Commentaries on the Revised Report

The following commentaries are issued by the Sub-committee on ALGOL 68 Support, a standing sub-committee of IFIP WG 2.1. They deal with problems which have been raised in connection with the Revised Report on the Algorithmic Language ALGOL 68, and mostly take the form of advice to implementers as to what action they should take in connection with those problems. These commentaries are not to be construed as modifications to the text of the Revised Report.

Note that commentaries are not being published on trivial misprints. Those concerned about such misprints (and especially those preparing new printings of the Report) should apply to the Editor of the ALGOL Bulletin for the latest list of agreed Errata.

{{Commentaries 1 through 30 have already been published (see AB42.3.1 and AB 43.3.1). The two new commentaries published here were accepted by the Support Sub-committee at its meeting in April 1979.}}

31) Overwriting of existing books and control of the write mood.

The Report provides that, where both "put" and "get" are "possible", an existing book with sequential access may be read up to some arbitrary point and overwritten with new information from there onwards. If a given implementation can only overwrite from the beginning of a line, or even from the start of the book, it should be arranged that "put possible" (which is a procedure) only returns TRUE when the current position is at a place from which overwriting may commence.

However, even if "get" is not "possible" (but "put" is), the Report permits such an arbitrary point to be reached by suitable calls of "space", "newline" and "newpage". However, these calls can only be implemented by reading the book from the beginning, counting characters and line and page terminators, and this is impossible by hypothesis. It is indeed strange that even "put(f, newline)" causes the book to be read in order to skip a line. It is even stranger that there is no way in which the first line of an existing book can be overwritten with an empty line. These difficulties all stem from the fact that it is the putting of an actual character which causes the logical end of the file to be retracted to the current position \{10.3.3.1.b\}. Implementers are therefore advised to test in "set write mood" \{(10.3.1.4.j)\} for the case where the logical file end is beyond the current line in a sequential access book, and to retract the logical file end there rather than in "put char". Moreover, it should now be the case that all calls of "put" and "putf", even "put(f, ())", should set the write mood and bring about this effect. To this end, implementers should always call "set write mood" at the start of "put" \{(10.3.3.1.a)\} and of "putf" \{(10.3.5.1.a)\} \{just after the tests for "opened" which are currently provided\}. Very few programs will be changed in meaning as a result of this and, moreover, the precise effect defined by the Report can always be obtained by writing "get(f, newline)" (in situations where "get" is "possible", of course).

32) On the scope of the particular-program.

According to the letter of the Revised Report the first environ created during the elaboration of the ENCLOSED-clause of the particular-program is newer in scope than the environ of the user-task in which it is contained. This would imply that the heap scope (see also Commentary 3) is newer than the scope of the variables (in particular "stand in", "stand out" and "stand back") declared in the particular-prelude. As a consequence, the elaboration
of, e.g., the call "open(stand in, ",", stand in channel)" in the particular-prelude would result in scope violation and thus be undefined. This is, however, not the intention. In effect, the environ in question should be considered nonlocal, so that the scopes concerned are the same. Also, the meaning of the following particular-program should be well defined:

```
BEGIN on logical file end(stand in, (REF FILE f)BOOL: GOTO lfe);
  DO STRING s; read((s, newline)); print((s, newline)) OD;
  lfe: print("###eof###")
END
```