Preliminary Notes on CAL SNØBOL

On or about Tuesday, October 15, or shortly thereafter, a preliminary version of a new SNØBOL4 system will be available on the CDC 6400. The language implemented is basically that described by the Preliminary Report on the SNØBOL4 Programming Language, by Griswold, Poage, and Polonsky, dated November 22, 1967. Major differences at the compiler input level:

1. Replacement of keywords by corresponding standard procedures having the same capabilities.
2. Omission of truth predicates.

The array and data structure features have been implemented. Arrays may be created by execution of the standard procedure ARRAY. ARRAY(P) creates an array described by the prototype P. This prototype describes the indexing and dimensionality of the array. For example

VECTØR = ARRAY('10')

assigns a one-dimensional array of length 10 to VECTØR. The initial value of each element is the null string. Indexing ordinarily starts at 1. Other lower bounds may be specified by using a colon or slash to separate the upper and lower limits:

LINE = ARRAY('-5:5') or LINE = ARRAY('-5/5')

will create an array with a lower bound of -5 and an upper bound of 5. Additional dimensions in a prototype are separated by commas. Thus

BOARD = ARRAY('8,8')

defines an eight by eight array.

If the value of a variable is an array, an element in the array may be referred to through the variable. Square brackets following any array-valued variable are used to specify the element.

VECTØR[2] = EXP

assigns the value of EXP to the second element of VECTØR. There is no requirement that all values of an array be the same kind of object. If an index referring to an element of an array falls outside the range of the array, the array reference fails. Thus

OUTPUT = VECTØR[12]

would fail.
New data types may be created by means of the standard procedure DATA. The result of executing DATA(P) is to create a data type and define field functions as given in the prototype P. For example

DATA('NODE(FATHER, LS0N, RSIB, VALUE)')

creates a new data type NODE with four fields: FATHER, LS0N, RSIB, and VALUE. Execution of this data function defines a function NODE which creates objects of data type NODE. Hence

\[ N1 = NODE() \]

creates a NODE which becomes the value of N1. The NODE function has four arguments corresponding to the fields FATHER, LS0N, RSIB, and VALUE. These fields may be assigned when a NODE is created.

\[ N2 = NODE(N1.., 'X') \]

creates a NODE with the NODE 1 as the value of its FATHER field and X as the value of its VALUE field. The LS0N and RSIB fields are null.

Execution of the DATA function also creates field functions FATHER, LS0N, RSIB, and VALUE which refer to the fields of a node. Thus

\[ LS0N(N1) = N2 \]

assigns the NODE N2 to the LS0N field of N1.

The character set of CAL SBDB0L is the CDC 6000 display code, and several characters of the reference language have been transliterated. Also, composite characters are allowed for some reference characters, to aid punching on an 026 keypunch. The following table shows the equivalences.

<table>
<thead>
<tr>
<th>Reference Language</th>
<th>CAL SBDB0L</th>
</tr>
</thead>
<tbody>
<tr>
<td>'</td>
<td>(4-8)</td>
</tr>
<tr>
<td>&quot;</td>
<td>(11-5-8)</td>
</tr>
<tr>
<td>&lt;</td>
<td>([7-8]), or (/</td>
</tr>
<tr>
<td>&gt;</td>
<td>(0-2-8), or /)</td>
</tr>
<tr>
<td></td>
<td>v (11-0), or //</td>
</tr>
<tr>
<td>;</td>
<td>(2-8), or unary / (see below)</td>
</tr>
</tbody>
</table>

Blanks (defined as a string of one or more blank characters) are required in several places in a SBDB0L program. A blank may be an
operator, signifying concatenation or a pattern match, depending on its context. Also, the binary operators (addition, subtraction, multiplication, division, exponentiation, alternation, and $ and . assignments) require blanks on both sides of the actual operator (+, -, *, /, **, v or //, $, and . respectively). Blanks are optional in several other places: on both sides of a comma, equal sign, or colon, after a left parenthesis or left array bracket, and before a right parenthesis or right array bracket.

By means of the standard procedures INPUT and OUTPUT, variables may be associated with files. These functions have the form

\[
\text{INPUT}(V,F,L) \text{ and } \text{OUTPUT}(V,F,C)
\]

where

1. \( V \) is the variable to be associated.
2. \( F \) is the name of the file with which the association is to be made, and must conform to the SCOPE Operating System limitations on filenames: 1 to 7 alphanumeric characters, beginning with a letter.
3. \( L \) is the length of a string to be read on input. If the string on the external medium is shorter, trailing blanks will be added. If string on the external medium is longer, the remainder will be lost.
4. \( C \) is a carriage control character which is to be prefixed to all strings placed on the external medium. If \( C \) is null, strings will be placed on the external medium without charge.

CAL SNOBOL has been designed in such a way that there will be no inherent limitation on the size or complexity of objects other than the size of the core store. In particular:

1. Integer arithmetic will be able to handle numbers of any size.
2. There is no limit on the size of identifiers, number of parameters in a procedure call, number of subscripts in an array, etc.

Furthermore it will be guaranteed that at the time storage overflow is signalled, no space will remain wasted.

At present integer arithmetic is limited to 10 digits.
Two standard procedures are available for manipulating files.

ENDFILE(F)

writes a logical end-of-record on the file F. (This corresponds to a 7-8-9 card in an input deck.) A reference to an input associated variable fails if the file to which it is associated is positioned at a logical end-of-record. A file may contain several logical records.

Two variables have standard associations. The effect is that the following two procedure calls occur at the beginning of every program.

INPUT('INPUT', 'INPUT', 80)
OUTPUT('OUTPUT', 'OUTPUT', ' ', ' ')
REWIND (F)

rewinds the file F. Examples of the usage of these procedures.

ENDFILE('OUTPUT')
REWIND('INPUT')

Currently SNØBØL is on the system as a common file named SNØBØL4.

Following is a standard job set-up.

Job card - See SCOPE 3.1 Reference Manual
COMMON SNØBØL4.
SNØBØL4.

SNØBØL source program

Data

A field length of 14000₆ should be sufficient for small jobs. If this is insufficient, the SNØBØL processor will abort with the message: INSUFFICIENT STORAGE.

Assistance may be obtained from Charles Simonyi or Paul McJones. They are in South Hall Annex, phone 2-1492.