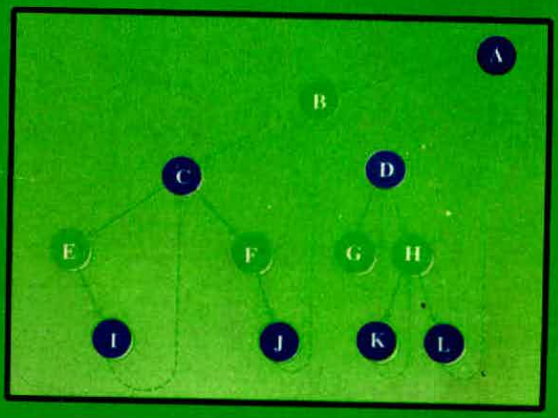
Eastern Economy Edition



SECOND EDITION

DATA STRUCTURES USING C AND C++

Yedidyah Langsam • Moshe J. Augenstein
Aaron M. Tenenbaum

PHI

Second Edition

DATA STRUCTURES USING C AND C++

Yedidyah Langsam
Moshe J. Augenstein
Aaron M. Tenenbaum

Brooklyn College

Prentice-Hall of India Private Limited
New Delhi - 110 001
2000

This Nineteenth Indian Reprint—Rs. 225.00
(Original U.S. Edition—Rs. 2964.00)

DATA STRUCTURES USING C AND C++, 2nd Ed.
by Yedidyah Langsam, Moshe J. Augenstein and Aaron M. Tenenbaum

ISBN-81-203-1177-9

Reprinted in India by special arrangement with Prentice-Hall, Inc., Upper Saddle River, New Jersey 07458, U.S.A.

Nineteenth Printing (Second Edition) ... ... December, 2000

Published by Asoke K. Ghosh, Prentice-Hall of India Private Limited, M-97, Connaught Circus, New Delhi-110001 and Printed by Syndicate Binders, B-167, Okhla Industrial Area, Phase I, New Delhi-110020.

Contents

Preface

This text is designed for a two-semester course in data structures and programming. For several years, we have taught a course in data structures to students who have had a semester course in high-level language 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 structures 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 conviction 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 those 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 languages used in this text are C and C++. C is well suited to such a course since it contains the control structures necessary to make programs readable and allows basic data structures such as stacks, linked lists, and trees to be implemented in a variety of ways. This allows the student to appreciate the choices and tradeoffs which face a programmer in a real situation. C is also widespread on many different computers and it continues to grow in popularity. As Kernighan and Ritchie indicate, C is "a pleasant, expressive, and versatile language."

We have included information on C++ in the early chapters, introducing the features of C++ and showing how they can be used in implementing data structures. No specific background in C++ is needed. Classes in C++ are introduced in a new Section 1.4. This section discusses classes, including function members. It also introduces inheritance and object orientation. The section includes an example of implementing abstract data types in C++, as well as polymorphism. To Section 2.3 we have added an implementation of stacks in C++ using templates. This shows how complex data structures can be parameterized for different base types. A new Section 4.6 has been added, showing how linked lists can be implemented in C++. Such an implementation shows the limitations, as well as the power, of encapsulation in implementing data structures. The point should be made that encapsulated data structures must be designed carefully to allow users to do what they need in a data structure. Also discussed in this context are C++ dynamic allocation and freeing of storage.

The only prerequisite for students using this text is a one-semester course in programming. Students who have had a course in programming using such languages as FORTRAN, Pascal, or PL/I can use this text together with one of the elementary C or C++ texts listed in the Bibliography. Chapter 1 also provides information necessary for such students to acquaint themselves with C.

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. Sections 1.2 and 1.3 introduce arrays and structures in C. The implementations of these two data structures as well as their applications are covered. Chapter 2 discusses stacks and their C implementation. Since this is the first new data structure introduced, considerable discussion of the pitfalls of implementing such a structure is included. Section 2.3 introduces postfix, prefix, and infix notations. Chapter 3 covers recursion, its application, and its implementation. Chapter 4 introduces queues, priority queues, and linked lists and their implementations both using an array of available nodes as well as using dynamic storage. Chapter 5 discusses trees, Chapter 6 introduces O notation and covers sorting, while Chapter 7 covers both internal and external searching. Chapter 8 introduces graphs, and Chapter 9 discusses storage management. At the end of the text, we have included a large Bibliography with each entry classified by the appropriate chapter or section of the text.

A one-semester course in data structures consists of Section 1.1, Chapters 2 through 7, and Sections 8.1, 8.2, and part of Section 8.4. Parts of Chapters 3, 6, 7, and 8 can be omitted if time is pressing.

This text is suitable for courses based upon the Algorithms and Data Structures knowledge unit (AL 1-6, 8) as well as sections of the Programming Languages knowledge unit (PL 3-6, 10, 11) as described in the report *Computing Curricula* 1991 of the ACM/IEEE-CS Joint Curriculum Task Force. It follows closely the sample Data Structures and Analysis of Algorithms course presented in the report and may be used in second- and third-tier classes of a typical computer science curriculum for both majors and nonmajors.

The text is suitable for course C82 and parts of courses C87 and C813 of Curriculum 78 (*Communications of the ACM*, March 1979), courses UC1 and UC8 of the Undergraduate Programs in Information Systems (*Communications of the ACM*, December 1973) and course I1 of Curriculum 68 (*Communications of the ACM*, March

1968). In particular, the text covers parts or all of topics P1, P2, P3, P4, P5, S2, D1, D2, D3, and D6 of Curriculum 78.

Algorithms are presented as intermediaries between English language descriptions and C programs. They are written in C style 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 real language. In transforming an algorithm into a program, we introduce these issues and point out the pitfalls that accompany them.

The indentation pattern used for programs and algorithms is based loosely on a format suggested by Kernighan and Ritchie (*The C Programming Language*, Prentice Hall, 1978) which we have found to be quite useful. We have also adopted the convention of indicating in comments the construct being terminated by each instance of a closing brace (}). Together with the indentation pattern, this is a valuable tool in improving program comprehensibility. We distinguish between algorithms and programs by presenting the former in italics and the latter in 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, multiword arithmetic, etc.) and may be treated as such. Other examples illustrate different implementation techniques (such as sequential storage of trees). 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 a 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.

All the programs and algorithms in this text have been tested and debugged. We wish to thank Miriam Binder and Irene LaClaustra 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.

The exercises vary widely in type and difficulty. Some are drill exercises to ensure 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 projects of this type.

We have attempted to use the C language, as specified in the second edition of the Kernighan and Ritchie text. This corresponds to the C ANSI Standard. Programs given in this book have all been developed using Borland C++ but have only made use of features as described in the evolving ANSI C++ draft standard. They should run without change on a wide variety of C++ compilers. See the reference manual for your particular system or consult the "Working Paper for Draft Proposed International Standard for Information System—Programming Language C++," available from the

American National Standards Institute (ANSI) Standards Secretariat: CBEMA, 1250 Eye Street NW, Suite 200, Washington, DC 20005. You should, of course, warn your students about any idiosyncrasies of the particular compiler they are using. We have also added some references to several personal computer C and C++ compilers.

Miriam Binder and Irene LaClaustra spent many hours typing and correcting the original manuscript as well as managing a large team of students whom we mention below. Their cooperation and patience as we continually made up and changed our minds about additions and deletions are most sincerely appreciated.

We would like to thank Shaindel Zundel-Margulis, Cynthia Richman, Gittie Rosenfeld-Wertenteil, Mindy Rosman-Schreiber, Nina Silverman, Helene Turry, and Devorah Sadowsky-Weinschneider for their invaluable assistance.

Vivienne Esther Langsam and Tziyonah Miriam Langsam spent many hours revising the index for the second edition of this book. We would like to thank them for helping us complete the book in the face of a fast approaching deadline.

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 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 Esther 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 J. AUGENSTEIN
AARON M. TENENBAUM

To my wife, Vivienne Esther
YL

To my wife, Gail
MA

To my wife, Miriam
AT
