BASIC ALGOL

a close-up of some clear-cut advantages

by DANIEL D. McCRACKEN, McCracken Assoc., Ossining, N.Y.

"ALGOL will collapse of its own weight in a year or two. It's nothing but an intellectual exercise for recursive procedure theorists." So said an acquaintance a while ago, and I suspect he speaks for a great many computing people.

I think it's time somebody spoke up for the power of ALGOL in doing "ordinary" programming—the kind of work in which recursive definition, "own" variables, and call-by-name seldom arise. The value of such "advanced" features of ALGOL should not be minimized, but neither should it be thought that until one understands these aspects thoroughly (a supposedly hopeless task), no use can be made of the language. In refutation of this position, let me list some obvious points of superiority of ALGOL over, say, FORTRAN, in doing *simple* calculations.

(FORTRAN is chosen for comparison simply because it is the most widely used scientific programming language, and thus provides a good frame of reference.)

1. Variable naming is easier: you don't have to worry about the IJKLMN business or the restriction to six characters.

2. Constants are simpler to write: no more recompilations because you forgot to write a floating point 2 with a decimal point.

3. Conditional expressions are much more powerful. Instead of the extremely restricted format of the FORTRAN IF statement, you can write any Boolean expression, and there are other flexibilities available if you want to use them. It takes no great effort to devise realistic examples of tests that take a half-dozen statements in FORTRAN and one in ALGOL.

4. ALGOL offers great power and flexibility in writing loops, at no great penalty in difficulty of learning. If you want to execute a loop for several values of a parameter, you can simply list them. In the form of the ALGOL "for" statement that most closely corresponds to the FORTRAN DO statement, the three parameters can be any expression, rather than "a single unsigned fixed point constant or variable." This one thing alone would take dozens of statements out of some FORTRAN programs. Another variation allows continued execution of a loop as long as a Boolean expression is still true.

5. Subscripting capabilities in ALGOL are more flexible and powerful. FORTRAN, for instance, will accept the subscript 2*N − 3, but reject N*2 − 3. ALGOL has no such annoyances; a subscript can be any expression, and that's that. Furthermore, the lower bound of a subscript is not restricted to the integer *one*, and array sizes can be established at execution time.

6. Compilers will be faster, I am told. The speed of the Burroughs Algebraic Compiler for the 220 and of MAD, both of which are more like ALGOL than FORTRAN, would seem to bear out the contention.

Other points could be listed, but these should demonstrate that there is a lot to be said for ALGOL besides the power of the advanced features. It may also be noted that this list contains nothing that is particularly difficult to learn. In fact, learning is easier in many particulars, because you don't have to keep track of a lot of bothersome little restrictions.

It is interesting to speculate on the origin of the myth of ALGOL's abstruseness, for which I suggest three reasons. First, the report in the ACM *Communications* of May, 1960, is excellent for its intended purpose of defining the language; but somewhat lacking when viewed as a beginner's primer. I assume I am not the only one who spent twenty minutes on the report and promptly gave up on ALGOL because I thought it would be more trouble to learn than it was worth. (I assume I am also not the only one who found that the report is not all that difficult, once I took the trouble to learn the notation.)

Second, most of the published discussion of ALGOL has centered around the advanced features, which is entirely reasonable, but misleading. It is only natural that those who are concerned with the development of advanced computer languages should spend their time on the difficult things, after noting that the "simple" advantages of the type listed above would be relatively easy to accomplish. But this sort of discussion leaves those of us on the fringes with the entirely mistaken impression that ALGOL consists only of the difficult things.

Third, the algorithms published in the *Communications* are slow going for some of us *because the problems they solve* are slow going for some of us. I for one don't know off-hand how a Riccati-Bessel function differs from a plain Bessel function; of course I'm not going to get much out of a casual reading of an algorithm to compute a Riccati-Bessel function—no matter *what* the language. Once again, this is proper, but misleading. There is not too much point in publishing algorithms to do "ordinary" things, which can obviously be done fairly easily in any language. But in the process of exhibiting how ALGOL can be used for difficult problems, some of us got the impression that that was the whole story.

In summary, it appears to me that ALGOL offers clear-cut advantages to anyone doing scientific computing, whether or not the application requires use of the more advanced features of the language. These features may very well turn out to be major advances in the computing art; in the meantime, there is no need to wait for the dust to settle before making use of the "simple" advantages.

It's time for some of us to take a fresh look. ■