Reviews of Books and Papers in the Computer Field

DONALD L. EPLEY, Reviews Editor
D. W. FIFE, A. I. RUBIN, R. J. SHORT, H. S. STONE
Assistant Reviews Editors

Please address your comments and suggestions to the Reviews Editor: Donald L. Epley, Department of Electrical Engineering, University of Iowa, Iowa City, Iowa 52240

A. PROGRAMMING


The title of this paper is misleading. It should read “Suggestions for New Data Types and Manipulations for PL/I.” It is not about programming without data, but about accessing highly structured data in a systematic way. The goal is to minimize the difficulty of changing the data structure at a late date in the programming cycle.

To accomplish this, the author adds a lot of new statement types and phrases to PL/I. These will presumably be converted to “standard” PL/I by a preprocessor (not yet implemented). This approach, you will recall, was previously adopted by the SIMSCRIPT designers (also at the RAND Corporation) who added numerous statements and data types to FORTRAN to make it more suitable for simulation programming. This is actually a fairly sound idea.

The major contribution of the paper, though, is somewhat concealed by extravagant claims for the “Dataless Programming System” and by the unorganized presentation of a large number of new statements and phrases. The principal accomplishment is really in determining the fundamental operations usually performed on a linearly ordered collection of data. These can be summarized (taking a few liberties) by noting that we can do the following.

1) Designate any element of a data structure as the reference element.
2) Locate the next, previous, first, last, or numerically specified element of a data structure.
3) Perform an iteration using successive elements of a data structure as the reference element.
4) Search a data structure to find the first (next) element meeting a required condition.
5) Count the elements in a data structure (meeting a specified condition).

The data structures for which standard routines are to be supplied include the following.

1) Arrays.
2) Forward-linked lists.
3) Forward- and backward-linked lists.
4) Forward-linked rings.

Other data arrangements are added by supplying procedures for accessing, updating, inserting, and deleting elements. Unfortunately, the author does not mention storage allocation and garbage collection problems. These problems tend to become serious in the larger, more complex applications for which powerful techniques are most needed.

Although the adding of still “higher-level” facilities to a very high-level language such as PL/I cannot be opposed on principle, I wonder whether it is worth the effort. The example program in the paper, for instance, can be programmed just as easily in basic PL/I. Also, I would like to know why the compile-time facilities (macros) of PL/I were not employed. This type of thing is exactly what the compile-time features are intended for.

Robert M. McClure
Goddard Computer Science Institute
Dallas, Texas

B. MULTIPLE MEMORY SYSTEMS


The purpose of this paper is to explore the effect of multiple modules of memory together with multiple bus lines for providing effective communication between memory and processors or channel controllers. Apparently the study was undertaken to determine an optimum configuration of equipment for an installation to be made by a computer manufacturer to a university. It suffers from the necessary evil which abides in studying a particular system under specific limitations applying to that system. The author has made no attempt to generalize his results regarding priority and processor performance as affected by the number of modules and buses available.

The special restrictions which apply are in terms of channel controller timing and drum unit datum acquisition.