SNOBOL Bulletin

Number 9 December, 1970

If any readers of the SNOBOL Bulletin could exhume a working copy of the SNOBOL3 system for the IBM 7044, they could earn the gratitude of a colleague south of the border. A request for a deck with "instructions to load" has come from:

Adilson T. Medeiros, Assistant
Computer Division
Centro Brasileiro de Pesquisas Fisicas
Av. Wenceslau Bras 71 - ZC 82
Rio de Janeiro, Brasil

SNOBOL4, version 3.4 is available for CDC 6000 series equipment. This version was implemented at the University of Colorado and is being distributed by CDC. Questions and problems should be directed to

Mr. Sam Coleman
3045 Ash Avenue
Boulder, Colorado 80302

CAL SNOBOL, an alternate version of SNOBOL4 for the CDC 6000 series, has been discussed in earlier versions of this Bulletin. Comparisons of CAL SNOBOL and SNOBOL4, version 3.4 have been carried out at Colorado by Mr. Coleman. They indicate that since version 3.4 is just slightly faster (see below) and smaller than Version 2.0, it is still worthwhile to live with the restrictions imposed by CAL SNOBOL for production-type programs. This is especially true for I/O bound programs. While CAL SNOBOL I/O is a bit slower than assembly language programs, it is far faster than FORTRAN or FORTRAN-dependent Version 3.4 routines.

The Colorado University implementation is particularly slow when a large number of strings are generated. The time is spent looking for these strings in dynamic storage, inserting them when necessary. A test run that read 3,200 card images (with no additional processing) consumed 86% of the time in this area. This situation will be improved in a future update of the system.

Another of the inefficiencies uncovered in the CDC implementation of SNOBOL4 is the use of FORTRAN formatted I/O. Tests on several machines have shown that formatted I/O is slower than unformatted operations by factors ranging from 2 to 6. In some cases the overall speed of a program has been tripled by replacing the FORTRAN I/O.

As I see it, there are two reasons for using FORTRAN I/O in an implementation of SNOBOL4:

1) It eliminates the need for major I/O coding;
2) It allows the SNOBOL programmer to use formatted output.

The first reason is open to question for many contemporary operating systems, provided that I/O is limited to character strings. But this is, in fact, the case for input and programmer-created output (see p. 160 of "The SNOBOL4 Programming Language"). The system uses the macro OUTPUT to write various messages and statistics. OUTPUT makes use of format statements to convert both integers and reals for these messages. Version 3.4 has 27 calls on OUTPUT, 12 of which use integer conversion and 1 real conversion. I assert that no coding effort is avoided by this macro. Two other macros, INTSPC and REALST, must be written to convert integers and reals to strings. It is true that the latter is optional, but it must be implemented if real arithmetic is implemented. If real arithmetic is not implemented, then the one real value printed by OUTPUT cannot be computed! The University of Colorado implementation of Version 3.4 utilizes INTSPC and REALST to avoid FORTRAN arithmetic conversions.

Whether or not the second reason is valid depends upon the programming style of the user. SNOBOL4 provides far more sophisticated facilities for constructing output lines than does a FORTRAN format statement. Is any advantage really gained by making the latter accessible? Another point is the use of A1 format as a default option for output. This has disastrous consequences on computers which are not byte-oriented. It requires that the string be unpacked into a buffer by the output routine and then packed up again by the FORTRAN I/O. Wouldn't it be preferable to use a format appropriate to the word length of the computer? (To handle possible multiple-line output, the format would have to write an integral number of words.)

The Colorado University implementation uses FORTRAN I/O throughout. On input, it laboriously constructs a format string only to pass it to the FORTRAN routines for decoding. Conversion of this process to use system macros directly would drastically reduce the processing required. If SNOBOL programmers were willing to forgo the use of formats, the output system could be similarly converted.

I would appreciate comments from both users and implementors on these points for inclusion in the next issue of the Bulletin. Please send contributions to:

Prof. William M. Waite
Department of Electrical Engineering
University of Colorado
Boulder, Colorado 80302