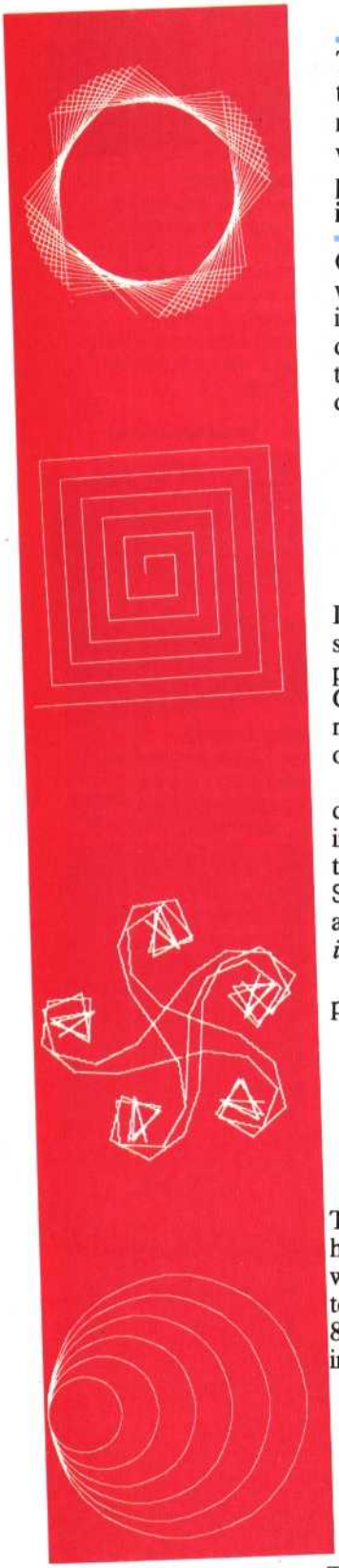
REPEAT PERFORMANCE



The LOGO language uses the mathematical technique of recursion (an instruction that refers to itself) to great effect. Coupled with variable inputs, the use of recursion in procedures can produce some extremely interesting results.

One of the first programs we defined in the course was a procedure to draw a square. The definition instructed the turtle to move forward a certain distance, turn right 90 degrees and repeat those two steps three more times. Here is another way to draw a square:

```
TO SQUARE
  FD 50
  RT 90
  SQUARE
END
```

If you were to try this out, the turtle would draw a square and then carry on moving around the perimeter of the square until you pressed Control-G or BREAK. The most noticeable thing about this new SQUARE procedure is that it calls itself — in other words, it is 'recursive'.

When this procedure is run, LOGO fetches the definition of SQUARE and begins to obey the instructions. The turtle is moved FORWARD 50 and then turned RIGHT 90. The next instruction is SQUARE, so LOGO fetches the definition of SQUARE and begins to obey it. This will go on *ad infinitum* if the program is not interrupted.

It is also possible to use recursive calls in procedures that require inputs:

```
TO POLY :SIDE :ANGLE
  FD :SIDE
  RT :ANGLE
  POLY :SIDE :ANGLE
END
```

This procedure can produce all the polygons we have defined so far in the course (see page 545) as well as many we haven't looked at (you might like to try using the procedure with an angle value of 89). It is also possible to change the value of the input in the recursive call. Thus:

```
TO POLYSPI :SIDE :ANGLE
  FD :SIDE
  RT :ANGLE
  POLYSPI ( :SIDE + 5 ) :ANGLE
END
```

The only difference between this procedure and POLY is that five is added to the value of SIDE each time it is called. So if you began with POLYSPI 10 90, then the first call would draw a line of length 10, the second would be 15, then 20, and so on. The result is a spiral. You might like to experiment with different inputs: 10 90, 10 95, 10 120, 10 117, 10 144 and 10 142 are interesting starters. You could also try modifying the procedure — one possibility is to change addition to subtraction or multiplication.

Here's a similar procedure that increments the angle rather than the side value:

```
TO INSPI :SIDE :ANGLE :INC
  FD :SIDE
  RT :ANGLE
  INSPI :SIDE ( :ANGLE + :INC ) :INC
END
```

Try various inputs: 5 0 7, 10 40 30, 15 2 20, 5 30 20 will do initially. Why do some shapes close and others not? Can you find a rule?

The simple repetition of a piece of code is referred to as *iteration*. LOGO uses REPEAT for this purpose, while other languages use a variety of constructs, such as FOR... NEXT, REPEAT... UNTIL, and WHILE... WEND. However, LOGO relies much more on recursion than it does on iteration. If you've programmed in other languages you may have difficulty in breaking away from using iteration, but turtle graphics is ideal for experimenting with recursive calls.

STOP RULES

All of the recursive procedures we have looked at so far continue repeating indefinitely. Clearly, we need a way to make a procedure stop at some point. Taking the SQUARE procedure as our example, a possible place to stop it would be after it has drawn a complete square and the turtle's heading is back to 0. This can be done by adding a 'stop rule' to the procedure:

```
TO SQUARE :SIDE
  FD :SIDE
  RT 90
  IF HEADING = 0 THEN STOP
  SQUARE :SIDE
END
```

The new primitives are STOP and IF. The first of these commands causes a procedure to stop running and returns control to the calling procedure. An IF statement is LOGO's way of making decisions. IF is followed by a condition, and THEN by an action that is carried out if the condition is true.