Structured programming, programming teaching and the language Pascal

Olivier Lecarme
Département d'Informatique
Université de Montréal

1 - Introduction

This paper is a series of considerations on the three subjects stated in its title, and on their mutual relationships. Since it is a survey of the present situation in three related and evolving areas, references play a very important role. A rather extensive bibliography is thus appended to this text, and more than doubles its length.

The structure of this paper may be very simply inferred from its title: the three subjects a, b, c are examined first separately, then two by two and finally all three at once, according to the scheme:

a - b - c - ab - ac - bc - abc

2 - Structured programming

Even if we give to this somewhat fuzzy term a wider signification than is customary, this subject was still unknown a few years ago, and has today a crucial importance. Is it only a fashionable topic, or does it correspond to a real need? If a need, it is a very recent one, and some meritorious books on programming risk to be rapidly obsolete because of its importance. Even Knuth's magnum opus [Knu68,Knu69,Knu73a] is not completely irreproachable from this aspect.

To see how much mentalities have changed, it is sufficient to browse among the Collected algorithms of the A.C.M. or the algorithms published by the Computer Journal: for example, see [Flo62] (which is a pathological case), [Sin68] or even more recently [Woo70]. Books also give instructive examples, as in [Har71] or [Dav73]. Weinberg's books [Wei70,Wei71] present symptoms of a change, but still rather slight.

The oldest papers are concerned by proofs of algorithms and axiomatisation of programming; it is only recently that these matters have been related to structured programming [Nau66,Flo67,MCP67,Bur69,Hoa69,Ars71,FoH71, ClH72,Gri72,Lon72,Hoa72b,Hoa72c,Hoa73a,MNV73]. The goto controversy, because of its theoretical and paradoxical aspects, produced many papers following Dijkstra's celebrated letter to the editor of the Communications [Dij68a,Ric68,AsM71, Wul71,KnF71,Lea72,Hop72,Wul72,Ars72,Boc73,NaS73, PKT73]. The necessity for structured programming and its practical usage are advocated in more and more papers [Dij68b,Dij69,Nau69,Bau71,Mil71,Woo71, Wir71a,Dij72a,Hes72,BrH72,Hoa72a,Hoa72d,Nau72, Wir72c,DaH72,Bak72,Dij72b,Bro73,ClH73,Lec73,Sha73, ShW73]. The first books have at last just been published [DDH72,Wir73,WMY73,ChC73]. In every meeting related to programming, these different aspects take great importance, even if they were not planned in the programme [ACM71,IFIP72,ACM73]. However, some of the most fundamental ideas are far from being new, since they were exposed by Polyà [Pol48], who found them (in part) in the works of Bolzano, Descartes and even Pappus of Alexandria.

The major ideas that we group under the heading of structured programming comprise: complete or partial banishment of the goto statement, by the way of logical constructs with nested structure; a novel approach to modularity; construction of programs by stepwise refinement; top-down programming; analytic verification or proof of correctness of algorithms; and the use in program construction of a strict but freely accepted discipline.

3 - Programming teaching

This is an up-to-date subject, but also a controversial one. The different curricula which have been proposed for teaching computer science [ACM68,Baz69,Las72,AuE73] contradict each other, about the whole subject and especially about the teaching of programming. The working conference sponsored by IFIP to examine this one subject [IFIP72] saw the proposition of just about all possible solutions.

The most important reason for differences in opinion is probably the confusion between a programming language and programming (or even computer science). This confusion is maintained by employers, who request that people master the language they will use on their computer, and by the specialized institutes who seriously promise wonderful wages after three weeks of a programming course by correspondence.

Even if people agree really to teach programming (this does not seem evident when we see how much the ACM curriculum for small colleges [AuE73] moves back compared to Curriculum 68 [ACM68]), the quarrel focuses around the precise language used: a pedagogical or a real one, a machine or a high-level one, a powerful or a restrictive one, all possible combinations are proposed, and often well justified [Ada72,Lec72, Org72,Pec72,VdP72,Wei72,Wor72,Hol73]. Even if we decide to use an existing high-level language, the most currently accepted solution, must we choose it because of the number of people using it daily, or independently of this?

4 - The language Pascal

This programming language has a surprising history, which puts it apart from its celebrated predecessors and contemporaries. It is a very recent language [Wir70], officially described in the first number of a new non-American journal [Wir71b], by a quite difficult paper, so condensed that one must find the newest aspects of the language "between the lines" (Haberman [Hab73] makes a spiteful and unfair criticism of Pascal because he was not able to read the report in such a way). It is not the work of a users' group, nor of an international committee, nor of an important computer manufacturer. The first implementation [Wir71c,Wir72b] was done on a costly and not widespread machine, using a method which seems a priori worrying. No useful implementation is presently available on an IBM machine. The language revision, made after two years of use [Wir72a, WiJ73], includes <u>no</u> extension, no new feature, and even suppresses some details, which seems a unique case in the history of programming languages.

All these facts seemed to sentence the language to a definitive obscurity. In fact, it has already a descendance [C1H71,IRH73,Jen73,McK73, Des74], and one says "Pascal-like language" as one said "Algol-like language". Numerous implementations are in progress [Des73,ThM73,Lyn73] or already completed [WeQ72]. During the Sigplan-Sigops interface workshop [ACM73], Pascal was the most quoted language. It is probably the only programming language of comparable power whose formal semantic description uses only twenty-six typed pages [HoW72] (compare to those of PL/1 [IBM69] or even Basic [Lee72]).

Pascal still has some weaknesses, which seem to arise especially from the fact that its author did not want to make its definition grow bigger with ill-formalized features, too machine-dependent or operating system-dependent, or too expensive to implement. The major reason for Pascal's success seems to be the fact that it corresponds exactly to what Dijkstra [Dij72b] requires: "I see a great future for very systematic and very modest programming languages."

5 - Teaching of structured programming

The question is not to teach structured programming as a particular programming technique, alongside other techniques, but to make it the basis of a programming course. The course descriptions which use this notion [Wei72,Ros72, Ros73] are however somewhat fuzzy about this point. They are rather courses "about programming" (as Rosin says), where a series of methods, habits and rules of behavior are taught.

These courses are often offered at an advanced level, i.e. to the few best students, as if one said: "Let us begin by giving people bad habits, then we shall correct them for people who deserve it". However, in an interval of a few months, there have been published the latest development of Dijkstra's celebrated "Notes" [Dij72a], and above all Wirth's book [Wir72c, Wir73], which both suggest making structured programming the basis of programming teaching.

This seems to be evident: if we are convinced that it is not only efficient and useful [Bak72], but even necessary if we do not want to fall into catastrophe [Dij72b], we must teach structured programming to everybody (even to those people who seem to be unrecoverable), immediately, and as soon as at the beginning of programming teaching.

6 - The language Pascal and structured programming

Is it possible to write structured programs (this has not to be confused with programming in a structured way) in any language? The answer is certainly no, and this has already been said several times [WGW71,Pec72,C1H73,Hol73]. However, if we eliminate the machine languages, Cobol and Fortran (there exists however, what is hardly credible, an attempt of extension to Fortran to allow structured programming [Mil73]), the different points of view begin to diverge. Many people indeed are interested only in structuring the program itself, and forget the data. The debate is then centered around that "obscene <u>goto</u>" (Laski in [IFIP72]), that "forbidden fruit" [Wei72], and on the means to prevent the programmer from using it, by completely removing it [C1H71,WRH71] or hiding from him that it exists [Wei72,Hol73]. To replace the <u>goto</u>, the most tricky and unnatural methods are used [C1H71,WRH71,Ars72,IRH73,Boc73], a fact which proves that the true question was left aside, and that people remain influenced by flowcharts [NAS73] and Fortran, since after having ignominously driven away the <u>goto</u> they try to reintroduce it in concealment.

However, the question of data structuring is equally important, and Hoare's papers [Hoa72a, Hoa72c] deal with it in a probably final way. The greatest strength of the language Pascal is precisely to be suggested by [Hoa72a], and to have borrowed from it all that can be realized at a moderate cost (the language LIS alone [IRH73] seems to go a step further in that sense). Besides that, the language offers the statement structures identified as necessary (<u>if</u>...<u>then</u>... <u>else</u>, <u>while</u>...<u>do</u> and the compound statement) or useful (<u>repeat</u>...<u>until</u> and <u>for</u>...<u>do</u>). As for the <u>goto</u>, it is present, even if in an ultra-restrictive form: only this statement (at least presently) allows a clear processing of error situations [Lan66,Lea72,Hop72], without using such powerful and complicated tools as the ones proposed for exit from nested constructs. In fact, the language Pascal, despite its simplicity (or better, because of it), supplies one with all the tools necessary for structured programming, if one is not the slave of flowcharts and not (or no longer) infected by Fortran [Lec73].

7 - The language Pascal and programming teaching

Laski [Las72] gave the best reasons for teaching programming using an algorithmic language, or rather to teach the development of algorithms and their expression by means of a programming language (we paraphrase): "Very few computers are von Neumann machines. Very few programs are written in machine language. A programming language has an a priori meaning; its translation into machine language does not give it an a posteriori meaning, it has only to be correct. Numerical analysis is not computer science. The object of computer science is to represent, store, retrieve, transform and interpret some information."

The advantage of Pascal, in these circumstances, is that it is machine-independent, but not too much: it never refers to the manner in which information is represented in memory, but it allows this to be done reasonably. Besides that, its principal advantage is that it is a simple and concise language, of which the clear and complete description, illustrated by examples, occupies less than a hundred pages. This is the only way for the programming language not to be the subject of the course but only its support, and one has only to browse through some of the innumerable books of introduction to programming, to see that by using Fortran or a subset (even a tiny one) of PL/I, this goal can never be attained.

It is evident that Pascal is not a language for doing everything, but its qualities make it usable in a great variety of problems (and not only in numerical calculations), with fair efficiency, and without frustrating restrictions [Lec72]. As Peck says [Pec72], "it is particularly distressing to think of the vast unfortunate herd of programmers whose only means of communication with the computer is the Fortran language".

8 - Teaching structured programming with the language Pascal

Structured programming must be taught at the time of the initiation to computer science, and the language Pascal is an excellent support for such a course. Is there a better choice? Holt [Hol73] is favourable to Pascal, but cannot use it and deems it to be too young; he is consequently reduced to the "fatal disease" [Dij72b], that is PL/I, which he tries to make innocuous thank to innumerable cuts. Weinberg, on the other hand, preaches the use of PL/I as a universal language [Wei70]. Besides the fact that a universal language is evidently a fruitless lure, it is particularly informing to compare his books [Wei70, WMY73] with Wirth's [Wir72c,Wir73], who evidently advocates the use of Pascal. Whereas the name of the language used as a support in the Wirth's book is quoted once only, in a footnote, a very short syntactic description being put in an appendix, in [WMY73] it is not only the name of the language used which appears in the title, but moreover the name of the precise compiler that must be used! This simple fact seems very significant of the difference of approach between the two books: while one can forget Pascal and concentrate on programming itself, this is completely impossible with PL/I.

Aside from the programming language, the teaching of these two books is easily condensed. Contrary to the opinion of most professional programmers, a good program is the result of 5% inspiration and 95% perspiration. The best means to obtain an error-free program is to manage to not put them into it, since if program testing may be used to show the presence of bugs, it can never prove their absence [Dij69,Dij72b,Wir73]. There is no miracle recipe, and the only thing to do is to work in a systematic manner and to apply the methods identified as good in other areas [Pol48,Nau69,Nau72].

9 - Bibliography

Under the reference abbreviation, we give (when it is available) the reference to the review in Computing Reviews, with the format: (year) volume, number, or to a review paper quoted in this bibliography.

ACM68 — ACM Curriculum Committee on Computer Education. "Curriculum 68. Recommendations for academic programs in computer science". *C. ACM 11*, 3 (1968).

ACM71 — Sigplan symposium on languages for systems implementation, Lafayette, Indiana. Proceedings available in *Sigplan Notices 6*, 9 (1971).

ACM73 — Sigplan-Sigops interface meeting, Savannah, Georgia. Proceedings available in *Sigplan Notices 8*, 9 (1973).

Ada72 — Adams W.S. "The use of APL in teaching programming". in [Tur73].

Ars71 — Arsac J. "Quelques remarques et suggestions sur la justification des algorithmes". *Revue Française d'Informatique et de Recherche Opérationnelle 5*, B-3 (1971).

Ars72 — Arsac J. "Un langage de programmation sans branchements". *RIRO 6*, B-2 (1972).

AsM71 (73)14,24932 — Ashcroft E. & Manna Z. "The translation of *goto* programs to *while* programs". in [Fre72].

AuE73 — Austing R.H. & Engel G.E. "A computer science course program for small colleges". *C. ACM 16*, 3 (1973).

Bak72 (73)14,25440 — Baker F.T. "System quality through structured programming". Proceedings of AFIPS 1972 FJCC.

Bau71 — Bauer F.L. "Software engineering". in [Fre72].

Baz69 — Bazerque G. (ed.) "Programmes d'enseignement de l'informatique". *RIRO 3*, B-2 (1969).

Boc73 — Bochmann G.V. "Multiple exits from a loop without the *goto*". *C. ACM 16*, 7 (1973).

BrH72 — Brinch-Hansen P. "Structured multiprogramming". *C. ACM 15*, 7 (1972).

Bro73
Brooks, N.B.
"Tactics, an integrated system for structured programming".
Sigplan Notices 8, 6 (1973).

Bur69
(69)10,17495
Burstall R.M.
"Proving properties of programs by structural induction".
Computer Journal 12, 1 (1969).

ChC73
Chion J.S. & Cleemann E.F.
"Le langage Algol W. Initiation aux algorithmes".
Presses Universitaires de Grenoble (1973).

ClH71
Clark B.L. & Horning J.J.
"The system language for project SUE".
in [ACM71].

ClH72
Clint M. & Hoare C.A.R.
"Program proving: jumps and functions".
Acta Informatica 1, 3 (1972).

ClH73
Clark B.L. & Horning J.J.
"Reflections on a language designed to write an operating system".
in [ACM73].

DaH72
Dahl O.J. & Hoare C.A.R.
"Hierarchical program structures".
in [DDH72].

Dav73
(73)14,25120
Davidson M.
"PL/I programming with PL/C".
Houghton Miffen Co., Boston (1973).

DDH72
[see Knu73b]
Dahl, O.J., Dijkstra E.W. & Hoare C.A.R.
"Structured programming".
Academic Press, London (1972).

Des73
Desjardins P.
"A Pascal compiler for the Xerox Sigma 6".
Sigplan Notices 8, 6 (1973).

Des74
Desjardins P.
"Dynamic data structure mapping".
Software practice and experience 4, 2 (1974).

Dij68a
[see Ric68]
Dijkstra E.W.
"*Goto* statement considered harmful".
C. ACM 11, 3 (1968).

Dij68b
(69)10,16717
Dijkstra E.W.
"A constructive approach to the problem of program correctness".
BIT 8, 2 (1968).

Dij69
Dijkstra E.W.
"Notes on structured programming".
Technische Hogeschool Eindhoven (1969).

Dij70
Dijkstra E.W.
"Structured programming".
in Buxton & Randell (ed.):
"Software engineering techniques".
NATO Science Committee (1970).

Dij71
Dijkstra E.W.
"A short introduction to the art of programming".
Department of Mathematics EWD316, Technological University, Eindhoven (1971).

Dij72a
Dijkstra E.W.
"Notes on structured programming".
in [DDH72].

Dij72b
(73)14,24552
Dijkstra E.W.
"The humble programmer".
C. ACM 15, 10 (1972).

Flo62
[see Ran62]
Flores I.
"Algorithm 76: Sorting procedures".
C. ACM 5, 1 (1962).

Flo67
Floyd R.W.
"Assigning meanings to programs".
in Schwartz J.T. (ed.): *"Mathematical aspects of computer science".*
American Mathematical Society, Providence (1967).

FoH71
Foley M. & Hoare C.A.R.
"Proof of a recursive program: Quicksort".
Computer Journal 14, 4 (1971).

Fre72
Freiman C.V. (ed.)
"Information Processing 71"
(proceedings of [IFIP71]).
North-Holland, Amsterdam (1972).

Gri72
Gries D.
"Programming by induction".
Information processing letters 1
(1972).

Hab73
Habermann A.N.
"Critical comments on the programming language Pascal" (see [Wir72a]).
Department of Computer Science, Carnegie-Mellon University, Pittsburgh (1973).

Har71
Harrison M.C.
"Data structures and programming".
Courant Institute, New York University, New York (1971).

HeS72
Henderson P. & Snowdon R.
"An experiment in structured programming".
BIT 12, 1 (1972).

Hoa69
(70)11,18328
Hoare C.A.R.
"An axiomatic approach to computer programming".
C. ACM 12, 10 (1969).

Hoa72a Hoare C.A.R.
"Notes on data structuring".
in [DDH72] .

Hoa72b (73)14,24718 Hoare C.A.R.
"A note on the *for* statement".
BIT 12, 3 (1972).

Hoa72c Hoare C.A.R.
"Proof of correctness of data representations".
Acta Informatica 1, 4 (1972).

Hoa72d (72)13,23801 Hoare C.A.R.
"The quality of software".
Software practice and experience 2, 2 (1972).

Hoa73a Hoare C.A.R.
"Proof of a structured program: the sieve of Eratosthenes".
Computer Journal 15 (1973).

Hoa73b Hoare C.A.R.
"Hints on programming language design".
SIGACT/SIGPLAN Symposium on principles of programming languages, Boston (1973).

Hol73 Holt R.C.
"Teaching the fatal disease (or) introductory computer programming using PL/I".
Sigplan Notices 8, 5 (1973).

Hop72 Hopkins M.E.
"A case for the *goto*".
Proceedings of ACM Nat. Conf., Boston (1972).

HoW72 Hoare C.A.R. & Wirth N.
"An axiomatic definition of the programming language Pascal".
Berichte der Fachgruppe Computer-Wissenschaften 6,
Eidgenössische Technische Hochschule, Zürich (1972).

IBM69 IBM Laboratory Vienna.
"Formal definition of PL/I".
Technical report TR 25.095-25.099 (1969).

IFIP71 International Federation of Information Processing.
IFIP Congress 1971, Ljubljana, (Yugoslavia) (1971).
Proceedings in [Fre72] .

IFIP72 International Federation of Information Processing.
IFIP Technical Committee 2 working conference on programming teaching techniques, Zakopane, Poland (1972).
Proceedings in [Tur73] .

IRH73 Ichbiah J.D., Rissen J.P. & Héliard J.C.
"The two-level approach to data definition and space management in the LIS system implementation language".
in [ACM73] .

Jen73 Jensen P.
"The grok project: data structures and process communication".
in [ACM73] .

KnF71 Knuth D.E. & Floyd R.W.
"Notes on avoiding *goto* statements".
Information processing letters 1, 1 (1971).

Knu68 (70)11,18477 Knuth D.E.
"The art of computer programming. Vol. I: Fundamental algorithms".
Addison-Wesley, Reading (1968).

Knu69 (70)11,18478 Knuth D.E.
"The art of computer programming. Vol. II: Seminumerical algorithms".
Addison-Wesley, Reading (1969).

Knu73a (73)14,25533 Knuth D.E.
"The art of computer programming. Vol. III: Sorting and searching".
Addison-Wesley, Reading (1973).

Knu73b Knuth D.E.
"A review of 'Structured programming'" (see [DDH72]).
Computer Science Department CS-371, Stanford University (1973).

Lan66 (69)10,15979 Landin P.J.
"The next 700 programming languages".
C. ACM 9, 3 (1966).

Las72 Laski J.G.
"An undergraduate computing course: a critical study of some formative concepts in a real environment".
in [Tur73] .

Lea72 Leavenworth B.M.
"Programming with(out) the *goto*".
Proceedings of ACM Nat. Conf., Boston (1972).

Lec72 Lecarme O.
"What programming language should we use for teaching programming?"
in [Tur73] .

Lec73 Lecarme O.
"An experience in structured programming and transferability".
in [ACM73] .

Lee72 Lee J.A.N.
"The formal definition of the Basic language".
Computer Journal 15, 1 (1972).

Lon72 London R.L.
"Correctness of a compiler for a Lisp subset".
Sigplan Notices 7, 7 (1972).

Lyn72 Lynch D. (SUNY at Buffalo)
Answer to Pascal questionnaire distributed by University of Colorado at Boulder (1973).

McK73 McKeag R.M.
"Programming languages for operating systems".
in [ACM73].

MCP67 McCarthy J. & Painter J.A.
"Correctness of a compiler for arithmetic expressions".
in Schwartz J.T. (ed.): *"Mathematical aspects of computer science"*
American Mathematical Society, Providence (1967).

Mil71 Mills H.
"Top-down programming in large systems".
in Rustin R. (ed.): *"Debugging techniques in large systems"*.
Prentice-Hall, Englewood Cliffs (1971).

Mil73 Miller E.F.
"Extensions to Fortran to support structured programming".
Sigplan Notices 8, 6 (1973).

MNV73 Manna Z., Ness S. & Vuillemin J.
"Inductive methods for proving properties of programs".
C. ACM 16, 8 (1973).

NaS73 Nassi I. & Shneiderman B.
"Flowchart techniques for structured programming".
Sigplan Notices 8, 8 (1973).

Nau66 (67)8,12088 Naur P.
"Proof of algorithms by general snapshots".
BIT 6, 4 (1966).

Nau69 (70)11,19420 Naur P.
"Programming by action clusters".
BIT 9, 3 (1969).

Nau72 (73)14,25213 Naur P.
"An experiment in program development".
BIT 12, 3 (1972).

Org72 Organick E.I.
"Information structure concepts for beginners. An informal approach".
in [Tur73].

Pec72 Peck J.E.L.
"Comparison of languages".
in [Tur73].

PKT73 Peterson W.W., Kasami T. & Tokura N.
"On the capabilities of *while, repeat* and *exit* statements".
C. ACM 16, 8 (1973).

Pol48 Polyà G.
"How to solve it".
Doubleday Anchor, New York (1948).

Ran62 Randell B.
"Remark on algorithm 76: sorting procedures" (see [Flo62]).
C. ACM 5, 6 (1962).

Ric68 Rice J.R.
"The *goto* statement reconsidered" (see [Dij68a]).
C. ACM 11, 8 (1968).

Ros72 Rosin R.F.
"Teaching 'about programming'".
in [Tur73].

Ros73 Rosin R.F.
"Teaching 'about programming'" (revision of [Ros72]).
C. ACM 16, 7 (1973).

Sha73 Shaw M.
"A system for structured programming".
Sigplan Notices 8, 6 (1973).

ShW73 Shaw M. & Wulf W.A.
"Global variables considered harmful".
Sigplan Notices 8, 2 (1973).

Sin68 Singleton R.C.
"An efficient algorithm for sorting with minimal storage".
C. ACM 12, 3 (1968).

ThM73 Thibault D. & Mancel P.
"Implementation of a Pascal compiler for the CII Iris 80 computer".
Sigplan Notices 8, 6 (1973).

Tur73 Turski W.M. (ed.)
"Programming teaching techniques" (proceedings of [IFIP72]).
North-Holland, Amsterdam (1973).

VdP72 van der Poel W.L.
"The programming language Hilt and its use in teaching".
in [Tur73].

Wei70 (70)11,20222 Weinberg G.M.
"PL/I programming: a manual of style".
McGraw-Hill, New York (1970).

Wei71 (72)13,23001 Weinberg G.M.
"The psychology of computer programming".
Van Nostrand Reinhold, New York (1971).

Wei72 Weinberg G.M.
"Diverse approaches to teaching programming".
in [Tur73].

WeQ72 Welsh J. & Quinn C.
"A Pascal compiler for the ICL 1900 series computer".
Software practice and experience 2, 1 (1972).

WGW71 Wulf W.A., Geschke C., Wile D. & Apperson J.
"Reflections on a systems programming language".
in [ACM71].

WiJ73 Wirth N. & Jensen K.
"A user manual for Pascal".
Eidgenössische Technische Hochschule, Zürich (1973).

Wir70 Wirth N.
"The programming language Pascal".
Berichte der Fachgruppe Computer-Wissenschaften 1,
Eidgenössische Technische Hochschule, Zürich (1970).

Wir71a (71)12,21630 Wirth N.
"Program development by stepwise refinement".
C. ACM 14, 4 (1971).

Wir71b Wirth N.
"The programming language Pascal".
Acta Informatica 1, 1 (1971).

Wir71c (72)13,23196 Wirth N.
"The design of a Pascal compiler".
Software practice and experience 1, 4 (1971).

Wir72a [do not see Hab73] Wirth N.
"The programming language Pascal (revised report)".
Berichte der Fachgruppe Computer-Wissenschaften 5,
Eidgenössische Technische Hochschule, Zürich (1972).

Wir72b Wirth N.
"On Pascal, code generation and the CDC 6000 computer".
Computer Science Department CS 257, Stanford University (1972).

Wir72c Wirth N.
"Systematisches Programmieren".
Teubner Verlag, Stuttgart (1972).

Wir73 Wirth N.
"Systematic programming: an introduction".
Prentice-Hall, Englewood Cliffs (1973).

WMY73 Weinberg G.M., Marcus R. & Yasukawa N.
"Structured programming in PL/C. An abecedarian".
John Wiley & Sons, New York (1973).

Woo70 Woodall A.D.
"An internal sorting procedure using a two-way merge".
Computer Journal 13, 1 (1970).

Woo71 Woodger M.
"On semantic levels in programming".
in [Fre72].

Wor72 Wortman D.B.
"Student PL. A PL/I dialect designed for teaching".
Proceedings of Canadian Computer Conference, Montreal (1972).

WRH71 (72)13,23011 Wulf W.A., Russell D.B. & Habermann A.N.
"Bliss: a language for systems programming".
C. ACM 14, 12 (1971).

Wul71 Wulf W.A.
"Programming without the *goto*".
in [Fre72].

Wul72 Wulf W.A.
"A case against the *goto*".
Proceedings of ACM Nat. Conf., Boston (1972).