Fundamental Structures of Computer Science

Fundamental Structures of Computer Science

William A. Wulf
Mary Shaw
Paul N. Hilfinger
Computer Science Department
Carnegie-Mellon University

Lawrence Flon
Computer Science Department
University of Southern California

ADDISON-WESLEY PUBLISHING COMPANY
Reading, Massachusetts • Menlo Park, California
London • Amsterdam • Don Mills, Ontario • Sydney

Reproduced from camera-ready copy provided by the authors. The copy was prepared at Carnegie-Mellon University on a photocomposer operated by the SCRIBE text editing system.

Library of Congress Cataloging in Publication Data
Main entry under title:

Fundamental structures of computer science.

Includes bibliographical references and index.
1. Electronic digital computers—Programming.
2. Data structures (Computer science) I. Wulf, William Allan.
QA76.7.F86 001.6'42 79-12374
ISBN 0-201-08725-1

ISBN 0-201-08725-1
CDEFGHIJK-DO-898765432

Preface

This textbook is designed to support an intermediate-level course in computer science. It is intended for students who have at least one or two semesters of programming and problem-solving experience and who wish to prepare for more advanced topics in computing. Ideally, the students will also have a semester's background in discrete mathematics.

The content and organization of the textbook was motivated by a single premise—that *programming is an engineering discipline.* Like other engineering desciplines, programming is concerned with building things, particularly *real* things of enormous practical importance. And, like other engineering, *good* programming is rooted in the application of science. Understanding that science is essential to being a good programmer.

This textbook is also an introduction to the science that underlies good programming. Many of the concepts involved are mathematical. Software concepts such as program organization, data structures, performance analysis, and so on, have traditionally been taught only as programming techniques. In this textbook we support these programming techniques by relating programming ideas to the underlying mathematical concepts of automata, formal languages, data types, and so on. Through examples we illustrate the relation and utility of the mathematical concepts to practical contexts.

Much of the material presented in this textbook has traditionally been taught (in greater depth) in advanced undergraduate and graduate courses. We believe, however, that it is both necessary and appropriate to introduce this material at an early stage. Just as introductory calculus and physics are

prerequisite to most engineering courses, the material presented here forms an appropriate background for courses in software engineering and advanced computer science. For the organization of this material, see the matrix on page 8 and the discussion immediately preceding it.

Using the Book

We originally designed this textbook for a second course in computer science, and since 1974 have used various generations of it for a one-semester sophomore course. One semester is not enough to cover all the material now contained in the textbook, but in a semester we can generally cover most of Parts One and Two and the theoretical material in Part Three. We also cover several of the example chapters in Part Four. Homework usually involves substantial programming exercises. Although we are aware that this is a large amount of material to cover in one semester, we have *not* observed that our students have any great difficulty with the *level* of the material.

Although we strongly believe that the material in the textbook is best presented early, that is in a second course in computer science, there are other ways that the book can be used. For example,

- As the basis of a two-semester course: This textbook contains ample material for a full-year course. In such a course, however, we would not only cover this material, but also emphasize advanced programming techniques and more advanced data structure representations.
- Merged with a software engineering course: Most of the material in this text is among the formal prerequisites for a rigorous treatment of software engineering. Half of a third or fourth year course in software engineering might profitably be spent on the chapters about formal models, specification, verification, and performance analysis.
- Incorporated into a theoretical computer science course: Much of the material in this course is predicated on a knowledge of discrete mathematics and formal logic and therefore constitutes a meaningful application of these formal mathematical ideas.
- As in-house training material for practicing programmers: Most of today's practicing programmers were educated when computing was little more than a collection of tricks. That's no longer the case. In order to compete with the recent graduates, the experienced professional will need to master the material we cover—especially the more formal material.

The language used throughout this book is essentially PASCAL. The material we cover, however, applies to programming in general, rather than to any particular language, and so we have felt free to deviate from PASCAL from time to time in presenting some material. Appendix C summarizes the major deviations. The programs in the example chapters have all been executed with a PASCAL compiler for the DEC PDP-10 originally developed at the University of Hamburg, Germany. We have, however, tried to avoid including in our examples any statements peculiar to that compiler.

Computer Science has made enormous advances in the past 10 to 15 years. It is no longer simply an art or an ad hoc collection of programming tricks. There is now a genuine body of science and mathematics that the programmer of the future must know and use. This book is hardly the last word that will be written on the subject, but it is a good beginning for new students and a sourcebook for the experienced professional.

Acknowledgments

The authors gratefully acknowledge the support and assistance of

- All those students over the past six years who have suffered through drafts of our material,
- Anita Andler, Sharon Carmack, and Dorothy Josephson, who (re)*typed much of the material, adapting to a series of dynamically developing computerized document production systems,
- The graduate students who helped teach sections of the course and who contributed material and comments: Sten Andler, Lee Cooprider, David Notkin, Gary Feldman, Lanny Forgy, Brian Reid, and Elaine Rich.
- Brian Reid, who developed and maintains SCRIBE, the document production system that allowed the manuscript of this textbook to be almost automatically produced,
- Bob Morgan, who gave some of us strong personal support,
- Jon Bentley, whose experiences with using the textbook in an industrial setting gave us new insights into ways to teach the material,
- A host of readers and reviewers, both known to us and anonymous, whose comments were invaluable, notably Terry Baker, Jon Bentley, Ron Brender, Mike Horowitz, David Jefferson, John Kender, David Lamb, David Levine, Tony Ralston, Brian Reid, Peggy Ross, Len Shustek, John Sowa, Bob Sproull, and Judy Zinnikas,

- The National Science Foundation and the Defense Advanced Research Projects Agency, which provided support for the research environment in which we learned much of what we teach here.

Pittsburgh, Pennsylvania
April 1980

W.A.W.
M.S.
P.N.H.
L.F.

Contents