Collection of Source on local variables#

The concept of local variables involves assigning parameter names to the values on the data stack on entry into a word. The parameter names can then be used instead of manipulating the data stack directly. The scheme generally involves moving the parameters from the data stack to some other location like the return stack or a dedicated buffer region. This implies a certain amount of overhead not normally associated with entering the word. There is also some overhead involved when the word is exited and the local variables are freed. The advantages come from not using up processing time with a series of DUP, SWAP, OVER, ROT, ROLL, or PICK functions.

For simple, one- or two-line words (the kind of Forth words everyone should strive to write anyway), the use of local variables does not help that much. A Forth programmer should train himself to visualize at least a three-deep stack in his head. By using comments with stack pictures, it is relatively easy to keep track. But for words that just have to have a lot of input parameters (graphics commands for example), using locals makes the code much easier to follow and can actually improve performance.

I've seen several different implementations of local variables. They range from the very simple to the very complex. Regardless of the complexity of implementation, they all seem quite easy to use, however. The parameter names assigned to the input stack can be either static, meaning that the same names are used in every case, or dynamic, meaning that they are assigned by the programmer at compile time. The implementations I have seen are all in high level code, but of course, they can be implemented in assembly code for greater performance. Implementations that use the return stack use a set of specialized return stack words that are transparently inserted into the compiled code by immediate words. These words automatically take into account changes in the depth of the return stack caused by DO-LOOPs, >R, R>, etc. I also saw an object-oriented approach that defines stack frame objects with their own set of operators. There are probably many more ways of doing it that I haven't seen yet.

After reviewing a set of files I downloaded from a local bulletin board system dedicated to Forth, the East Coast Forth Board, I selected the following code as an example of implementing local variables. It is the simplest possible implementation I could find (1 block of code). It was written by Wil Baden for the F83X Forth systems, but except for the word CELLS (which is just another word for "2*") it looks to me to be Forth-83 Standard. It's not the prettiest Forth code in the world, but it should be easy enough to figure out by reading the text.

```
SCR # 0
  0 ( Simple implementation of local variables.      WWB 01JAN87WWB)
  1 This is an attempt to implement local variables  as  simply  as
  2 possible.  If the effort proves to be worthwhile then it should
  3 be redone in assembler.
  4
  5 An earlier experiment demonstrated that pushing local variables
  6 onto  the  return  stack  is  too restrictive.  Local variables
  7 should be usable within DO-loops.
  8
  9
 10
 11
 12
 13
 14
 15

SCR # 1
  0 ( Simple implementation of local parameters.     WWB 01JAN87WWB)
```

```
  1 100 ( Multiple of two ) CELLS CONSTANT /LOCALS
  2 CREATE 'LOCALS  /LOCALS ALLOT
  3 VARIABLE ^LOCALS   'LOCALS /LOCALS + ^LOCALS !
  4 VARIABLE 'OP   ' @ 'OP !
  5 : LOCAL  CREATE CELLS , DOES> @ ^LOCALS @ + 'OP @EXECUTE ;
  6 0 LOCAL n1   1 LOCAL n2   2 LOCAL n3   3 LOCAL n4   4 LOCAL n5
  7 5 LOCAL n6   6 LOCAL n7   7 LOCAL n8   8 LOCAL n9
  8 : LOCALS  ( ... n -- )
  9    ^LOCALS @ OVER CELLS - 'LOCALS U<
 10    IF 'LOCALS DUP /LOCALS 2/   DUP UNDER+  ( ... n a a' len)
 11       DUP ^LOCALS +!  CMOVE> THEN ( ... n)
 12    0 DO CELL NEGATE ^LOCALS +!  ^LOCALS @ ! LOOP  ['] @ 'OP ! ;
 13 : LOCAL!  ( n a -- ) !   ['] @ 'OP ! ;
 14 : ->  ( value ^ localname -- ) ['] LOCAL! 'OP ! ;
 15 : FREE  ( n -- ) CELLS ^LOCALS +! ;

SCR # 2
  0 ( Example of use of local parameters.            WWB 10APR86WWB)
  1 : GCD  ( n n -- greatest common divisor.)
  2    BEGIN ?DUP WHILE TUCK MOD REPEAT ;
  3 : ROTATE  ( addr len rotation | stride -- : Rotate string.)
  4    2DUP GCD   4 LOCALS   n3 0= IF EXIT THEN  ( - )
  5    n1 n4 BOUNDS                       ( addr+stride addr)
  6    DO  I C@ I n2 0                    ( char addr' len 0)
  7       DO  n3 -   DUP n1 U<   n2 AND + ( char addr'')
  8          DUP >R   DUP C@ >R C!        ( - R: addr'' char')
  9          R> R> n4                  ( char' addr'' stride R: - )
 10       +LOOP 2DROP                     ( - )
 11    LOOP  4 FREE ;
 12 ( Try this without variables, local or otherwise.)
 13
 14
 15

SCR # 3
  0
  1 User defined transient names were intentionally not implemented
  2 as  well  as  automatic  freeing.  Even when they are local the
  3 number of variables should be kept small, and the  overhead  is
  4 not  justified.   The  ability  to  use  the  locals which were
  5 defined in the word that called you also seems valuable.
  6
  7 The  stack  area  is twice as large as anticipated to side-step
  8 the problems of error restart.  On stack overflow  the  earlier
  9 half  of  the  stack  is discarded.  No check is made for stack
 10 underflow.
 11
 12
 13
 14
 15

SCR # 4
  0 ( Simple implementation of local parameters.     WWB 10APR86WWB)
  1 There are nine, pre-named, local variables, N1 .. N9 .  These
  2    can be used for arguments to the routine taken from the
  3    stack or additional local values known in the routine.
  4 Local variables are defined at the beginning of the definition
  5    of a word.  Example:
```

```
  6          4 LOCALS
  7 This assigns the top 4 values on the parameter stack to N1 N2
  8    N3 and N4 with N4 containing what was the top of the stack.
  9 The previous values of N1 .. N4 can be gotten by N5 .. N8 .
 10 The user must deallocate the locals used before exiting by
 11          n FREE
 12    where n is the number of locals that were assigned.
 13 New values can be given to locals by
 14          value -> Nx
 15

SCR # 5
  0 ( Simple implementation of local parameters.     WWB 05MAY86WWB)
  1
  2 The value of local variables can be seen in the following
  3 definition to draw a box, using MOVE-TO and DRAW-TO .
  4
  5 : BOX  ( top left bottom right -- )
  6    4 LOCALS
  7    N1 N2 MOVE-TO
  8    N1 N4 DRAW-TO
  9    N3 N4 DRAW-TO
 10    N3 N2 DRAW-TO
 11    N1 N2 DRAW-TO
 12    4 FREE ;
 13
 14
 15
```

-- Lee Brotzman (FIGI-L Moderator)
-- BITNET: ZMLEB@SCFVM   SPAN: CHAMP::BROTZMAN   GEnie: L.BROTZMAN