

Cambridge University Press
0521318831 - Program Construction
R. G. Stone and D. J. Cooke
Frontmatter
[More information](#)

Program construction

Also in this series

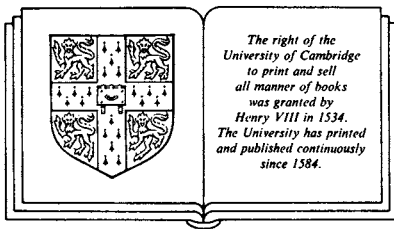
- 1 An Introduction to Logical Design of Digital Circuits
C. M. Reeves 1972
- 2 Information Representation and Manipulation in a Computer
E. S. Page and L. B. Wilson, Second Edition 1978
- 3 Computer Simulation of Continuous Systems
R. J. Ord-Smith and J. Stephenson 1975
- 4 Macro Processors
A. J. Cole, Second Edition 1981
- 5 An Introduction to the Uses of Computers
Murray Laver 1976
- 6 Computing Systems Hardware
M. Wells 1976
- 7 An Introduction to the Study of Programming Languages
D. W. Barron 1977
- 8 ALGOL 68 – A first and second course
A. D. McGettrick 1978
- 9 An Introduction to Computational Combinatorics
E. S. Page and L. B. Wilson 1979
- 10 Computers and Social Change
Murray Laver 1980
- 11 The Definition of Programming Languages
A. D. McGettrick 1980
- 12 Programming via Pascal
J. S. Rohl and H. J. Barrett 1980
- 13 Program Verification using Ada
A. D. McGettrick 1982
- 14 Simulation Techniques for Discrete Event Systems
I. Mitrani 1982
- 15 Information Representation and Manipulation using Pascal
E. S. Page and L. B. Wilson 1983
- 16 Writing Pascal Programs
J. S. Rohl 1983
- 17 An Introduction to APL
S. Pommier 1983
- 18 Computer Mathematics
D. J. Cooke and H. E. Bez 1984
- 19 Recursion via Pascal
J. S. Rohl 1984
- 20 Text Processing
A. Colin Day 1984
- 21 Introduction to Computer Systems
Brian Molinari 1985

22 Cambridge Computer Science Texts

Program construction

R. G. Stone and D. J. Cooke

Department of Computer Studies, Loughborough University of Technology



Cambridge University Press

Cambridge

New York Port Chester

Melbourne Sydney

Cambridge University Press
0521318831 - Program Construction
R. G. Stone and D. J. Cooke
Frontmatter
[More information](#)

CAMBRIDGE UNIVERSITY PRESS
Cambridge, New York, Melbourne, Madrid, Cape Town, Singapore, São Paulo

Cambridge University Press
The Edinburgh Building, Cambridge CB2 2RU, UK

Published in the United States of America by Cambridge University Press, New York

www.cambridge.org
Information on this title: www.cambridge.org/9780521268233

© Cambridge University Press 1987

This publication is in copyright. Subject to statutory exception
and to the provisions of relevant collective licensing agreements,
no reproduction of any part may take place without
the written permission of Cambridge University Press.

First published 1987
Reprinted 1988, 1990

A catalogue record for this publication is available from the British Library

Library of Congress Cataloguing in Publication data

Stone, R. G., 1950–
Program construction.
(Cambridge computer science texts; 22)
Bibliography: p.
Includes index.
I. Electronic digital computers – Programming.
I. Cooke, D. J. (Derek John), 1947–
II. Title. III. Series.
QA76.6S767 1987 005.1 86-12954

ISBN-13 978-0-521-26823-3 hardback
ISBN-10 0-521-26823-0 hardback

ISBN-13 978-0-521-31883-9 paperback
ISBN-10 0-521-31883-1 paperback

Transferred to digital printing 2006

Contents

<i>Preface</i>	ix
1 A modern approach to computing	1
1.1 An appraisal of the current situation	1
1.2 A way ahead	6
2 Specifications I	11
2.1 The nature of a specification	11
2.2 Pre and post conditions	13
2.3 Type constraints	13
2.4 Sequences of operations	16
2.5 More on types	18
2.5.1 <i>Primitive and constructed data types</i>	18
2.5.2 <i>Pairs</i>	18
2.5.3 <i>Triples</i>	21
2.5.4 <i>Tuples</i>	22
2.5.5 <i>Lists</i>	24
2.5.6 <i>Sets</i>	28
2.6 The characteristics of a specification	31
2.7 Refinement and transformation of specifications	32
2.8 States in specifications	32
2.9 States vs. Input/Output	37
2.10 Conclusion	37
3 Diagrams	39
3.1 Diagrams used in the program development process	39
3.2 An algebra of diagrams	46
3.3 Other diagramming systems	50
3.4 Graphs, networks and trees	55
4 Specifications II	61
4.1 Concise notation	61
4.2 Transformation and proof in specifications	63
4.3 What comes next?	70

vi	<i>Contents</i>	
	5 PDL	72
	5.1 Imperative and declarative languages	72
	5.2 Why a PDL?	75
	5.3 The PDL stage	76
	5.4 The description of a PDL	78
	5.4.1 <i>Function definition</i>	78
	5.4.2 <i>Statement</i>	79
	5.4.3 <i>Statements</i>	79
	5.4.4 <i>Type</i>	80
	5.4.5 <i>Variable</i>	81
	5.4.6 <i>Expressions</i>	81
	5.5 PDL data types – list and record	82
	5.6 Representing specification data types in PDL	83
	5.7 Examples	85
	5.8 Other PDL issues	86
	5.9 PDL summary	87
	6 Code generation	88
	6.1 Templates	88
	6.2 Templates for Pascal	89
	6.2.1 <i>Templates for control structures in Pascal</i>	90
	6.2.2 <i>Templates for data structures in Pascal</i>	95
	6.3 Templates for FORTRAN	99
	6.3.1 <i>Templates for control structures in FORTRAN</i>	100
	6.3.2 <i>Templates for data structures in FORTRAN</i>	106
	6.4 Templates for COBOL	115
	6.4.1 <i>Templates for control structures in COBOL</i>	115
	6.4.2 <i>Templates for data structures in COBOL</i>	121
	6.5 Templates for a minicomputer assembly language	126
	6.5.1 <i>Templates for control structures</i>	129
	6.5.2 <i>Templates for data structures</i>	139
	6.6 Templates for a microprocessor assembly language	143
	6.6.1 <i>Templates for control structures</i>	144
	6.6.2 <i>Templates for data structures</i>	151
	7 Verification	155
	7.1 The implication operator	156
	7.2 Control-flow diagrams and data-flow specification diagrams	166
	7.3 Sequencing and alternation	171
	7.4 Repetition	183
	7.4.1 <i>Simple recursion</i>	184
	7.4.2 <i>Quantifiers and induction</i>	195
	7.4.3 <i>Iteration</i>	201
	7.5 Conclusion	215

<i>Contents</i>	vii
8 Examination of templates and target code	217
8.1 Assignment statements	218
8.2 Control statements	219
8.3 Parameter passing	224
8.4 Summary	226
9 Abstract data types	227
9.1 ADT example – a siding	228
9.2 ADT example – an In_Tray	233
9.3 ADT example – LR Lookup store	236
9.4 ADT example – a binary tree	237
9.4.1 <i>Recursive implementation of tree operations in PDL</i>	240
9.4.2 <i>Non-recursive implementation of tree operations</i>	242
9.5 On preserving ADT discipline	254
9.5.1 <i>What is ADT discipline?</i>	254
9.5.2 <i>Data Type Encapsulation</i>	258
10 The mathematical basis of abstract data types	262
10.1 Booleans	262
10.2 Lists	269
10.3 Some numeric types	271
10.4 Sets	282
10.5 Equations versus conditions	285
11 Utilisation of existing programs	289
11.1 Testing for good structure	290
11.2 Restructuring of unstructured programs	302
11.3 Analysis of programs	309
12 A small scale study – topological sorting	326
12.1 Problem formulation	326
12.2 Transformations	331
12.3 Towards PDL	333
12.4 Data structure considerations	336
12.5 PDL	340
Appendices	
A Glossary of symbols	342
B Syntax of standard specifications	344
C The description of a PDL	348
D Transformations that remove recursion	353
References	365
Index	367

Preface

This text promotes the disciplined construction of procedural programs from formal specifications. As such it can be used in conjunction with any of the more conventional programming texts which teach a mixture of ‘coding’ in a specific language and *ad hoc* algorithm design.

The awareness of the need for a more methodical approach to program construction is epitomised by the use of phrases such as ‘software engineering’, ‘mathematical theory of programming’, and ‘science of programming’. The hitherto all-too-familiar practices of ‘designing’ a program ‘as you write it’ and ‘patching’ wrong programs being more appropriate to a cottage industry rather than a key activity in the current technological revolution.

The cost of producing hardware is decreasing while the production of software (programs) is becoming more expensive by the day. The complexity and importance of programs is also growing phenomenally, so much so that the high cost of producing them can only be justified when they are reliable and do what they are supposed to do – when they are correct.

No methodology can exist by which we can produce a program to perform an arbitrary task. Consequently that is **not** the aim of the book. What we **shall** do is to show how, by using a Program Design Language and templates for your chosen target language, you can develop programs from certain forms of specification.

Although programming is essentially a practical activity, the degree of formality adopted throughout the development process means that sufficient information is available to enable correctness proofs to be investigated if and when required. Moreover, the structured programming forms used throughout the text are all supported by verification rules derived from their total correctness proofs – the notion of correctness never being far from our thoughts.

The material presented has grown out of courses presented to first year Computer Science undergraduates, to ‘conversion’ M.Sc. students and in

x *Preface*

industrial short courses in software engineering, and as such has been under development since 1980.

During the evolution of the teaching material included herein we have been influenced by many sources. Of particular note is the work of Cliff Jones (now at Manchester University but previously at Oxford PRG and various IBM research establishments), on Specification; and the work of John Darlington (now at Imperial College, London and previously at Edinburgh University), on Program Transformations. At a more tangible level we wish to record our thanks to Terry Butland of UKAEE Winfrith and Morry van Ments of Loughborough's Centre for Extension Studies for their help in organising our industrial courses, to our colleagues, Mike Woodward and Dave Gittins, who helped modify earlier drafts of the text. to Jacqui Bonsor, Carole Hill and Deborah Harrison who produced the bulk of the typescript and to Ernest Kirkwood of Cambridge University Press for his encouragement and patience.

R. G. Stone
D. J. Cooke
Loughborough, 1985