The ruler's edge revisited

In Session

Ray Polivka
polivkar@acm.org

"What a great inquisitive programming problem to give to a APL programming class!" was the thought that entered my mind when I read Stephen Taylor's article "The ruler's edge" that appeared on pages 118-120 in the January 2008 issue of *Vector*. I have recently taken a fancy to the term *inquisitive programming* as opposed to *software development programming*. The term was coined by Brian Hayes in his excellent paper entitled "Calculemus" in the *American Scientist* Vol.96 N°5, September-October 2008. One could also say that *inquisitive programming* is another term for personal problem solving. So be it. But this is the initial path that students should take when either just learning to program or learning a new programming language. Stephen's problem is a fine example to offer along this path.

Since I am a person who learns by doing, I decided to create my own solution. I wrote it in conventional defined-function form in APL2, since I am not very agile with the dynamic-function notation. Furthermore, I will be able to use it with any of the APL vendor's APL systems. Here are my two slightly different solutions.

```
[0]   Z←D RULERA L;I;N;T;IO;PW
[1]  ⍝L: Length of the ruler
[2]  ⍝D: Distance between ticks
[3]  ⍝Z: a ruler of length L with ticks every D places
[4]  ⍝   modelled after S Taylor's problem in Vector
[5]   ⎕PW←L+⎕IO←1
[6]   Z←Lρ((D-1),1)/'¯^'
[7]   I←(Z='^')/⍳L
[8]   N←⍉⍕,[''']I
[9]   T←((↑ρN),L)ρ' '
[10]  T←[;I]←N
[11]  Z←T,[1]Z
    ∇ 2008-09-18 19.03.04 (GMT-4)
```

Or with a different ending:

```
[0]   Z←D RULERB L;I;N;T;IO;PW
      …
[9]   E←Lρ((D-1)ρ0),1
[10]  T←E\[2]N
[11]  Z←T,[1]Z
    ∇ 2008-09-18 19.03.19 (GMT-4)
```

Actually, I modified Stephen's problem slightly. Rather than having the number straddle the tick mark, the numbers appear vertically about the tick mark. This allows one to display a number every position in the ruler if one so chose. Thus,

```
      3 RULERA 17
            1  1
  3  6  9  2  5
--^--^--^--^--^--
```

There are several ways in which this problem might be used in an APL programming class. In a more advanced class one could just state the problem and see what comes back. However, when the given solution illustrates some perhaps unusual approach or less-used aspects of APL, then presenting the solution directly may be better for an in-class discussion. Or one could present a solution with just comment symbols following all or selected lines of the function. The student now is asked to fill in a set of comments. This way the student can determine what the function does at specific lines while determining how the function achieves its purpose. Then too, the student can use the available debugging tools to ferret out the behaviour of the function. For the Dyalog APL programmers, another exercise might be to translate the defined function into its dynamic function form too.

As an additional exercise, the student could be asked to modify the function so that the ruler prints vertically using `|` and `+`. Thus

```
      3 RULERC 7
  |
  |
3+
  |
  |
6+
  |
```

This certainly can be a fun learning exercise, Stephen, as well as a morning honing exercise. Now where is my cup of tea?