql_instr(d$,e$)
18610   REMark case-dependent INSTR by Timo Salmi
18620   LOCal i%,p%,loop
18630   i%=0:p%=LEN(e$)
18640   REPeat loop
18650     i%=i%+1:IF i%>p% THEN RETurn 0
18660     IF e$(i%)=d$ THEN RETurn i%
18670   END REPeat loop
18680 END DEFine ql_instr
18690 :
```

Notice the intentional use of the REPeat loop and integer variables. In compiled programs, loops built with integers and REPeat are considerably faster than the more familiar FOR loops. Also notice that the LENgth of the e$ string is evaluated outside the loop. This speeds up the function significantly.

TO BE OR NOT TO BE

One of the ever-recurring tasks in writing SuperBASIC programs is finding out whether a file

exists. Here, once again, is a function returning 1 (true) if the file exists and 0 (false) if not.

```
27500 DEFine FuNction ql_exist(f$)
27510   REMark existence of a file by Timo Salmi
27520   LOCal hl$(16),al$(36),fbl$(36),search,exists
27530   IF LEN(f$)<5 THEN RETurn 0
27540   hl$=f$(1 TO 5)&"tempdir_tmp"
27550   DELETE hl$:OPEN_NEW#6,hl$:DIR#6,f$(1 TO 5)
27560   CLOSE#6:OPEN_IN#5,hl$:INPUT#5,al$,al$
27570   fbl$="":IF LEN(f$)>5 THEN fbl$=f$(6 TO LEN(f$))
27580   REPeat search
27590     IF EOF(#5):exists=0:EXIT search:END IF
27600     INPUT#5,al$
27610     IF fbl$==al$:exists=1:EXIT search:END IF
27620   END REPeat search
27630   CLOSE#5:DELETE hl$:RETurn exists
27640 END DEFine ql_exist
27650 :
```

Notice the habit of dimensioning all strings, which is good programming practice, especially if the program will be compiled.

TIME FLIES

The time elapsed in using a program can be found in HH:MM:SS format by applying the following function.

```
1 start_time=DATE
.
```
the program (e.g. Quill-boot)
```
.
9998 INK#0,7:ql_elapsed(0):REMark output to channel #0
9999 :
30000 DEFine PROCedure ql_elapsed(ch%)
30010   REMark elapsed time by Timo Salmi
30020   LOCal tl,ml,sl
30030   PRINT#ch%,"ELAPSED"!;
30040   sl=DATE-start_time
30050   tl=INT(sl/3600):sl=sl-tl*3600
30060   ml=INT(sl/60):sl=sl-60*ml
30070   PRINT#ch%,tl DIV 10;tl MOD 10;":";
30080   PRINT#ch%,ml DIV 10;ml MOD 10;":";
30090   PRINT#ch%,sl DIV 10;sl MOD 10
30100 END DEFine ql_elapsed
30110 :
```

TOUCHING ALL BASES

Conversions between number bases are very easy to do with the following SuperBASIC functions. In order to convert a binary number to an ordinary decimal you would use

```
PRINT f_todec('1011',2)
```

For converting a decimal value to hexadecimal could be done with

```
PRINT f_fromdec$(131072,16)
```

Finally converting e.g. 34 from base 5 to octal would just need

```
PRINT f_fromdec$(f_todec('34',5),8)
```

```
18000 DEFine FuNction f_todec(number$,base%)
18010   LOCal i%,dl,kl,digit$(31),loop
18020   IF base%<2 OR base%>32 THEN RETurn -1
18030   digit$="123456789ABCDEFGHIJKLMNOPQRSTUV"
18040   kl=1:dl=0:i%=LEN(number$)+1
18050   REPeat loop
18060     i%=i%-1:IF i%<1 THEN EXIT loop
18070     j%=number$(i%) INSTR digit$
18080     IF j%>base%-1 THEN RETurn -1
18090     dl=dl+j%*kl:kl=base%*kl
18100   END REPeat loop
18110   RETurn dl
18120 END DEFine f_todec
18130 :
18140 DEFine FuNction f_mod(a,b)
18150   RETurn a-b*INT(a/b)
18160 END DEFine f_mod
18170 :
18180 DEFine FuNction f_div(a,b)
18190   RETurn INT(a/b)
18200 END DEFine f_div
18210 :
18220 DEFine FuNction f_fromdec$(number,base%)
18230   LOCal numberl,result$(36),digit$(32),loop
18240   IF base%<2 OR base%>32 OR number<0 THEN RETurn -1
18250   numberl=number:result$=""
18260   digit$="0123456789ABCDEFGHIJKLMNOPQRSTUV"
18270   REPeat loop
18280     result$=digit$(f_mod(numberl,base%)+1)&result$
18290     numberl=f_div(numberl,base%)
18300     IF numberl<=0 THEN RETurn result$
```

```
18310   END REPeat loop
18320 END DEFine f_fromdec$
18330 :
```

Prof. Timo Salmi
School of Business Studies
University of Vaasa
Raastuvankatu 31
SF-65100 Vaasa
Finland