PROGRAMMING CONCEPTS AND PRINCIPLES IN THE INTRODUCTORY COMPUTER SCIENCE TEXTBOOK

Dr. Ralph Czerwinski
Department of Mathematical
and Computer Sciences
University of Evansville
Evansville, IN 47702-0329

Introduction

In the last five years we have seen a large number of new textbooks introduced for use in the Curriculum '78 CS1 and CS2 courses [Austing]. The Curriculum '78 report coupled with the Pascal language's skyrocketing popularity as a pedagogical tool has resulted in the presentation of Pascal as the introductory language in many of these textbooks. The Curriculum '78 report includes in its list of recommended topics such items as: "programming style," "proving programs correct," "program verification," "concept and properties of algorithms," "comparative efficiency of sorting and searching methods," and "algorithm analysis." While these topics are certainly not the heart of the recommended CS1 and CS2 courses, their presence indicates an important facet of the efforts to define computer science as an academic discipline and it is interesting to take note of their coverage in the new textbooks.

My impression, from having reviewed several of these textbooks for possible adoption in our introductory sequence, was that one could differentiate between "how to program" texts and texts which presented the general concepts and principles of computer science in a systematic way teaching actual problem solving and programming as an application of the concepts and principles. By "how to program" texts, I mean those texts which place almost exclusive emphasis on the constructs, syntax, and applications of a particular language. Upon systematically inspecting 16 textbooks, which my colleagues and I had received from publishers, I found that it was not so easy to classify the textbooks. I found varying degrees of coverage of the above mentioned topics and formed some conclusions about future trends in this regard.

The topics whose coverage seemed to be indicative of an attempt to approach the subject matter, as a discipline, through its concepts and principles can be divided into two general categories: algorithm analysis and program analysis. Algorithm analysis includes such topics as time and space efficiency. The presentation of the Big-O notation demonstrated a commitment to a rather involved discussion of time efficiency. Almost all texts which discussed efficiency at all included the admonishment that efficiency was to be sacrificed for such program virtues as correctness and clarity. Some texts discussed the relatively subtle use of variable parameters for arrays to avoid the extra time and space required to copy the array's values into new storage space when a procedure is called [Starkey p. 369]. Almost all of the textbooks at least mentioned efficiency when presenting searching and sorting algorithms. Algorithm analysis in some cases included correctness and verification. In other textbooks these topics were referred to and treated as program correctness and program verification. Correctness here is to be demonstrated through logical analysis, walkthroughs, and other techniques akin to mathematical proof as opposed to various testing and debugging techniques. Finally, I include under the general heading of program analysis the topic programming style. Programming style is, of course, hard to define. Cooper and Clancy [Cooper p. 42] define The Golden Rule of Style as: "A program should be as easy for a human being to read and understand as it is for a computer to execute." Kernighan and Plauger [Kernighan] define programming style through a series of program examples they collected and analyzed.

Methodology

My method of textbook review consisted of reading the Preface to determine generally whether or not the text was geared towards an introductory course in computer science programming. This reading also gave me the author's view of what he or she had done and it gave me some insight into the author's philosophy of teaching introductory computer science. I then studied the

table of contents for chapter and section titles which might indicate the presentation of topics for which I was checking. When I found a promising title I skimmed through the relevant section looking for final verification that a topic was presented; and, if it was, I read the material thoroughly to make a judgment as to the level of presentation. Then I looked through the index for any terms which suggested a presentation of my key topics. Example index terms (under which I checked) included: algorithm analysis, correctness, efficiency, O-notation, program verification, programming style, and walkthrough. If they were present, chapter summaries provided my next searching area. By this time I had developed a sense for the textbook's coverage of the algorithm and program analysis topics; and, if I thought it promised success, I would browse through the text looking for more coverage of the key topics.

I used five categories to classify the coverage of each of six topics (see Table 1). The categories of coverage are N(one), V(ery)L(ittle), S(ome), M(ore)t(han)S(ome), and M(ore)t(han)U(sual). The classification N means that I could not find any reference to the topic. It is, of course, possible that I missed a passing reference to the topic; but, if I did, it was well hidden and would ordinarily only be noticed during a very thorough reading of the text. I used VL as a classification in those cases where all I could find was a reference(s) within a discussion of some other topic. The classification S was used whenever there was an extended discussion of the topic in some part of the book or when there was repeated reference to the topic throughout the text. Finally, MtU was used in a limited number of cases where the coverage seemed to be especially deep and sustained.

Observations

There is some argument for the conclusion that there is an increase in coverage of these six topics as the publication date advances. Table 1 is arranged so that the publication date advances as you move down the table. From Grogono through Pollack the publication dates range from 1980 up through 1983. From Belford on the texts were published in 1984 or 1985. Very few of the texts published between 1980 and 1983 rate high in all six topics. There are some notable exceptions such as Schneider, Weingart, and Perlman's [Schneider] inclusion of "style Clinics" throughout their text and Grogono's rather extensive presentation of program verification [Grogono p. 282-291]. About half of the texts published in 1984 and 1985 rate high across the six topics. A comparison of the first and 2nd editions of Cooper and Clancy's textbook [Cooper 82, Cooper 85] shows a rather dramatic increase in coverage of all six topics.

If we perform a crude quantification of Table 1 by assigning 0 to a N rating, 1 to a VL rating, 2 to a S rating, 3 to a MtS rating, and 4 to a MtU rating, we get the results in Table 2. The overall average ratings total is about 10. The average ratings total for the texts published between 1980 and 1983 is 7.5 while the average for the texts published in 1984 and 1985 is 11.6. These values support the observations in the above paragraph.

Programming Style has the best overall coverage of the six topics in the 16 textbooks. Efficiency is covered in most of the textbooks I reviewed with the inclusion of the O-notation indicating a serious commitment to the concept of efficiency. The inclusion of the O-notation only occurred at the MtS or higher level in three texts [Cooper, Dale, Starkey] all published in 1984 or 1985. Correctness is at least mentioned in all of the textbooks. The inclusion of information on formal program proof or verification indicates a more serious treatment of the correctness concept. This only happened in two texts [Dale, Grogono] at the MtS or higher level, one published in 1980 and the other in 1985.

Conclusions

I think we can expect to see more inclusion of computer science concepts and principles, such as the algorithm and program analysis topics discussed in this paper, in introductory computer science textbooks which present Pascal as their main teaching language. It is a sign of the maturation of computer science into a traditional academic discipline. The review of textbooks presented in this paper gives evidence for this conclusion. Also, any authors who decided to use Pascal as their exemplary language have already commited themselves to emphasizing the principles of structured programming. Having made this much commitment to computer science as a discipline with concepts and principles, they would understandably be more apt to include some of these other concepts and principles in their textbooks. The top-down design methodology taught with beginning Pascal textbooks leads naturally to a teaching philosophy which starts with general concepts and principles and moves to particular applications.

An additional factor is that the more recently published texts are designed to be used for full year introductory courses which will teach only one language. This is in contrast to earlier texts which in many cases had to be designed for one semester courses in Pascal, since some other language was taught during the other semester.

Table 1

Note: For brevity's sake the textbooks are identified by the last name of the first author.

	Topics					
	Time Eff.	Space Eff.	O-notation	Correctness	Proofs	Style
Grogono	VL	VL	N	MtS	MtS	VL
Cooper 82	S	VL	N	S	N	MtS
Schneider	S	S	S	VL	N	MtU
Jones	VL	VL	N	VL	VL	S
Mazlack	VL	VL	N	MtS	N	S
Pollack	VL	VL	N	VL	N	VL
Belford	S	S	N	VL	VL	S
Hume	MtS	VL	N	S	S	MtS
Skvarcius	VL	VL	N	S	S	VL
Starkey	MtS	MtS	MtS	MtS	S	S
Cooper 85	MtS	S	MtS	MtS	VL	MtU
Dale	MtS	MtS	MtS	MtU	MtU	S
Gottfried	S	S	N	VL	N	S
Graham	MtS	MtS	MtS	MtS	MtS	VL
Koffman	S	S	N	S	N	MtS
Lamprey	S	S	N	VL	N	S

Table 2

First Author	Publication Year	Ratings Total
Grogono	1980	9
Cooper 82	1982	8
Schneider	1982	11
Jones	1983	6
Mazlack	1983	7
Pollack	1983	4
Belford	1984	8
Hume	1984	11
Skvarcius	1984	7
Starkey	1984	16
Cooper 85	1985	16
Dale	1985	18
Gottfried	1985	7
Graham	1985	17
Koffman	1985	9
Lamprey	1985	7

References

1. Austing, R. H., et al., "Curriculum '78: Recommendations for the Undergraduate Program in Computer Science," Communications of the ACM, 1979, pp. 147-166.

2. Belford, G. G. and Liu, C. L., Pascal, McGraw-Hill, 1984.

3. Cooper, D. and Clancy, M., Oh! Pascal!, 1st Edition, W. W. Norton, 1982.

4. Cooper, D. and Clancy, M., Oh! Pascal!, 2nd Edition, W. W. Norton, 1985.

5. Dale, N. and Lilly, S. C., Pascal Plus Data Structures, Heath, 1985.

6. Gottfried, B. S., Programming with Pascal, McGraw-Hill, 1985.

7. Graham, N., Introduction to Computer Science, 3rd Edition, West, 1985.

8. Grogono, P., Programming in Pascal, Revised Edition, Addison-Wesley, 1980.

9. Hume, J. N. P. and Holt, R. C., VAX Pascal, Reston, 1984.

10. Jones, R. M., Introduction to Pascal and Computer Applications, Allyn and Bacon, 1983.

11. Kernighan, B. W. and Plauger, R. J., The Elements of Programming Style, McGraw-Hill, 1974.

12. Koffman, E. B., Problem Solving and Structured Programming in Pascal, 2nd Edition, Addison-Wesley, 1985.

13. Lamprey, R. H., Macdonald, R. N., and Roberts, M. W., Programming Principles Using Pascal, Harper & Row, 1985.

14. Mazlack, L., Structured Problem Solving with Pascal, Holt, Rinehart and Winston, 1983.

15. Pollack, S. V., Introducing Pascal, Holt, Rinehart and Winston, 1983.

16. Schneider, G. M., Weingart, S. W., and Perlman, D. M., An Introduction to Programming and Problem Solving with Pascal, 2nd Edition, Wiley, 1982.

17. Skvarcius, R., Problem Solving Using Pascal: Algorithm Development and Programming Concepts, PWS, 1984.

18. Starkey, J. D. and Ross, R. J., Fundamental Programming with Pascal, West, 1984. ▲