ALGOL 68 AND STRUCTURED PROGRAMMING FOR LEARNER-PROGRAMMERS

Bryan Ratcliff, University of Aston in Birmingham

Abstract

A method is described of introducing to absolute beginners basic concepts of structured programming, including constructing programs by "step-wise refinement". This is interleaved with a "top-down" description of a simple mini-subset of ALGOL 68, which is closely followed by the concepts of data structuring and procedurisation, as expressed in this language. Emphasis throughout is on the programming philosophy behind the approach controlling the teaching of ALGOL 68, rather than vice versa.

1. Introduction

The philosophy behind the teaching method to be outlined is founded upon the following set of simple beliefs:-

(i) ALGOL 68 is a good programming language

(ii) it is no more difficult to teach ALGOL 68 to beginners (up to a certain level and given enough time) than any other programming language

(iii) it is only slightly more difficult to teach "top-down, structured" programming to beginners than no programming methodology at all

(iv) it is possible to combine (ii) and (iii).

As to the merits of ALGOL 68, either in general or as a suitable language for structured programming in particular, this has long been debated (see [1] for example); suffice it to say that, at this level of teaching, discussion of features such as a virtually non-existent escape control structure and unimaginative loop control structure (no repeat ..... until ..... etc.) becomes (literally) academic; if one is content with a series, a generalised conditional and a while loop, then ALGOL 68 proves adequate to the task. References to both ALGOL 68 and structured programming are well-known and in the main need no repeating here. However, for the sake of completeness, [2] remains a classic, [3] is typical of such a distinguished author, while [4] is representative of more recent and radical outlooks. [5] is by one of the leading proponents of programming by "step-wise refinement", and [6] provides a pragmatic all-round account of modern theories of program design and related topics. For theories and discussion of techniques of teaching programming, one could do no better than consult [7]. Finally, [8] tends to sparkle by default amongst an otherwise sadly sparse array of current ALGOL 68 primer literature; for a more "practical" approach, try [9].

2. The method

A survey of the manner in which FORTRAN, COBOL, ALGOL 60, PL/1 etc. are introduced to learner-programmers would surely reveal that the following order of "elaboration" of a language's constructs predominates: numeric "constants", integer and real variables (and their declarations if appropriate), simple formulas involving basic numerical operators, pre-defined functions, assignment, simple input and output, trivial "straight-line" programs (such as computing the area of a triangle or the roots of a quadratic equation - this truly is formula translation!), boolean expressions and conditional statements, loops, arrays and programmer-defined procedures. There would of course be variations depending upon the actual language involved, teacher preferences and so on, but the overall tendency towards a bottom-up approach would be obvious.

It may be argued, however, that if one's initial overriding concern is to develop within students a feeling for "structure" and "top-downness", then this will be achieved with far greater effect by largely reversing the traditional order, particularly with a language as structurally expressive as ALGOL 68. This is the basis of the method which is employed. Currently, topics are introduced in the following order:-

(i) series, **if**-**then**-**else**-**fi**, **while**-**do**-**od**

(ii) **int** variable declarations, assignment, **int** denotations and formulas (+, −, *, ÷ and ↑ only), simple transput (including **string** denotations with "print") and **bool** formulas (=, ≠ etc. plus **and**, **or**, **not**)

(iii) modes **real**, **char** and a few selected operators; correspondence between applied and defining occurrences of identifiers ("scope")

(iv) data structuring - one and two-dimensional arrays of **int**, **real**, **bool** and **char**, mode **string**, structures

(v) procedurisation

Each of these steps represents a major advance in the students' experience; a few important points need to be made:

Step (i): Effectively, the students are taught that a **bool** series can be written between **while** and **do**; this is to avoid awkward and inelegant constructions of the form: S_1; **while** B **do** S_2; S_1 **od** (there is no analogous problem with the conditional clause). Students tend to have far more initial difficulty with the loop than the conditional clause, and there are two main reasons for this: firstly, the syntax of the former does not directly reflect its elaboration whereas with the latter it does (it reads "naturally"), and secondly, it is often not obvious how to express oneself within the confines of the **while** form. Program schemata help here, and an important schema students meet at an early stage is:-

```
set v to initial value;
while v ≤ (or ≥) final value
do S; increment (or decrement) v by step value od
```

(note how, at this stage, informal computer "Esperanto" is used for all

other programming notation required)

Step (ii): Low-level details now emerge. An abstract reference machine whose storage model is one of a set of blackboards helps to give a vivid interpretation of the most important concept of this step - that of a variable (casually introduced in Step (i)) and the associated actions of assignment and dereferencing. On completion of Step (ii), students are introduced explicitly to the technique of top-down problem-solving and coding; with the mini-subset of ALGOL 68 now within their grasp, they are able to tackle many non-trivial and interesting algorithms as their first practical programming exercises.

Step (iii): Modes real and char widen the basis for programming examples. "Scope of identifiers" is always a difficult topic but it is best to explain it now before discussing procedures, and before the need to explain it has arisen anyway, because some student's program has mysteriously (to the student) failed to compile.

Step (iv): Slices are a somewhat tiresome subject to teach; to avoid getting bogged down in syntactic/semantic minutiae, simplification is achieved by always using trimmers with explicit lower and upper bounds. Mode string is taught as an entirely separate data type (flex is never mentioned for everyone's safety and peace of mind).

Step (v): To ease the technicalities of the material involved, introducing procedures in the following order seems best:

(a) parameterless void procedures
(b) parameterless non-void procedures
(c) procedures with parameters

At point (b), the idea of a closed clause yielding a value can easily be conveyed as a "write-the-expression-last-in-the-series" rule, since goto and exit are never discussed. At point (c), ref and identity declarations appear for the first time in order to explain properly parameter correspondence.

3. Concluding remarks

The main features of the way in which ALGOL 68 is taught can be summarised as follows:-

(i) By starting with a top-down development of the simple mini-subset described, the essence of structured programming embodied in the three basic types of control mechanism is emphasised from the outset; moreover, "top-downness" in teaching encourages "top-downness" in program construction.

(ii) In effect, point (i) illustrates the general principle that the programming methodology being taught should control the order of introduction of the language's constructs and concepts, rather than vice versa; pleasant but initially unnecessary constructions such as assigning operators, initialised declarations, brief symbol forms, if clauses and assignations as expressions, case clauses etc. are left until the end of the course.

(iii) Constructs and concepts are introduced only at the point where they are deemed to be first needed e.g. identity declarations and ref are not mentioned until procedures with parameters are discussed.

(iv) A "theological" approach is completely shunned; to confront a beginner with ref int i = loc int (rather than an int blackboard named i!) will serve only to confuse rather than enlighten.

(v) Simplicity is of prime importance (as point (iv) implies); experience has shown that the subset of ALGOL 68 described in steps (i) - (v) inclusive of section (2) can be taught without resort to most of the more difficult concepts and terminology embodied in ALGOL 68 (particularly as regards coercions and context strengths).

It might be argued that, since structured programming is initially intellectually more demanding than a conventional approach, it will serve to discourage certain students from persevering with learning to program. However, there is always a small proportion of any class who are apparently incapable of expressing themselves in a "programmatic" way, whatever the language or methodology being taught. Since habits, whenever acquired, die very hard indeed, it is vital for those people who do not belong to the afore-mentioned group, that the first habits they acquire are good ones. The more traditional bottom-up teaching methods must be questioned as to their effectiveness in achieving this objective. Certainly, the initial results obtained from the teaching method described are extremely encouraging, with students constructing highly structured programs. The main area of difficulty lies in developing the problem-solving capabilities of students and the systematic application of these in top-down design; it is envisaged that the course will undergo further modification in an attempt to improve its effectiveness in this crucial area.

References

[1] Sintzoff M., "A brief review of Algol 68", Algol Bulletin, No. 37, July 1974.

[2] Dahl O.J., E.W. Dijkstra & C.A.R. Hoare, Structured Programming, Academic Press, 1972.

[3] Knuth D.E., "Structured Programming with Go-To Statements", ACM Computing Surveys, Dec. 1974.

[4] Weinberg G.M., D.P. Geller & T.W-S Plum, "IF-THEN-ELSE considered harmful", SIGPLAN Notices, Vol.10, No.8, Aug.1975.

[5] Wirth N, Systematic Programming, Prentice-Hall, 1973.

[6] Yourdon E., Techniques of Program Structure and Design, Prentice-Hall, 1975.

[7] Turski W.M. ed., Programming Teaching Techniques, North-Holland, 1973

[8] Pagan F.G., A Practical Guide to Algol 68, Wiley, 1976.

[9] Learner A & A.J. Powell, An Introduction to Algol 68 Through Problems, Macmillan, 1974.