Implementation of ALGOL 60 for the English Electric KDF9

By F. G. Duncan

Our decision to implement ALGOL 60 for the KDF9 was taken about 18 months ago. We saw in ALGOL 60 the possibility of its use as an automatic programming language for a wide range of users, particularly those who had become accustomed to working with earlier languages. On the one hand, ALGOL 60 was quite obviously much more powerful than any of the others, and on the other hand we wanted to make a contribution towards the effort to establish a language that could be universally understood and implemented. From the first our ambition was to implement the ALGOL 60 Report (Naur, ed., 1960) as fully as we possibly could. Experience in writing and using an "optimizing" compiler for DEUCE (Duncan and Hawkins, 1959; Duncan and Huxtable, 1960) led to the desire to combine full ALGOL translation with the generation of "efficient" (or relatively fast-operating) object programs. We realized, of course, that this would entail considerable original work, and planned to put about five man-years into the project.

This was at Kidsgrove, in the Data Processing and Control Systems Division. There was also an interest in ALGOL in the Atomic Power Division at Whetstone, where, in fact, there were people who had produced real, working, compilers. They were prepared to put in one man-year or so on an independent ALGOL compiler, to be made available before the more elaborate Kidsgrove version. This would both help us to get ALGOL established among users, and provide checks on the development of the elaborate scheme. At first we saw it as a very restricted ALGOL compiler; but following some informal contacts with Professor van Wijngaarden and Dr. Dijkstra of Amsterdam, whose compiler (Dijkstra, 1961)* had by this time been completed, the Whetstone people were entertained magnificently at Amsterdam for a week, and returned able to remove most of the restrictions they had previously imposed.

This occasion is a suitable one for us to express publicly our thanks to the Mathematisch Centrum, Amsterdam, for their great generosity in making sure that their work was fully understood by us; and also to Dr. Naur's group at Copenhagen, with whom I spent a very useful week rather more than a year ago. Our relations with both these groups have always been very friendly.

Towards the end of last year, therefore, we found ourselves with the possibility of having two compilers, both able to deal with almost the whole generality of the ALGOL Report. Naturally the question arose as to whether one or other of these projects should be dropped. We decided to continue with both. Our earlier experience and our discussions with a variety of prospective users indicated that there would be a place for each of two compilers with complementary characteristics.

* See also the paper on p. 125 of this issue.—Ed.

Compiler Characteristics

These characteristics, briefly, are as follows:

Whetstone. The aim is fast compiling. The operation is what is sometimes called "one-pass-load-and-go." There is no particular attempt to obtain efficiency in the object program. As in the case of the Elliott compiler described by Mr. Hoare,* compilation time is much the same as paper-tape reading time. It is thought that there will be about 3,000 words of instructions in the compiler, and we hope that it will be available during the autumn of 1962.

Kidsgrove. Compiling takes longer, but the time should not be more than three minutes. There are several passes through the program, aimed at recognizing and giving special treatment to certain situations which are amenable to "optimization." Examples of the situations are simple "for" statements making simple use of the controlled variable, and exceptionally well-behaved procedure declarations and their corresponding calls. Situations which cannot clearly be recognized as qualifying for optimization are given the full general treatment—a "fail safe" method. An account of the detection processes is given by Hawkins and Huxtable (1962).

Now for the properties which the two compilers have in common. First, both accept identical versions of ALGOL 60. The restrictions are:

1. No integer labels.
2. A complete specification part must be provided with each procedure declaration.
3. The declarator **own** is given the sense of Reformulation 23 of *ALGOL Bulletin* 14, except that we do not allow **own** arrays with "dynamic" bounds.
4. Where a formal parameter is specified as a procedure, all the corresponding actual procedures must have identical specifications. Second, both will be able to accept the same paper-tape versions of the ALGOL texts. Third, both will be able to deal with the same procedures whose bodies are in KDF9 User Code. This is an assembly code for the machine whose instructions are in one-to-one correspondence with those of the computer itself. This point will be expanded later.

Thus there should be complete two-way compatibility. Here one might mention an experimental compiler for

* See p. 127.

```
integer k, p; real y, a, b;
KDF9  4, 0, 0, 2;
      ZERO; DUP; DUP; =V1; =V2;
   2; SET 1; +; ="p";
      "p"; "k"; −; J1 > Z;
      V2; V1; "a"; "b"; ×+F; =V1; =V2;
      "p"; J2;
   1; V2; V1; ROUND F; ="y";
ALGOL
```

The formal parameters, which are each enclosed in quotation marks in the KDF9 text, are replaced automatically by sequences of instructions for getting access to the required quantities. The system allows calls by name and by value. In the example, V1 and V2 correspond to the local variable *s* of the Report version. In this version the sum is accumulated double length and then rounded.

The reason for providing so fully for machine-code procedures is to simplify the introduction of features which it would otherwise be inconvenient or impossible to express within ALGOL. For example, one might need a set of procedures for evaluating some frequently-used functions, and the running speed of the ALGOL versions, even when translated by an optimizing compiler, might not be sufficiently close to that of the corresponding machine-code versions. Input and output, and magnetic-tape procedures must either themselves be in machine code or make use of machine-code procedures. Something like the scheme we have proposed is necessary if one wishes to get beyond the stage of "read one number, punch one number."

As was mentioned earlier, machine-code procedures and ALGOL procedures can be included in the library. It follows that the user of library procedures for input and output need know nothing about the KDF9 User Code.

A matrix scheme proposed for use with our ALGOL system (Denison, 1962) makes use of a number of procedures which have already been expressed in ALGOL. It is probable that they will be rewritten in User Code for the sake of speed.

Acknowledgement

Acknowledgement is due to those colleagues whose work is described in these notes. Publication is by permission of The English Electric Company Limited.

References

Denison, S. J. M. (1962). "A Proposed ALGOL 60 Matrix Scheme." Paper to be presented to the IFIP Congress 62.
Dijkstra, E. W. (1961). "ALGOL 60 Translation," *ALGOL Bulletin*, Supplement No. 10, Mathematisch Centrum, Amsterdam.
Duncan, F. G., and Hawkins, E. N. (1959). "Pseudo-Code Translation on Multi-level Storage Machines," *Proceedings of ICIP, Paris*, p. 144.
Duncan, F. G., and Huxtable, D. H. R. (1960). "The DEUCE Alphacode Translator," *The Computer Journal*, Vol. 3, p. 98.
The English Electric Co. Ltd. (1961). *KDF9 Programming Manual.*
Green, J. S. (1961). *Introduction to ALGOL 60 Programming for the KDF9*, The English Electric Co. Ltd.
Hawkins, E. N., and Huxtable, D. H. R. (1962). "A Multi-pass Translation Scheme for ALGOL 60," *Annual Review Automatic Programming*, Vol. 3 (to be published).
Naur, P. , ed. (1960). *Report on the Algorithmic Language ALGOL 60*, Regnecentralen, Copenhagen.
Naur, P., ed. (1962). "*ALGOL Bulletin*, No. 14," Regnecentralen, Copenhagen.
Randell, B., and Russell, L. J. (1961 and 1962). Descriptions of work for DEUCE and KDF9 in internal memoranda of the Atomic Power Division, The English Electric Co. Ltd.

Operating experience with FORTRAN

By A. E. Glennie

My purpose in this talk will be to describe the lessons that I have learned from my own experience, and that of my colleagues, in using FORTRAN during the last three years. Some of the points I shall make are specifically about the FORTRAN language itself; others are about automatic coding in general, and computer operating systems incorporating compilers. I hope that what I have to say about the latter aspects, as revealed in the use of FORTRAN, may be of interest and value to those of you whose interests and preferences may be in other languages—or in fields other than scientific computation.

The History of Our Use of FORTRAN

I think that you will be interested to hear how the use of FORTRAN in the U.K.A.E.A. developed, as our story is quite typical of the evolution of a laboratory's technique and practice. When, in 1958, we started our first experiments with FORTRAN on the IBM 704, we had had a long tradition of machine-language coding, but had also had some experience of automatic coding.

We were not, I think, prejudiced against automatic coding, yet we were somewhat disappointed with our first experience with FORTRAN. This was the