LOOP

PURPOSE

Execute a sequential loop.

DESCRIPTION

A sequential loop is one that has a defined start and stop value and a constant increment. The values for the start, increment, and stop can have real values (i.e., DATAPLOT is not limited to integer loops). DATAPLOT loops can have either a positive or a negative increment. The start, stop, and increment values can be either integers or real numbers.

SYNTAX

LOOP FOR <par> = <start> <inc> <stop>

where <par> is a parameter that specifies the loop index variable;

<start> is a number or parameter that is the value for <param> on the first iteration of the loop;

<inc> is a number or parameter that <param> is incremented by after each iteration is completed;

and <stop> is a number or parameter that determines when the loop is terminated (i.e., when <param> exceeds this value, no more iterations are performed).

EXAMPLES

LOOP FOR K = 1 1 100
LOOP FOR K = START INC STOP
LOOP FOR K = 10 -2 1
LOOP FOR X = 0.1 0.001 0.2

NOTE 1

The stop condition is tested at the end of the loop. This means all loops are executed at least once even if <stop> is less than <start>.

NOTE 2

Loops can be nested up to 7 levels. A unique index variable should be used for each loop.

NOTE 3

A maximum of 200 commands can be contained in a loop. If you need more, put some of the commands in a macro file and use the CALL command.

NOTE 4

Loops only save the first 80 characters of a command. If you use the continue character ("...") to stretch a command over 2 lines, this counts as a single command and only the first 80 characters are stored. For more than 80 characters, do something like the following:

LET STRING T = <part of the command line greater than 75 columns>
<com> ^T

where <com> contains the beginning of the the command line.

NOTE 5

The BREAK LOOP command can be used to terminate a loop early. This can be used to implement a DO WHILE type loop (LOOP and IF are the only control structures that DATAPLOT supports). This is a recent command, so earlier versions of DATAPLOT provide no mechanism for exiting a loop early.

NOTE 6

IF blocks can be nested inside of a loop and a loop can be nested inside an IF block.

NOTE 7

Although loops can be used for data manipulations, it is more efficient to do this without loops when possible. As a rule of thumb, it is usually efficient to loop over the number of variables (i.e., columns) while it is usually rather slow to loop over the number of observations (i.e., rows), particularly if N is fairly large. DATAPLOT's wide array of data manipulation commands combined with clever use of tag variables and the SUBSET and RETAIN commands can often be used to avoid writing loops. The following example (the data file has 527 elements) ran in 120.7 CPU seconds on a Sun SPARC. After recoding, it ran in 9.4 CPU seconds. Original version:

```
FEEDBACK OFF
LET N=75
READ CR.DAT OX A
```

```
LOOP FOR K = 1 1 N
        LET K1 = 7*K-6
        LET K2 = 7*K-5
        LET K3 = 7*K-4
        LET K4 = 7*K-3
        LET K5 = 7*K-2
        LET K6 = 7*K-1
        LET K7 = 7*K
        LET OXX=OX(K1)
        LET OX1(K)=OXX
        LET OXX=OX(K2)
        LET OX2(K)=OXX
        LET OXX=OX(K3)
        LET OX3(K)=OXX
        LET OXX=OX(K4)
        LET OX4(K)=OXX
        LET OXX=OX(K5)
        LET OX5(K)=OXX
        LET OXX=OX(K6)
        LET OX6(K)=OXX
        LET OXX=OX(K7)
        LET OX7(K)=OXX
END OF LOOP
DELETE OX
LET SUM1=0; LET SUM2=0
LET SUM3=0; LET SUM4=0
LET SUM5=0; LET SUM6=0
LET SUM7=0
LOOP FOR J = 1 1 N
        LET OXX1=OX1(J)
        LET SUM1=SUM1+OXX1
        LET OXX2=OX2(J)
        LET SUM2=SUM2+OXX2
        LET OXX3=OX3(J)
        LET SUM3=SUM3+OXX3
        LET OXX4=OX4(J)
        LET SUM4=SUM4+OXX4
        LET OXX5=OX5(J)
        LET SUM5=SUM5+OXX5
        LET OXX6=OX6(J)
        LET SUM6=SUM6+OXX6
        LET OXX7=OX7(J)
        LET SUM7=SUM7+OXX7
END OF LOOP
```

Recoded version:

```
FEEDBACK OFF
LET N=75
READ CR.DAT OX A
LET NLAST = 7*N
LET KTAG = PATTERN 1 2 3 4 5 6 7 FOR I = 1 1 NLAST
.
LOOP FOR K = 1 1 7
        LET OX^K = OX
        RETAIN OX^K SUBSET KTAG = K
        LET SUM^K = SUM OX^K
END OF LOOP
```

Loops that involve computing lags (e.g., Y(I) = X(I-2)) can also be computed without loops. For example, the following code computes the number of turning points where a turning point is defined as a point where the series changes direction. That is, the point is smaller than both the point before it and the point after it or it is larger than both of these points. The analyst originally coded it using loops and performing a direct comparison. Coding it without loops as below obtains the same results much faster.

```
. Calculate number of turning points up and number of turning points down.
LET Y = NORMAL RANDOM NUMBERS FOR I = 1 1 1000
LET N = SIZE Y
LET N1 = N - 1
LET N2 = N - 2
LET Y1 = Y
RETAIN Y1 FOR I = 1 1 N2
LET Y2 = Y
RETAIN Y2 FOR I = 2 1 N1
LET Y3 = Y
RETAIN Y3 FOR I = 3 1 N
LET TAG1 = Y2 - Y1
LET TAG2 = Y2 - Y3
.
LET TPDOWN = 0 FOR I = 1 1 N2
LET TPDOWN= 1 SUBSET TAG1 < 0 SUBSET TAG2 < 0
LET NUMDOWN = SUM TPDOWN
LET TPUP = 0 FOR I = 1 1 N2
LET TPUP = 1 SUBSET TAG1 > 0 SUBSET TAG2 > 0
LET NUMUP = SUM TPUP
LET NUMTOT = NUMDOWN + NUMUP
PRINT "Number of turning points up = ^NUMUP"
PRINT "Number of turning points down = ^NUMDOWN"
PRINT "Number of turning points total = ^NUMTOT"
```

DEFAULT

None

SYNONYMS

None

RELATED COMMANDS

END OF LOOP	=	Terminate a loop.
BREAK LOOP	=	Exit a loop early.
IF	=	Conditionally execute commands.

APPLICATIONS

Program control structure

IMPLEMENTATION DATE

Pre-1987

PROGRAM

```
LET Y1 = NORMAL RANDOM NUMBER FOR I = 1 1 100
LET Y2 = EXPONENTIAL RANDOM NUMBERS FOR I = 1 1 100
LET NU = 20
LET Y3 = T RANDOM NUMBERS FOR I = 1 1 100
LET Y4 = CAUCHY RANDOM NUMBERS FOR I = 1 1 100
LET STRING T1 = NORMAL RANDOM NUMBERS
LET STRING T2 = EXPONENTIAL RANDOM NUMBERS
LET STRING T3 = T RANDOM NUMBERS
LET STRING T4 = CAUCHY RANDOM NUMBERS
LET X = SEQUENCE 1 1 100
MULTIPLOT 2 2; MULTIPLOT CORNER COORDINATES 0 0 100 100
LOOP FOR K = 1 1 4
    TITLE ^T^K
    HISTOGRAM Y^K
END OF LOOP
END OF MULTIPLOT
```