STICHTING
MATHEMATISCH CENTRUM
2e BOERHAAVESTRAAT 49
AMSTERDAM

DR 35

An ALGOL Publication Form.

A.J. Perlis
and
B.A. Galler

1966

An ALGOL Publication Form

by

A.J. Perlis and B.A. Galler

In this note we suggest a convention for the publication of ALGOL 60 algorithms. The main concern here is for the occurrence and degree of indentation in a published algorithm. While this may seem to some unworthy of serious attention, there are several reasons for establishing a convention:

(1) If each indentation occurs as a result of a specific construction in the algorithm, and especially if it represents a change in level of control (or scope), then the author has a check on his intentions while he writes the program, and the reader is guided into correct interpretations of control sequencing.

(2) Once a convention exists, editing programs can easily provide the service of reconstructing the appropriate indentations whenever changes are made in the program which affect the level of control (providing a powerful error checking facility as a by-product).

(3) The advantages to publishers and proofreaders are obvious.

The goals of an indentation convention must be:

(1) Change of level of indentation should coincide with important and recognizable constructions in the algorithm.

(2) Needless indentation, resulting in crowding of the algorithm to the right and/or elongation, should be avoided.

(3) The rules of the convention should be simply stated and easy to carry out.

The proposed convention assumes that each publisher will define for himself a standard amount of indentation depending on the style and size of type, column width, etc., to be called here the indentation unit. We shall also refer to the left-most column in which characters may be

printed as the (current) indentation origin. (This is initially the left margin.) The column one indentation unit to the right of the indentation origin is the secondary origin. Certain constructions in ALGOL 60 will move the origin to the right, others will move it to the left; unless otherwise indicated, the change is always one indentation unit. Establishing an origin always causes a new line to begin at the new origin and causes subsequent lines to originate at the corresponding secondary origin.

In the rules which follow for establishing the position of the origin, else if and else for (with or without labels immediately after the else) are regarded as occurrences of else. The indentation rules then are:

(a) When entering the scope of a for clause or an if clause in a conditional statement (other than in an else combination as described above), establish the indentation origin one unit farther to the right.

(b) Each else establishes a new origin position without moving it right or left; i.e., it is printed at the same origin as its initiating if, and subsequent lines have the usual secondary origin one unit to the right.

(c) Labels, excepting when occurring immediately after an else, always start a new line at the current secondary origin.

(d) The declarators for a block establish a new origin one unit to the right of the current secondary origin.

(e) The semi-colon terminating the scope of a then, else, do or a procedure body establishes a new origin (after it is printed) which coincides with the previous origin.

(f) The termination of a block establishes the origin one unit to the left of the previous origin.

(g) Comment suspends the observance of these conventions. The terminating semi-colon reinstates the origin as if the comment had not been present.

As an example of the use of this convention and the advantages incurred, a typical algorithm from the Commun. of the A.C.M. has been reproduced in Figure 1 with everything except for the comment, not relevant to control suppressed, the length of the line indicates the amount of space devoted to a suppressed piece of text. The same algorithm appears in Figure 2, using the convention given above. Not only is the number of lines of print reduced, but the flow of control is immediately clear.

We now observe that there is enough information contained in the indentation that all occurrences of begin and end may be suppressed. This remark means that if one starts with an ordinary ALGOL 60 algorithm in indented form (i.e., indented according to the above conventions), and if one suppressed occurrences of begin and end, then an algorithm can be given which introduces the symbols begin and end to produce a representation of the original algorithm,"equivalent to" the original indented representation. (In fact it must differ at most in the elimination of redundant occurrences of begin and end.). This re-introduction algorithm may be stated as follows (assuming a correctly indented representation):

(i) : The following rules do not apply to any text appearing between and including the symbol comment, and the following semi-colon.

(ii) : Insert a begin at the beginning of the algorithm.

(iii) : Reading from left to right, insert a begin after each occurrence of a conditional statement then or a do, and after each else not followed immediately (except possibly for a label) by an if or for. Also insert a begin to the left of any declarator which starts a doubly indented line.

(iv) : Whenever a line of print (including the last) is not followed by a line of print which starts farther to the right than the current indentation origin, append n occurrences of the symbol end at its right end (but in front of any final semicolon that might be there), where n is the difference between the number of occurrences of begin already introduced for origins to the right of the next line's indentation origin and the number of occurrences of end already introduced for origins to the right of the next line's indentation origin.

```
real procedure ————————————; value ————————————; integer ————————————;
     real ——;
comment ————————————————————————————————————————————————————————————;
begin integer ————————; real ——;
     ————; ————;
A: if ———————— then begin ————————————————————————————————; go to
   E end else
   if ———————— then begin ————————; ————————————; go to A end else
   if ———————— then begin ————————; ————————————; go to A end else
   begin real ————————;
        ————;
U:      if ————————————————————————————————————————————————————
           ———————————————————————————— then begin ————————————;
           go to U end;
        ————————————————————————————————————————————————;
        ————————; ————;
        for ——————————————————————— do
        ---
        begin
             ————————————————————————; ————————————————————————;
             ————————————————————————————————————————————————
        end
   end
   ————————————————————————————————————————————————————————————————
        ————————————————————————;
E:
end ————————————————————
```

Figure 1. Algorithm # 225, Comm. of A.C.M.
p 295, May, 1964.

```
real procedure ————————————; value ————————————; integer ————————————;
     real ————————;
comment ————————————————————————————————————————————————————————————;
          begin integer ————; real ————————————; ————————; ————————————;
A:   if ———— then begin ————————————————————; go to E end
else if ———— then begin ————————; ————————; go to A end
else if ———— then begin ————————; ————————; go to A end
else
     begin real ————; ————————;
        U: if ————————————————————————————————————————————————————.
                  ———————————— then begin ————————————————————————;
                  go to U end;
           ————————————————————————————————————————————————————————.
           for ———————————————— do begin ————————————————————————;
                  ————————————————————; end end;
————————————————————————————————————————————————————————————————————
————————————————————————————————————————————————————————————————————;
E: end
```

Figure 2.

(v) : After step (iv) is completed, delete every pair (begin, end) which bounds a single block (already including its own begin and end, an isolated declaration (including a procedure declaration standing alone or a basic statement. (We ignore here text which may follow an end). Figure 3 shows the algorithm of Figure 2 with the symbols begin and end suppressed.

The authors recommend that the Algorithm Section of the Communications and other interested publishers adopt this version of the publication language for ALGOL 60. It is clearly within the intent and letter of the published ALGOL 60 specifications that the publication

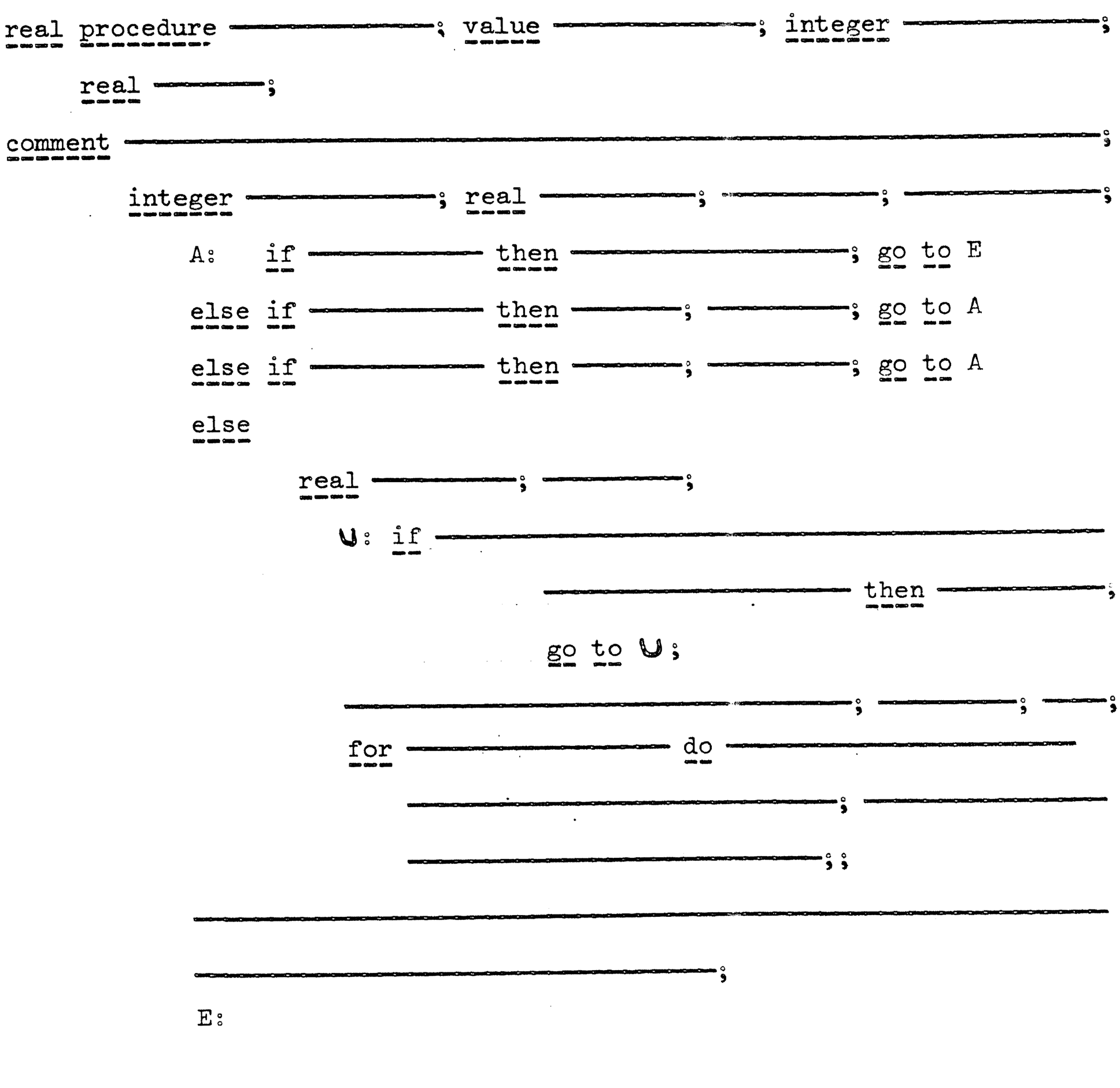

Figure 3.

language be as readable as possible, while still allowing a univocal mapping of programs and algorithms into the reference language. Since the preceding algorithm produces an equivalent, correct ALGOL 60 from an indented representation, this recommendation is in full accord with this policy.