Data Structures for Personal Computers

YEDIDYAH LANGSAM
MOSHE J. AUGENSTEIN
AARON M. TENENBAUM

Department of Computer and Information Science
Brooklyn College of The City University of New York

PRENTICE-HALL, INC.
Englewood Cliffs, New Jersey 07632

Library of Congress Cataloging in Publication Data

Langsam, Yedidyah (date)
Data structures for personal computers.

Bibliography: p.
Includes index.
1. Data structures (Computer science) 2. Microcomputers —Programming. 3. Basic (Computer program language)
I. Augenstein, Moshe (date). II. Tenenbaum, Aaron M.
III. Title.
QA76.9.D35L36 1985 001.64'2 84-3326
ISBN 0-13-196221-3

Editorial/production supervision and interior design: Nancy Milnamow
Cover design: Lundgren Graphics, Ltd.
Manufacturing buyer: Gordon Osbourne

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

ISBN 0-13-196221-3

Prentice-Hall International, Inc., *London*
Prentice-Hall of Australia Pty. Limited, *Sydney*
Editora Prentice-Hall do Brasil, Ltda., *Rio de Janeiro*
Prentice-Hall Canada Inc., *Toronto*
Prentice-Hall of India Private Limited, *New Delhi*
Prentice-Hall of Japan, Inc., *Tokyo*
Prentice-Hall of Southeast Asia Pte. Ltd., *Singapore*
Whitehall Books Limited, *Wellington, New Zealand*

Preface

This text is designed with two audiences in mind. One group consists of programmers who have already acquired a basic level of proficiency in programming, preferably in BASIC. Such skills may have been acquired by reading an introductory text in BASIC programming complemented by some hands-on experience on a personal computer. The programming skills acquired at this level may be disorganized and the programmer may realize that in order to solve more involved and complex problems it is necessary to learn about more high-level programming techniques. The subject of data structures coupled with enhanced programming skills is the next step in the pursuit of these high level skills.

A second group consists of those who are studying computer science in an academic environment. With the proliferation of personal computers, computer science education is becoming more popular, even in schools which previously had only one or two introductory courses in programming. Although this description will typically fit two-year schools or high schools, a number of four-year colleges with small budgets for computing also fit into this category. BASIC is frequently the language used at such institutions.

The purpose of this book is to introduce the reader to the elementary concepts of data structures in conjunction with reinforcement of high-level programming skills.

For several years, we have taught a course in data structures to students who have had a semester course in high-level programming and a semester course in assembly language programming. We found that a considerable amount of time was spent in teaching programming techniques because the students had not had sufficient exposure to programming and were unable to implement abstract struc-

tures on their own. The brighter students eventually caught on to what was being done. The weaker students never did. Based on this experience, we have reached the firm conclusion that a first course in data structures must go hand in hand with a second course in programming. This text is a product of that conviction.

The text introduces abstract concepts, shows how these concepts are useful in problem solving and then shows how the abstractions can be made concrete by using a programming language. Equal emphasis is placed on both the abstract and the concrete versions of a concept, so that the student learns about the concept itself, its implementation, and its application.

The language used in this book is BASIC. Although there are several languages which support good programming techniques and are better than BASIC for implementing abstract data structures, we have selected BASIC for several reasons. BASIC is the most widely-used high-level language today because of its widespread accessibility on personal computers. Within nonacademic circles, there is a growing interest in computer science. Many people who have an interest in data structures, but without programming skills in another high level language, have few sources to which to turn. Furthermore, although BASIC has become far from universally accepted (and will probably never be) within academic circles, its use in recognized computer science programs is spreading (particularly, as we mentioned earlier, at smaller institutions). Although BASIC has been criticized as being very problem-prone, it can be used correctly. In Chapter 2 we introduce a consistent approach to BASIC and continue to emphasize that approach throughout the remainder of the book. The only prerequisite for students using this book is the equivalent of a one-semester course in programming in BASIC. Readers who are not familiar with BASIC are referred to the Bibliography for a selection of introductory texts in the language.

Chapter 1 is an introduction to data structures. Section 1.1 introduces the concept of an abstract data structure and the concept of an implementation. Section 1.2 introduces arrays—their implementation as well as their application. Section 1.3 introduces data aggregates and how they can be implemented in BASIC.

Chapter 2 introduces and discusses structured programming techniques in BASIC and their algorithmic counterparts. These techniques present a style of programming that is used throughout the remainder of the text.

Chapter 3 discusses stacks and their BASIC implementation. Because this is the first new data structure introduced, considerable discussion of the pitfalls of implementing such a structure is included. Section 3.4 introduces postfix, prefix, and infix notations.

Chapter 4 introduces queues and linked lists and their implementations using an array of available nodes.

Chapter 5 discusses recursion and its applications. Because recursion is not implemented on most versions of BASIC, methods of simulating recursion are presented as well.

Chapter 6 discusses trees and Chapter 7 introduces graphs.

Chapter 8 covers sorting and Chapter 9 covers searching.

At the end of the book, we have included a bibliography listing a selected set of texts in the areas of BASIC programming and data structures, to which the reader is referred for further reading. In a one-semester course, Chapter 7 and parts of Chapters 1, 2, 6, 8, and 9 can be omitted.

The text is suitable for course I1 of Curriculum 68 (Communications of the ACM, March 1968), courses UC1 and UC8 of the Undergraduate Programs in Information Systems (Communications of the ACM, Dec. 1973) and course CS2 and parts of courses CS7 and CS13 of Curriculum 78 (Communications of the ACM, March 1979). In particular, the text covers parts or all of topics P1, P2, P3, P4, P5, S2, D1, D2, and D6 of Curriculum 78.

Algorithms (which we introduce in Chapter 2) are presented as intermediaries between English language descriptions and BASIC programs. They are written in a style consisting of high-level constructs interspersed with English. These algorithms allow the reader to focus on the method used to solve a problem without concern about declaration of variables and the peculiarities of a real language. In transforming an algorithm into a program, we introduce these issues and point out the pitfalls which accompany them.

The indentation pattern used for BASIC programs and algorithms is based on a format introduced in Chapter 2 which we have found to be a useful tool in improving program comprehensibility. We distinguish between algorithms and programs by presenting the former in lower case italics and the latter in upper case roman.

Most of the concepts in the text are illustrated by several examples. Some of these examples are important topics in their own right (e.g., postfix notation, multi-word arithmetic, etc.) and may be treated as such. Other examples illustrate different implementation techniques (such as sequential storage of trees). When using this text for a one-semester course, the instructor is free to cover as many or as few of these examples as he or she wishes. Examples may also be assigned to students as independent reading. It is anticipated that an instructor will be unable to cover all the examples in sufficient detail within the confines of a one- or two-semester course. We feel that, at the stage of student's development for which the text is designed, it is more important to cover several examples in great detail than to cover a broad range of topics cursorily.

The exercises vary widely in type and difficulty. Some are drill exercises to insure comprehension of topics in the text. Others involve modifications of programs or algorithms presented in the text. Still others introduce new concepts and are quite challenging. Often, a group of successive exercises includes the complete development of a new topic which can be used as the basis for a term project or an additional lecture. The instructor should use caution in assigning exercises so that an assignment is suitable to the student's level. We consider it imperative for students to be assigned several (from five to twelve, depending on difficulty) programming projects per semester. The exercises contain several pro-

jects of this type. The instructor may find a great many additional exercises and projects in the *Exercise Manual* of one of our earlier texts, *Data Structures and PL/I Programming* (Prentice-Hall, 1979). Although many of the exercises in that manual are presented using PL/I, they can readily be recast in a BASIC setting. The *Exercise Manual* for *Data Structures and PL/I Programming* is available from the publisher.

One of the most difficult choices which had to be made in writing this book was the question of which BASIC dialect to use. In order to present programs which would run on a wide variety of personal computers, it is desirable to choose the "lowest common denominator" of all commonly available BASIC dialects. On the other hand, by choosing a very small proper subset of BASIC, our programs would not be able to take advantage of "standard" BASIC features provided by the vast majority of personal computers. We decided to ensure that the programs in this book would run under each of Radio Shack BASIC Level II, Microsoft BASIC-80, and BASIC for the IBM PC. Of these three, Radio Shack BASIC Level II is fairly close to being a proper subset of the other two, and yet provides all the features which we deemed essential. One of the limitations of Radio Shack BASIC Level II is that it distinguishes variables by only the first two characters of their names and forbids the use of embedded reserved words. The same restriction applies to Applesoft BASIC. We have taken great pains to use meaningful variable names and yet to abide by these limitations. Naturally, in those versions of BASIC which do not have these limitations, the programmer is free to substitute somewhat less awkward variable names. We have deliberately not taken advantage of those advanced features (e.g., the WHILE-WEND construct, the MOD built-in function, etc.) of Microsoft BASIC-80 and BASIC for the IBM PC that are not supported by the majority of BASICs currently available for personal computers. However, we do introduce these constructs in Chapter 2 and do use them in presenting algorithms.

One feature which we felt we could not omit was the ELSE clause for the IF-THEN construct. Without the availability of the IF-THEN-ELSE, programs would become unwieldy and their pedagogical value would be greatly diminished. Unfortunately, Applesoft BASIC does not support the ELSE clause. The Applesoft programmer may simulate ELSE clauses by methods presented in Chapter 2. We also use the DEF statement to declare variable types rather than relying on the special type symbols. This is also invalid in Applesoft BASIC but can easily be remedied by inserting the type symbols. All other features used throughout this book are also valid in Applesoft BASIC. Each program (or subroutine) in this book has been tested on a Radio Shack Model III using BASIC Level II, on an Apple II Plus equipped with a Softcard using Microsoft BASIC-80, and on an IBM PC using cassette BASIC. We wish to thank Imran Khan, Linda Laub, Diana Lombardi, Joel Plaut, and Chris Ungeheuer for their invaluable assistance in this task. Their zeal for the task was above and beyond the call of duty and their suggestions were always valuable. Of course, any errors that remain are the sole responsibility of the authors.

We have prepared two sets of diskettes containing the BASIC source code of programs and subroutines in the text. One set of diskettes was prepared under BASIC-80 using the Microsoft CP/M Softcard for the Apple II Plus and the second set using IBM PC BASIC. These diskettes are available from the publisher using the tear-off card bound into the book.

Linda Laub, Carl Markowitz, and Chris Ungeheuer spent many hours typing and correcting the original manuscript. Their cooperation and patience as we continually changed our minds about additions and deletions are most sincerely appreciated. We wish to single them out for their extraordinary enthusiasm and dedication in all phases of the book's production, for which we are deeply grateful.

We would like to thank Maria Argiro, Mirrel Eissenberg, Beverly Heller, Gun Kim, Amalia Kletsky, Sholom Krischer, Linda Laub, Diana Lombardi, Chaim Markowitz, Joel Plaut, Barbara Reznik, Chris Ungeheur, and Shirley Yee for their invaluable assistance.

The staff of the City University Computer Center deserves special mention. They were extremely helpful in assisting us in using the excellent facilities of the Center. The same can be said of Julio Berger and Lawrence Schweitzer and the rest of the staff of the Brooklyn College Computer Center.

We would like to thank the editors and staff at Prentice-Hall and especially the reviewers for their helpful comments and suggestions.

Finally, we thank our wives, Vivienne Langsam, Gail Augenstein, and Miriam Tenenbaum, for their advice and encouragement during the long and arduous task of producing such a book.

Yedidyah Langsam
Moshe Augenstein
Aaron Tenenbaum

Contents

2 PROGRAMMING IN BASIC 43

1

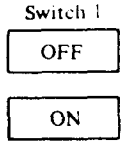
Introduction to Data Structures

A computer is a machine that manipulates information. The study of computer science includes the study of how information is organized in a computer, how it can be manipulated, and how it can be utilized. Thus it is extremely important for a student of computer science to understand the concepts of information organization and manipulation in order to continue study of the field.

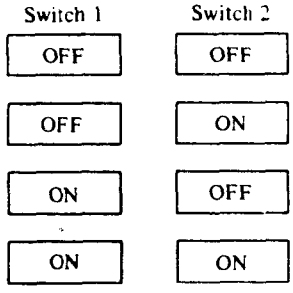
1. INFORMATION AND MEANING

If computer science is fundamentally the study of information, the first question that arises is: What is information? Unfortunately, although the concept of information is the bedrock of the entire field, this question cannot be answered precisely. In this sense, the concept of information in computer science is similar to the concepts of point, line, and plane in geometry—they are all undefined terms about which statements can be made but which cannot be explained in terms of more elementary concepts.

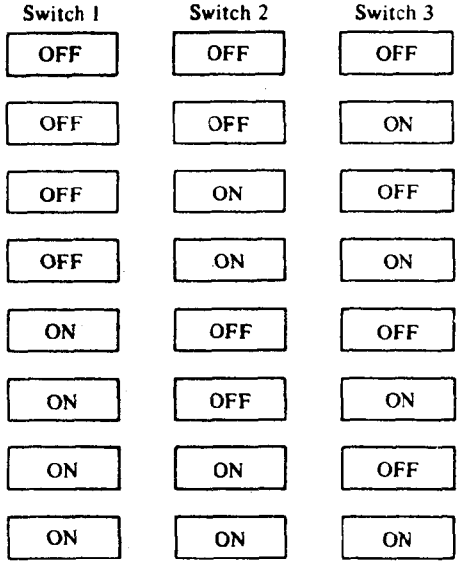
In geometry, it is possible to talk about the length of a line despite the fact that the concept of a line itself is undefined. The length of a line is a measure of quantity. Similarly, in computer science, we can measure quantities of information. The basic unit of information is the ***bit***, whose value asserts one of two mutually exclusive possibilities. For example, if a light switch can be in one of two positions but not in both simultaneously, the fact that it is either in the "on" position or the "off" position is 1 bit of information. If a device can be in more than two possible states, the fact that it is in a particular state is more than 1 bit of



(a) One switch (two possibilities).



(b) Two switches (four possibilities).



(c) Three switches (eight possibilities). **Figure 1.1.1**