Report on Input-Output Procedures for ALGOL 60

1. Introduction

It was recognized in the IFIP/WG 2.1 on ALGOL that some procedures to be used in connection with input and output are considered as being primitives, which cannot be expressed otherwise than by means of a code body. Among these are the following ones:

insymbol
outsymbol
length
inreal (1)
outreal
inarray
outarray.

Apart from these primitives one needs in practice a fuller set of input-output procedures. However, the language ALGOL 60 is so flexible that different schemes of I/O procedures can be defined in it largely by means of the primitives mentioned above. A few examples of this will be given in section 4 of this report.

2. Definitions

It is recommended that, if not otherwise declared, the identifiers (1) will be associated with procedures which transfer values between the variables of the program and values carried in any kind of foreign media not otherwise accessible from the program.

The corresponding procedure declarations are:

procedure *insymbol* (*channel, string, destination*); **value** *channel*;
integer *channel, destination*; **string** *string*; ⟨procedure body⟩
procedure *outsymbol* (*channel, string, source*); **value** *channel, source*;
integer *channel, source*; **string** *string*; ⟨procedure body⟩
integer procedure *length* (*string*); **string** *string*; ⟨procedure body⟩
procedure *inreal* (*channel, destination*); **value** *channel*;
integer *channel*; **real** *destination*; ⟨procedure body⟩
procedure *outreal* (*channel, source*); **value** *channel, source*;
integer *channel*; **real** *source*; ⟨procedure body⟩
procedure *inarray* (*channel, destination*); **value** *channel*;
integer *channel*; **array** *destination*; ⟨procedure body⟩
procedure *outarray* (*channel, source*); **value** *channel*;
integer *channel*; **array** *source*; ⟨procedure body⟩

The procedure statements and the function designator calling these procedures must have the following forms:

insymbol (⟨arithmetic expression⟩ ⟨parameter delimiter⟩ ⟨string⟩
⟨parameter delimiter⟩ ⟨variable⟩)

outsymbol (⟨arithmetic expression⟩ ⟨parameter delimiter⟩ ⟨string⟩
⟨parameter delimiter⟩ ⟨arithmetic expression⟩)
length (⟨string⟩)
inreal (⟨arithmetic expression⟩ ⟨parameter delimiter⟩ ⟨variable⟩)
outreal (⟨arithmetic expression⟩ ⟨parameter delimiter⟩ ⟨arithmetic expression⟩)
inarray (⟨arithmetic expression⟩ ⟨parameter delimiter⟩ ⟨array identifier⟩)
outarray (⟨arithmetic expression⟩ ⟨ parameter delimiter⟩ ⟨array identifier⟩)

In all these cases, except for the call of length, the value of the first actual parameter must be a positive integer identifying an input or output channel available to the program.

3. Actions of the procedure bodies

The pair of procedures *insymbol* and *outsymbol* provides the means of communicating between foreign media and the variables of the program in terms of single basic symbols or any additional symbols. In either procedure the correspondence between the basic symbols and the values of variables in the program is established by mapping the sequence of the basic symbols given in the string supplied as the second parameter, taken in the order from left to right, onto the positive integers 1, 2, 3, ... Using this correspondence the procedure *insymbol* will assign to the type **integer** variable given as the third parameter the value corresponding to the next basic symbol appearing on the foreign medium. If this next basic symbol does not appear in the string given as the second parameter, the number 0 will be assigned. If the next symbol appearing in the input is not a basic symbol of ALGOL 60 a negative integer, corresponding to the symbol, will be assigned.

Similarly the procedure *outsymbol* will transfer the basic symbol corresponding to the value of the third parameter to the foreign medium. If the value of the third parameter is negative a symbol corresponding to this value will be transferred. It is understood that where the foreign medium may be used both for *insymbol* and *outsymbol*, the negative integer values associated with each additional symbol will be the same for the two procedures. More generally, if additional symbols are used the corresponding values must be given as accompanying information with the program (cf. the footnote to section 1 of the Revised ALGOL 60 Report).

The type procedure *length* is introduced to enable the calculation of the length of a given (actual or formal) string to be made (cf. example *outstring*). The value of *length*(*s*) is equal to the number of basic symbols of the open string enclosed between the outermost string quotes.

The two procedures *inreal* and *outreal* form a pair. The procedure *inreal* will assign the next value appearing on the foreign medium to the **real** type variable given as the second parameter. Similarly, procedure *outreal* will transfer the value of the second actual parameter to the foreign medium.

The representation of values on the foreign media will not be further described, except that it is understood that in so far as a medium can be used for both input and output a value which has been transferred to a given medium with the aid of a call of *outreal* will be represented in such a way that the same

value, in the sense of numerical analysis (cf. section 3.3.6), may be transferred back to a variable by means of procedure *inreal*, provided that an appropriate manipulation of the foreign medium has also been performed.

Procedures *inarray* and *outarray* also form a pair; they transfer the ordered set of numbers forming the value of the array given as the second parameter, the array bounds being defined by the corresponding array declaration rather than by additional parameters (the mechanism for doing that is already available in ALGOL 60 for the value call of arrays). The order in which the elements of the array are transferred corresponds to the lexicographic order of the values of the subscripts, i.e.

$$a[k_1, k_2, \ldots, k_m] \quad \text{precedes} \quad a[j_1, j_2, \ldots, j_m]$$
$$\text{provided} \quad \left.\begin{array}{ll} k_i = j_i & (i = 1, 2, \ldots, p-1) \\ k_p < j_p & \end{array}\right\} \quad (1 \leqq p \leqq m) \tag{2}$$

It should be recognized that the possibly multidimensional structure of the array is not reflected in the corresponding numbers on the foreign medium, where they appear only as a linear sequence as defined by (2).

The representation of the numbers on the foreign medium conforms to the same rules as given for *inreal* and *outreal*; in fact it is possible for example to input numbers by *inreal* which before have been output by *outarray*.

4. Examples

```
procedure outboolean (channel, boolean); value boolean; integer channel;
    Boolean boolean; comment this procedure outputs a Boolean value as a
    basic symbol true of false;
    if boolean then outsymbol (channel, 'true', 1)
               else outsymbol (channel, 'false', 1)

procedure outstring (channel, string); value channel; integer channel;
    string string; comment outputs the string string to the foreign medium;
    begin integer i;
          for i := 1 step 1 until length (string) do outsymbol (channel, string, i)
    end

procedure ininteger (channel, integer); value channel; integer channel, integer;
    comment inputs an integer which on the foreign medium appears as a se-
    quence of digits, possibly preceded by a sign, and followed by a comma.
    Any other symbol in front of the sign is discarded;
    begin integer n, k; Boolean b;
          integer := 0; b := true;
          for k := 1, k+1 while n = 0 do insymbol (channel, '0123456789−+', n);
          if n = 11 then b := false; if n > 10 then n := 1;
          for k := 1, k+1 while n ≠ 13 do
          begin integer := 10 × integer + n − 1;
                insymbol (channel, '0123456789−+,', n)
          end 1;
          if ¬ b then integer := − integer
    end
```

```
begin
      begin array a[1 : 10];
            ⟨statements⟩;
            outarray(15, a)
      end;
      begin array b[0:1, 1 : 5];
            inarray(15, b);
            ⟨statements⟩
      end
end
```

The following example exhibits the use of inarray and outarray for inversion of a matrix including transfer of the matrix elements from and to the foreign medium. It requires that an appropriate declaration for a matrix inversion procedure as well as the declaration of outstring as given above are inserted at appropriate places in the program.

```
begin integer n;
      inreal(5, n); comment the matrix elements must be preceded by the order;
      begin array a[1 : n, 1 : n];
            inarray(5, a);
            matrix inversion(n, a, singular);
            outarray(15, a);
            goto ex
      end;
singular: outstring(15, 'singular');
ex: end
```

5. Concluding remarks

WG 2.1 does not propose any further means for input-output operations, but would like to draw attention to

"A proposal for input-output conventions in ALGOL 60", by the ACM programming languages committee (Subsommittee on ALGOL, D. E. KNUTH, chairman), and to the extensive list of references at the end of that report.

This report has been reviewed by IFIP/TC2 on Programming Languages in May 1964 and has been approved by the Council of the International Federation for Information Processing. Reproduction of this report for any purpose is explicitly permitted only in full.

International Federation for Information Processing
Working Group 2.1 on ALGOL
Chairman: W. L. VAN DER POEL
Technological University Delft
Delft, Netherlands