In the last issue of the SNOBOL Bulletin I made some remarks regarding the use of Fortran I/O in implementing SNOBOL4. These comments were motivated primarily by consideration of the Control Data 6000 series implementation at the University of Colorado. Ferris E. McCormick, of the Research Computing Center at Indiana University, writes that they came to similar conclusions for their implementation on the CDC 3600 and provided their own I/O package:

“We wrote this package with the following guide-lines in mind:

1) We should buffer whenever feasible, because a system request in the 3600 was very time-consuming.

2) The standard input-output variable, INPUT, OUTPUT, and PUNCH should behave the way most programmers would want. For example, if the user should happen to assign a very long string to output, he should have carriage control characters inserted where needed, and the string printed on several lines (if necessary).

3) It should be easy for the user to generate tapes output efficiently. Furthermore, it should be in a form which could be read easily by other programs (for example, to print the tape at a later time). On the 3600, this meant that the tape should be blocked into fairly large physical records, with perhaps the beginning and/or ending of smaller logical records delimited within these physical records.

4) For convenience and efficiency, some special functions such as TRIM should be optionally a part of an I/O variable specification.

From the user's point of view, the major difference between this and the I/O specified in the Bell Manual [Griswold, R.E. et al. The SNOBOL4 Programming Language. Prentice-Hall, Inc. (1968)] was the method of association of an input (or output) attribute with a variable. Instead of specifying a FORTRAN format string, he had to provide a string which contained various parameters describing the form of physical record. At this time we would allocate buffer storage and establish a (three-word) descriptor for the variable. Thus, under normal circumstances we had to scan these parameters only once.

In conjunction with this, we made two further changes. One of these was the streamlining of the OUTPUT macro, and the other was changing the method of handling numbers. It
seemed ridiculous to us that it should be necessary to set up FORTRAN format statements just to print headings and compiler statistics. As a corollary to this, we decided that the macro did not have to call the I/O system at all, especially since our buffers could never contain any information when it was used. Thus, the OUTPUT macro became simply a system request to do a WRITE into the printer unit.

The changes we made to the internal form of numeric data are more interesting. We soon discovered that 24 bit integers and floating point numbers were not worth very much in the 3600. To get full word precision, we had to make each number a part of a structure, similar to strings. The descriptor for a number contained a pointer to this structure rather than the number itself. The structure consisted of one word of the necessary linking information, and one word containing the number itself."

I have also received a comment on Fortran I/O from Mr. J. M. Newcomer, a graduate student at Carnegie-Mellon University. Although he is not a SNOBOL user, he has fallen foul of Fortran I/O in other symbol manipulation tasks. His major complaints are speed for character data and poorly designed interfaces for handling named files.

Dr. R. E. Griswold has clarified the position of the Bell Laboratories group with regard to the use of Fortran I/O as follows:

"While I agree with most of your points, I point out that use of FORTRAN I/O routines is optional to the implementor (many have now chosen I/O and file conventions more suitable to their environment). Our documentation on use of formats emphasizes that these vary from machine to machine, and the 360 is only an example. Implementors are expected to change the internal formats even if they use FORTRAN I/O.

SPITBOL has written its own I/O interface and added their own format conversion routines for compatibility. They support standard OS facilities. We expect to replace our I/O package with theirs."

I made the claim in the last issue that the macro REALST must be implemented if real arithmetic is implemented, and that real arithmetic must be available to compute the only real number actually printed by OUTPUT. M. D. Shapiro (Purdue University) took me to task for this statement:

"With respect to your comment in the SNOBOL Bulletin 9 about the necessity of implementing REALST, you will note from version 2 that I did not have REALST but did have the required output. This occurs because I am dealing with a fixed point output which is represented internally in floating point for convenience (no scaling is necessary). Generalized real-to-string conversion is not required, but is included in our version 3.4."
Further investigation of the matter shows that the code alluded to is specially concocted for printing the time. In this case, the result is known to have two decimal places. Hence it can be multiplied by 100, converted as an integer, and then a decimal point inserted.

The main point of the discussion in the last issue was to establish that the use of Fortran I/O did not avoid the need for coding effort. Shapiro's example shows that this is not quite true. If one wishes to obtain the average execution time, but is not concerned about real arithmetic in general, one need only implement DVREAL. The Fortran I/O will then output the only real number ever created. This is contrary to the instructions in the Implementation Guide, which states that all real arithmetic operations (including REALST) should be implemented or omitted as a group.

The University of Colorado implementation of SNOBOL4 for CDC 6000 series has now been updated to version 3.6. Fortran I/O has been replaced throughout, but formatted output is still available using a home-built format cracker. A revision of Shapiro's method is used to print the average execution time. The outer system has been modified to obtain this average using integer arithmetic. OUTPUT inserts the decimal point before calling STPRNT to print the result as a string.

Purdue/IDA SNOBOL4 is also available for CDC 6000 series equipment. The last information which I had (10 February) was that it had been updated to version 3.4. An optional feature, not yet fully debugged, is linkage to external (usually Fortran or assembly language) functions. Both the Purdue/IDA and CU systems can be obtained through VIM.

I think that the Fortran I/O has been definitely proven as a source of inefficiency in several implementations of SNOBOL4. Would-be implementors should therefore examine this aspect of their design very carefully. We should now attempt to spotlight other problem areas. The existence of high-speed versions like CAL SNOBOL might lead us to suspect that the distributed macros are too dependent upon the structure of System/360. Some attempt should be made to either substantiate or refute this suspicion by making comprehensive measurements on implementations for other machines. It does little good to theorize — hard facts are necessary. Comments on instrumentation techniques or measurements should be sent to:

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