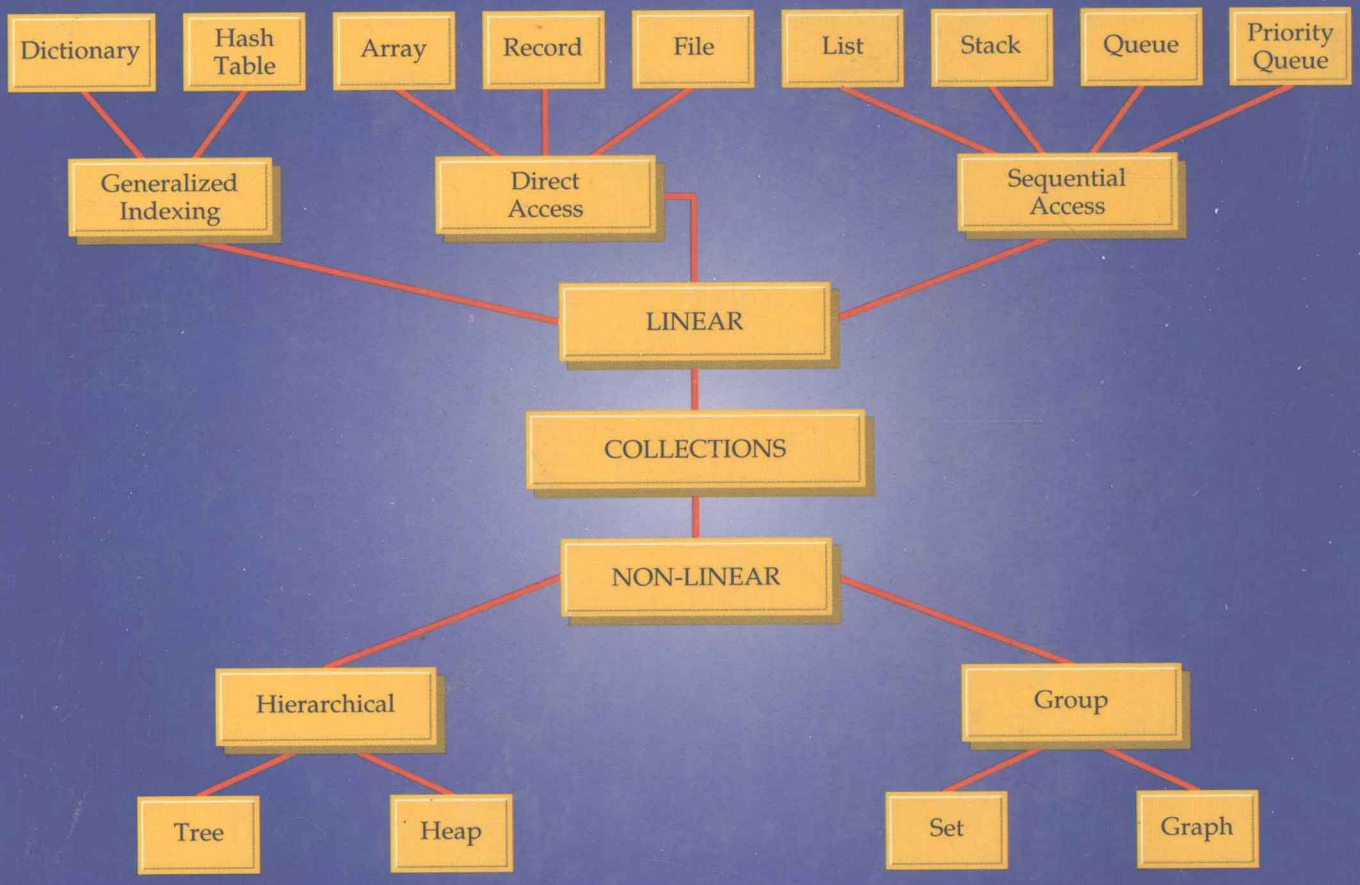
Data Structures with C++



William Ford • William Topp

DATA STRUCTURES

WITH

C++

William Ford
University of the Pacific

William Topp
University of the Pacific

Prentice Hall, Upper Saddle River, New Jersey 07458

Library of Congress Cataloging-in-Publication Data
Ford, William
Data structures with C++ / by William Ford/William Topp.
p. cm.
Includes bibliographical references and index.
ISBN 0-02-420971-6 : $36.00
1. C++ (Computer program language) 2. Data structures (Computer science) I. Topp, William R., 1939– . II. Title.
QA76.73.C153F67 1996 94-10482
005.7'3—dc20 CIP

Editor-in-Chief: Marcia Horton
Acquisitions Editor: Alan Apt
Production Editor: Bayani Mendoza de Leon
Developmental Editors: Elizabeth Jones/Marcia Holman
Cover Designer: Precision Graphics
Production Coordinator: Spectrum Publisher Services
Buyer: Donna Sullivan
Editorial Assistant: Shirley McGuire

A Simon & Schuster Company
Upper Saddle River, New Jersey 07458

Printed in the United States of America

10 9 8 7

ISBN 0-02-420971-6

Prentice-Hall International (UK) Limited, *London*
Prentice-Hall of Australia Pty, Limited, *Sydney*
Prentice-Hall Canada, Inc., *Toronto*
Prentice-Hall Hispanoamericana, S.A., *Mexico*
Prentice-Hall of India Private Limited, *New Delhi*
Prentice-Hall of Japan, Inc., *Tokyo*
Simon & Schuster Asia Pte. Ltd., *Singapore*
Editora Prentice-Hall do Brasil, Ltda, *Rio de Janeiro*

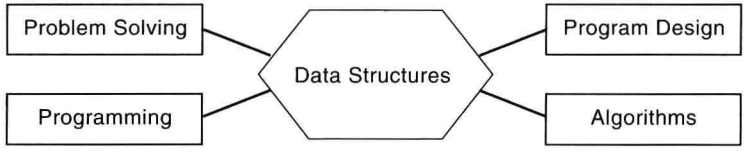
To

David Johnstone, Editor

He shared a vision with us. Despite his tragic death from an act of random violence, we kept alive the vision in our work. We hope it is a fitting tribute.

PREFACE

This book is designed to present the fundamentals of data structures from an object-oriented perspective. The study of data structures is core to a computer science curriculum. It provides a rich context for the study of problem-solving techniques and program design and utilizes powerful programming constructs and algorithms.



This book uses the versatile language C++ whose classes and object-oriented constructs are specifically designed to efficiently implement data structures. Although a number of object-oriented languages are available, C++ has developed a preeminence due to its origins in the popular C programming language and its use by many software vendors. We develop each data structure around the concept of an abstract data type (ADT) that defines both data organization and data handling operations. We are supported by the C++ language that provides a class type to represent an ADT and to efficiently use the structures in an object.

Design of the Book

Data Structures with C++ organizes the study of data structures around collection classes that include lists, trees, sets, graphs, and dictionaries. In the process, we cover the fundamental topics of data structures and develop object-oriented programming methodology. The structures and methodology are implemented in a series of complete programs and case studies. To evaluate the efficiency of algorithms, we give a simple and early introduction to Big-O notation.

Chapters 1 to 11 provide the traditional topics in a first course in data structures (CS 2). A formal treatment of inheritance and virtual functions is given in Chapter 12 and the topics are used to implement the advanced data structures in Chapters 13 and 14. Overall, the material in Chapters 12 to 14 defines topics traditionally covered in an advanced data structures/algorithms course (CS 7) and an advanced programming course. We include a careful development of templates and operator

overloading to support generalized structures. We use these powerful C++ language constructs to simplify our use of the data structures.

A computer professional could use *Data Structures with C++* as a self-study guide to data structures, which would make it possible to understand most class libraries, research articles, and advanced trade publications.

Chapter Descriptions

Most of the book's chapters develop abstract data types and describe their implementation as a C++ class. The declaration of each class and its key methods also are included in the book. In many cases, the full definition is given, yet in others, the definition of selected class methods are given. The full implementation of the classes are included in a program supplement.

CHAPTER 1: INTRODUCTION

This chapter is an overview chapter that introduces abstract data types and object-oriented programming using C++. The concept of an ADT and the related attributes of data encapsulation and information hiding are developed. This chapter also introduces inheritance and polymorphism, which are formally covered in Chapter 12.

CHAPTER 2: BASIC DATA TYPES

Programming languages provide primitive numeric and character types that cover integer and floating point numbers, character data, and user-defined enumeration types. The primitive types combine to create array, record, string, and file structures. This chapter describes ADTs for language types using C++ as an example.

CHAPTER 3: ABSTRACT DATA TYPES AND CLASSES

This book as a whole provides a formal study of ADTs and their representation as C++ classes. Specifically, this chapter defines basic class concepts including data members, constructors, and method definitions.

CHAPTER 4: COLLECTION CLASSES

A collection is a storage class with data handling tools to add, delete, or update the items. The study of collection classes is the main focus of this book. Therefore, this chapter provides an example of the different collection types that are presented in the book. The chapter includes a simple early introduction to the Big-O notation, which measures the efficiency of an algorithm. The notation is used throughout the book to compare and contrast different algorithms. The chapter concludes with a study of the SeqList class that is a prototype of a general list structure.

CHAPTER 5: STACKS AND QUEUES

This chapter discusses stacks and queues, which are fundamental collection classes that maintain data in LIFO (last-in first-out) and FIFO (first-in first-out) order. It also develops the priority queue, a modified version of a queue in which the client always deletes the item of highest priority from the list. A case study uses priority queues to perform event-driven simulation.

CHAPTER 6: ABSTRACT OPERATORS

An abstract data type defines a set of methods to initialize and manage data. In this chapter, we extend language-defined operators (e.g., +, *, <<, etc.) to abstract data types. The process, called operator overloading, redefines standard operator symbols to implement operations in the ADT. A fully developed rational number class illustrates operator overloading and type conversion, as well as introducing friends to overload the standard C++ I/O operators.

CHAPTER 7: GENERIC DATA TYPES

C++ uses the template mechanism to provide for generic functions and classes that support different data types. Templates provide powerful generality to our data structures. This concept is illustrated with a template-based version of the Stack class and its application to infix expression evaluation.

CHAPTER 8: CLASSES AND DYNAMIC MEMORY

Dynamic data structures use memory allocated by the system at run time. They allow us to define structures without size constraints and enhance the usability of our classes. Their use, however, requires careful attention. We introduce the copy constructor, overloaded assignment operator, and destructor methods, which allow us to properly copy and assign dynamic data and then deallocate it when an object is deleted. The power of dynamic data is illustrated with the Array, String, and Set classes. These classes are used throughout the remainder of the book.

CHAPTER 9: LINKED LISTS

The use of lists to store and retrieve data is a continuing theme in the book because lists are fundamental to the design of most data applications. This chapter introduces linked lists, which allow for dynamic list handling. We use a twofold approach that first develops a basic node class and creates functions for adding or deleting items from the list. A more abstract approach creates a linked list class with a built-in traversal mechanism to scan the items in the list. The LinkedList class is used to implement the SeqList class and the Queue class. In each case, a linked list object is included by composition. The approach provides a powerful tool for developing data structures. This chapter also discusses circular and doubly linked lists that have interesting applications. The chapter features a printer queue case study as well.

CHAPTER 10: RECURSION

Recursion is an important problem-solving tool in both computer science and mathematics. We introduce recursion and illustrate its use in a variety of contexts. A series of applications uses recursion with mathematical formulas, combinatorics, maze traversal, and puzzles. The Fibonacci sequence is used to compare the efficiency of a recursive algorithm, an iterative algorithm, or direct calculations in computing a term of the sequence.

CHAPTER 11: TREES

Linked lists define a set of nodes that are sequentially accessed beginning at the head. The data structure is called a linear list. In many applications, objects exhibit a nonlinear order in which a member may have multiple successors. In Chapter 11, we introduce a basic nonlinear structure called a tree in which all data items emanate from a single source—the root. A tree is an ideal structure for describing a hierarchical structure such as a computer file system and a business reporting chart. In this chapter, we restrict our analysis to binary trees in which each node has, at most, two descendants. We develop the TreeNode class to implement these trees and present applications that include the classical preorder, inorder, and postorder scan algorithms. Binary trees find application as a list structure that efficiently stores large volumes of data. The structure, called a binary search tree, is implemented in the BinSTree class. The class is featured in a case study that develops a document concordance.

CHAPTER 12: INHERITANCE AND ABSTRACT CLASSES

Inheritance is a fundamental concept in object-oriented programming. This chapter discusses the main features of inheritance, carefully develops its implementation in C++, and introduces virtual functions as tools that utilize the power of inheritance. It also develops the concept of an abstract base class with pure virtual functions Virtual functions are fundamental to object-oriented programming and are used with subsequent topics in the book. This chapter includes the introduction of iterators that define a uniform and general traversal mechanism for the different lists in the book. It concludes with an example of inheritance and virtual functions to develop heterogenous arrays and linked lists.

CHAPTER 13: ADVANCED NONLINEAR STRUCTURES

This chapter continues the development of binary trees and introduces additional nonlinear structures. It describes array-based trees that model an array as a complete binary tree. An extensive study of heaps is provided, and the concept is used to implement the heap sort and priority queues. Although binary search trees are usually good structures with which to implement a list, degenerate cases can be inefficient. Data structures provide different height-balanced structures that ensure fast average search time. Using inheritance, a new search tree class called AVL trees is derived. The chapter concludes with an introduction to graphs that features a series of classic algorithms.

CHAPTER 14: ORGANIZING COLLECTIONS

This chapter looks at searching and sorting algorithms for general collections. In the process, the classical array-based selection, bubble, and insertion sort algorithms are developed. Our study includes the famous QuickSort algorithm. In this book, data that is stored in internal memory is emphasized. For larger sets, data can be stored on disk and external methods to search and sort the data can be used. We develop the BinFile class for direct file access, and use its methods to illustrate both the external index sequential search and the external merge sort algorithm. A section on associative arrays, or dictionaries, generalizes the concept of an array index.

Required Background

This book assumes the reader has completed a first course in programming and is fluent with basic C++. Chapter 2 defines the primitive data structures of C++ and illustrates their uses in several complete programs. This chapter can be used as a standard for defining the C++ prerequisites. For the interested reader, the authors provide a C++ tutorial that defines the primitive types of the language and the syntax for arrays, control structures, I/O, functions, and pointers. The tutorial includes a discussion of each topic along with examples, complete programs, and exercises.

Supplements

Complete source code listings for all classes and programs are available through an Internet ftp connection from the authors' institution, the University of the Pacific. The C++ code in the book has been tested and run using the latest Borland compiler. With very few exceptions, the programs also compile and run on a Macintosh system using Symantec C++ and on a Unix system using GNU C++.

For those having Internet connection, execute an ftp to "ftp.cs.uop.edu". Upon connecting to the system, your login name is "anonymous" and your password is your Internet mail address. The software is located in the directory "/pub/C++".

Readers may contact the authors directly to receive a copy of the tutorial. Order information is available by electronic mail—send to "billf@uop.edu"—or by the U.S. mail—write to Bill Topp, 456 S. Regent, Stockton, CA 95204.

The Instructor's Guide offers teaching tips for each chapter, answers to most written exercises, and sample tests. The guide features solutions to many of the programming exercises and is available from Prentice Hall.

Acknowledgments

The authors have been supported by friends, students, and colleagues throughout the preparation of *Data Structures with C++*. The University of the Pacific has generously provided resources and support to complete the project. Prentice Hall

offered a dedicated team of professionals who handled the book design and production. We are especially grateful to editors Elizabeth Jones, Bill Zobrist, and Alan Apt, and to production editor Bayani de Leon. Production was jointly implemented by Spectrum Publisher Services and Prentice Hall. We were greatly assisted by Kelly Ricci and Kristin Miller at Spectrum.

Students have offered valuable criticism of the manuscript by giving us explicit feedback or unsolicited blank stares. Our reviewers offered guidance for early writing of the manuscript, providing detailed comments on both the content and the pedagogical approach. We took most of their recommendations into account. Special thanks go to Hamid R. Arabnia, University of Georgia; Rhoda A. Baggs, Florida Institute of Technology; Sandra L. Bartlett, University of Michigan–Ann Arbor; Richard T. Close, U.S. Coast Guard Academy; David Cook, U.S. Air Force Academy; Charles J. Dowling, Catonsville (Baltimore County) Community College; David J. Haglin; Mankato State University; Jim Murphy, California State University–Chico; and Herbert Schildt. Two colleagues, Ralph Ewton at the University of Texas–El Paso, and Douglas Smith at the University of the Pacific made extensive contributions. Their insights and support were invaluable to the authors and greatly improved the final design of the book.

William Ford
William Topp

CONTENTS

CHAPTER 2 BASIC DATA TYPES 38

CHAPTER 3 ABSTRACT DATA TYPES AND CLASSES 91