BASIC for Beginners

Lesson III

By David W. Ostler

So far in this series we have talked about commands that allow you to do some fairly impressive programming. But by no means can we cover all the commands utilized by the Color Computer BASIC language in only four installments.

We have already covered many common BASIC commands, giving you a solid base of programming skills to expand upon. You will eventually need to know how to make a program accessible to disk and tape input/output, so that you can save your results, but we'll cover disk and tape input/output in detail next month.

This month we will cover the commands necessary to determine whether a disk drive or cassette recorder is present. We will also discuss variable memory allocation and how to do logical comparisons of variables. In addition to a few new commands, we are presenting some variations on commands described in previous installments.

DIM

The DIM command sets up memory for use by predefined variable *arrays*.

An array is a group of variables with attached labels that relate them to a particular label. An example is an array that has 10 parts, all labeled M. The variables have these assignments: M(1), M(2), M(3), M(4), M(5), M(6), M(7), M(8), M(9), M(10).

Please note that each variable is unique and can be manipulated independently of the others. To display or manipulate the variable, you must access its label and assignment. The proper syntax for the command that lets you enter a new amount in a variable is DIM*x*(*yyy*), where *x* is the variable

---

Dave Ostler is an IC layout designer and the systems manager for a CAD mainframe system. He teaches CAD and electronics at Guilford Technical Community College. Dave is married and has three children, Avis, Chuck and Erik.

the array sets up (also called the label) and *yyy* is the number of variables in the group to be defined. The DIM command must always take place early in the program, *before* use of the CLEAR command.

PEEK

The PEEK command allows you to look at memory locations and determine various function values in the computer such as printer baud rate, disk or tape I/O, whether LIST and LLIST are disabled, etc.

The proper syntax for this command is PEEK (*xxx*), where *xxx* is the location to be examined or "peeked." Also, the value returned can be viewed only when used in conjunction with a variable, as in these lines:

```
10 A = PEEK(xxx)
20 PRINT A
```

Logical Comparisons

A valuable function of the BASIC language, for the Color Computer as well as other computers, is the ability to compare variables in a logical manner. *Logical comparison* determines whether variables are generated by program manipulation or entered by an external source by comparing the results of two variables. The logical comparison operators are AND, NOT, OR. Here is an example of logical comparison:

```
10 IF (A = X AND B = Y) THEN
GOSUB 1000
```

This line is basically an IF/THEN command, but with something extra — the logical comparison operator AND. The command line reads: If A equals X *and* B equals Y, then jump to the subroutine at Line 1000. Try this one:

```
10 IF NOT (A = X AND B = Y) THEN
GOSUB 1000
```

This line is also basically an IF/THEN command, but uses the logical comparison operator NOT. This command line reads: If A does *not* equal X *and* B does *not* equal Y, then jump to the subroutine at Line 1000. Try another:

```
10 IF (A = X OR B = Y) THEN GOSUB
1000
```

Again, the line is basically an IF/THEN command, but it includes the logical comparison operator OR. The command line reads: If A equals X *or* B equals Y, then jump to the subroutine at Line 1000.

These examples cover some of the ways to use the logical operators. The commands can be combined to obtain very elaborate logical comparisons of variables and are invaluable in programming.

LINEINPUT

The LINEINPUT command is exactly like the INPUT command covered in Lesson 1 (September 1987, Page 27), but with one exception: Where the INPUT command restricted the entry of variables to characters without punctuation, the LINEINPUT command al-

Line	Description
0	a remarked line.
10	clears 1,000 bytes for variable storage, sets T equal to 100, sets N equal to 0, and dimensions variables B$, C$, D$ and E$ to the size of 10 variables each.
20	a remarked line.
30	sets A equal to the value peeked at location 188.
40	a remarked line.
50	sets B equal to the value peeked at location 116.
60-80	test the values of A and B and steer the program to the proper location after these tests.
85	forces a jump to Line 2000.
95	a remarked line.
100	clears the screen, prints text and the value of N, prints a blank line, and allows the entry of variable B$ as related to the dimensioned variable label, N.
110	prints a blank line and allows the entry of variable C$ as related to the dimensioned variable label, N.
120	prints a blank line and allows the entry of variable D$ as related to the dimensioned variable label, N.
130	prints a blank line and allows the entry of variable E$ as related to the dimensioned variable label, N.
140-170	clear the screen and print the text with the variables entered in lines 100, 110, 120 and 130, respectively.
180	prints the text at the desired locations.
190	sets I$ to an INKEY$ function and tests the keyboard for the conditions found in this line. These conditions are used to correct any errors found in the variables entered in lines 140 to 170.
200	adds one count to the variable label N and checks the value of N — if N is equal to 10, it forces the program to Line 6000.
210	clears the screen and prints the text at the desired locations.
220	sets I$ to an INKEY$ function and tests the keyboard for the conditions found in this line.
300-330	the error correction lines for the INKEY$ function called in lines 180 to 190. After the corrections are made, the line forces a jump to Line 140.
900	sets up a FOR/NEXT value for B. Note: When displaying variables entered in arrays, you must use the variable label used in incrementing the array (at this time we are using N, as a label), minus 1 to count down the variable. Remember, the computer always counts — in this case, from 0 to 10.
910-940	display the text with its associated variables that are related to the variable array label value, B.
950	prints the text at the location.
960	sets I$ to an INKEY$ function and, if any key is pressed, will continue on with the program.
1000-2010	subroutines called by previous lines. The subroutine starting at Line 1000 identifies the type of computer system that the program has detected. The subroutine starting at Line 2000 is the menu for the data entry of the program.
5000	terminates the program.
6000	prints the message that the maximum file size for the array defined has been reached, and then sends the program to Line 900.

lows punctuation in the variables entered. That's the only difference between the two commands.

Programming Exercise:

Using the methods presented in this article and the listing, write a program that will allow you to enter items you would want to list or find later in the program into arrays.

It is often difficult for new programmers (and, sometimes, old programmers, too!) to decipher the meaning of a line of BASIC code. I have embedded remark statements in the program shown in the listing to help. Refer to the chart on Page 24 for a line-by-line description.

In the final installment, Lesson IV, we will take this month's program, add data I/O, and enhance it further to allow easier data entry and correction.

Hints and Tips

Nothing puts a damper on a struggling beginner more quickly than trying to edit a program with BASIC's built-in editor. I find it cumbersome and difficult to use.

To make programming easier, you can use word processors such as *VIP* or *Telewriter* to write your programs. Save these programs in ASCII (SAVE" *filename*",A) with the proper extensions, etc. It may take longer to load and save, but the editing capabilities of these programs make this a minor inconvenience.

Those of you who want to know more about the commands available for your computer can purchase the TRS-80 *Pocket Handbook* from Radio Shack (Cat. No. 62-2024). It is one of the best investments you can make to assist you in learning programming. And if you're interested in learning more about the peeks and pokes available for the Color Computer, I recommend you read *500 POKES, PEEKS 'N EXECS for the TRS-80 CoCo*, marketed by Microcom Software.

Remember: Work smarter, not harder! □

The listing: DATABASE

```
Ø 'BASIC NAME DATABASE PROGRAM.
THIS PROGRAM IS TO BE USED WITH
THE BASIC PROGRAMMING COURSE
WRITTEN BY DAVID W. OSTLER, COPY
RIGHT 1987
1Ø CLEAR1ØØØ:T=1ØØ:N=Ø:DIMB$(1Ø)
:DIMC$(1Ø):DIMD$(1Ø):DIME$(1Ø)
2Ø 'CHECK FOR TAPE OR DISK SYSTE
M
3Ø A=PEEK(188)
4Ø 'CHECK FOR 16K OR 64K SYSTEM
5Ø B=PEEK(116)
6Ø IF(A=14 AND B=127) THEN GOSUB
1ØØØ
7Ø IF(A=6 AND B=127) THEN GOSUB1
Ø1Ø
8Ø IF(A=6 AND B=63) THEN GOSUB1Ø
2Ø
85 GOTO2ØØØ
95 'FILES ENTERED HERE
1ØØ CLS:PRINT"ADDRESS DATABASE #
 OF FILES";N:PRINT:LINEINPUT"ENT
ER NAME
     ";B$(N)
11Ø PRINT:LINEINPUT"ENTER ADDRES
S
";C$(N)
12Ø PRINT:LINEINPUT"ENTER CITY,
ST, &ZIP                  ";D$(N)
13Ø PRINT:LINEINPUT"ENTER TELEPH
ONE NO.
";E$(N)
14Ø CLS:PRINT:PRINT"1. NAME- ";B
$(N)
15Ø PRINT:PRINT"2. STREET-";C$(N
)
16Ø PRINT:PRINT"3. STATE- ";D$(N
)
17Ø PRINT:PRINT"4. PHONE- ";E$(N
)
18Ø PRINT@357,"PRESS <C> TO CONT
INUE":PRINT@399,"OR":PRINT@416,"
  PRESS THE NUMBER TO CORRECT"
19Ø I$=INKEY$:IFI$=""THEN19ØELSE
IFI$="1"THEN3ØØELSEIFI$="2"THEN3
1ØELSEIFI$="3"THEN32ØELSEIFI$="4
"THEN33ØELSEIFI$="C"THEN2ØØELSE1
9Ø
2ØØ N=N+1:IFN=1ØGOTO6ØØØ
21Ø CLS:PRINT@456,"ANOTHER ENTRY
```

About The One-Liner Contest . . .

THE RAINBOW's One-Liner Contest has now been expanded to include programs of either one or two lines. This means a new dimension and new opportunity for those who have "really neat" programs that simply just won't fit in one line.

Here are the guidelines: The program must work in Extended BASIC, have only one or two line numbers and be entirely self-contained — no loading other programs, no calling ROM *routines*, no poked-in machine language code. The program has to run when typed in directly (since that's how our readers will use it). Make sure your line, or lines, aren't packed so tightly that the program won't list completely. Finally, any instructions needed should be *very* short.

Send your entry (preferably on cassette) to:

THE RAINBOW One-Liner Contest
P.O. Box 385
Prospect, KY 40059

```
 (Y/N)"
22Ø I$=INKEY$:IFI$=""THEN22ØELSE
IFI$="Y"THEN1ØØELSEIFI$="N"THEN9
ØØELSE22Ø
3ØØ CLS:PRINT:PRINT:LINEINPUT"EN
TER NAME
      ";B$(N):GOTO14Ø
31Ø CLS:PRINT:PRINT:LINEINPUT"EN
TER ADDRESS
        ";C$(N):GOTO14Ø
32Ø CLS:PRINT:LINEINPUT"3. STATE
- ";D$(N):GOTO14Ø
33Ø CLS:PRINT:PRINT:LINEINPUT"EN
TER TELEPHONE NO.
        ";E$(N):GOTO14Ø
9ØØ FORB=ØTO N-1
91Ø CLS:PRINT:PRINT"1. NAME- ";B
$(B)
92Ø PRINT:PRINT"2. STREET-";C$(B
)
93Ø PRINT:PRINT"3. STATE- ";D$(B
)
94Ø PRINT:PRINT"4. PHONE- ";E$(B
)
95Ø PRINT@355,"PRESS ANY KEY TO
CONTINUE"
96Ø I$=INKEY$:IFI$=""THEN96Ø
97Ø NEXTB
98Ø GOTO2ØØØ
1ØØØ CLS:PRINT@23Ø,"32/64K DISK
SYSTEM":FORX=1TO1ØØØSTEP1:NEXTX:
RETURN
1Ø1Ø CLS:PRINT@228,"32/64K CASSE
TTE SYSTEM":FORX=1TØ1ØØØSTEP1:NE
XTX:RETURN
1Ø2Ø CLS:PRINT@229,"16K CASSETTE
 SYSTEM":FORX=1TO1ØØØSTEP1:NEXTX
:RETURN
2ØØØ CLS:PRINT:PRINT" WELCOME TO
 THE BASIC DATABASE ":PRINT:PRIN
T"       WOULD YOU LIKE TO:":PRI
NT:PRINT:PRINT"     S)TART A NEW
 DATABASE":PRINT:PRINT"          E)
ND THIS PROGRAM":PRINT@458,"[SEL
ECT ONE]"
2Ø1Ø I$=INKEY$:IFI$=""THEN2Ø1ØEL
SEIFI$="S"THEN95ELSEIFI$="E"THEN
5ØØØELSE2Ø1Ø
5ØØØ CLS3:PRINT@224,"       REBO
OTING TO BASIC":SOUND2ØØ,2:SOUND
1ØØ,3:FORX=1TO1ØØØSTEP1:NEXTX:CL
S:END
6ØØØ CLSØ:PRINT@224,"   MAXIMUM
FILE SIZE REACHED":SOUND2ØØ,2:SO
UND1ØØ,3:FORX=1TO1ØØØSTEP1:NEXTX
:GOTO9ØØ
```