Comment on   A Note on Dynamic Arrays in PASCAL

N. Wirth, ETH Zürich, Switzerland

In his recent contribution B.J. MacLennan hopes to generate a lively
discussion on a proposal to introduce dynamic arrays into the language
PASCAL [1] . As designer of this language I feel particularly
challenged to comment.

The absence of dynamic arrays is clearly the most frequently cited
shortcoming of PASCAL. Both disadvantages and benefits of this lack
have been expounded before and need not be discussed here [2,3]. It
is clear that a simple and cheap means of introducing dynamic arrays
when needed would be most welcome. Hence, Mr. MacLennan's attempt is
certainly well motivated. It also tackles the problem - and the
language - at the one place that is most likely to yield success,
namely where dynamic allocation is provided. Yet, I must admit
reservation about the particular "solution" presented. It epitomizes
the art of language grafting, and with due respect for the cleverness
of the grafter I dare to point out some misconceptions underlying this
art.

The indicated solution to the array problem is natural, even evident,
to the professional PASCAL programmer, because he has learned to see
the implementation of the various facilities behind their facade.
However, to the programmer dealing exclusively with the language's
high-level abstractions, the proposed formulation appears as highly
artificial and unmotivated. To him the reason for this choice of
notation for dynamic arrays are obscure; the virtues of a high-level
language are tarnished and its purpose is compromised.

A second reservation against the proposed solution is that it suggests
generality where there is none. The variant record declaration offers
many more constructions than would be meaningful when declaring a
"varying" component.

Perhaps most important is the fact that introduction of dynamic arrays
in the language PASCAL presents no problems at all; merely admit
expressions instead of constants only in the bound specifications of
array declarations. But what Mr. MacLennan (and others) have tried
to achieve is the incorporation of dynamic arrays in their PASCAL
compiler in the cheapest possible way. Perhaps such solutions,
although valuable in the context of a particular project, should not
be considered as general extensions of a language, but rather as what
they are: fixes to achieve some desired effect in an expeditious way.

In order to end in a positive note, let me propose a compromise that
should satisfy the man in need and at the same time avoid deleterious
effects on the high-level character of the language.

1. Introduce a new construct that can be used in conjunction, with the definition of a pointer type only:
   $$\underline{type}\ T = \uparrow\underline{row}\ \underline{of}\ T_o$$

2. Extend the procedure <u>new</u> such that it allows the specification of a row length n for such types:
   ```
   new(t,n)
   ```

3. Introduce the functions <u>length</u> applicable to such rows:
   ```
   length(t)
   ```

4. Allow indexing of "rows":
   $$t\uparrow[i] \qquad 1 \leq i \leq n$$

(Evidently, one might introduce the two array index bounds instead of the length; use of <u>array</u> instead of <u>row</u> would then be appropriate.)
The obvious representation of such a row would be as a record whose first field contains the (unchangeable) length (or index bounds), and whose second field represents the array with elements of type T_o.
This compromise shares with all other proposals the drawback
that it extends rather than simplifies an already sufficiently complex language. It should therefore be followed only after careful deliberation.

References

1. B.J. MacLennan, "A note on dynamic arrays in PASCAL", SIGPLAN Notices <u>10</u>, 9, 39-40 (Sept. 1975)

2. O. Lecarme and P. Desjardins, "Reply to a paper by A.N. Habermann on the programming language PASCAL", SIGPLAN Notices <u>9</u>, 10, 21-27 (Oct. 1974)

3. N. Wirth, "An assessment of the programming language PASCAL", IEEE TSE, <u>1</u>, 2, 192-198 (June 1975).