0 → TOTAL;
  FOR X IN HYBRID H:
  X + TOTAL → TOTAL
ENDFOR

(The code generated is not given—it is left to the reader to realise that it is not particularly attractive.)

Summary
What does, and what does not, constitute 'convenience' in programming languages is somewhat a question of individual choice. The author feels that coding for iteration is more easily understood if, whenever a data path is involved, the nature of the data path is immediately apparent.

Like many language features, the above facility is decidedly no 'cure-all', but its use appears advantageous for a certain range of commonly occurring situations.

The techniques presented here are only suitable for iterations involving a single data path. This is quite a serious limitation (e.g. how does one handle element-by-element assignment?). More general techniques are being developed to handle such cases (this appears to involve the manipulation of text which represents program actions having multiple entries and/or multiple exits—and the use of 'connector functions' to combine these components in a more general manner than is allowed in 'conventional' structured programming—however that is a matter for further research).

One would expect the above techniques to produce (source) program code which is relatively insensitive to choices of data structure. This may allow convenient implementation of decisions concerning the mapping of abstract data structures to physical data structures when implementing very high level languages. However, this is again a topic of further research.

The author would like to thank the referee for useful suggestions on improving the presentation of this paper.

References