

大学计算机教育丛书（影印版）

DATA  
STRUCTURES  
with **C++**

数据结构

**C++**语言描述

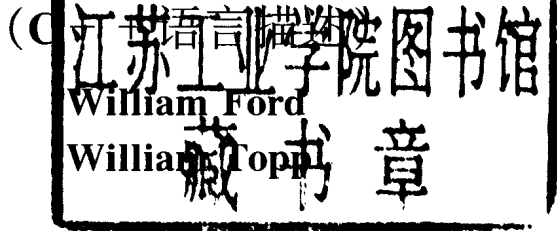
William Ford  
William Topp



清华大学出版社 • PRENTICE HALL

**DATA STRUCTURES**  
**WITH**  
**C + +**

数据结构



清华大学出版社  
Prentice-Hall International, Inc.

# **(京)新登字 158 号**

Data structures with C++ / William Ford, William Topp.

© 1996 by Prentice Hall, Inc.

Original English Language Edition Published by Prentice Hall, Inc.

All Rights Reserved.

For sale in Mainland China only.

本书影印版由西蒙与舒斯特国际出版公司授权清华大学出版社在中国境内(不包括香港、澳门特别行政区和台湾地区)独家出版、发行。

未经出版者书面许可,不得用任何方式复制或抄袭本书的任何部分。

**本书封面贴有清华大学出版社激光防伪标签,无标签者不得销售。**

北京市版权局著作权合同登记号: 01-97-0169

## **图书在版编目(CIP)数据**

数据结构(C++语言描述): 英文/(美)福特(Ford, W.), (美)托普(Topp, W.)著. - 北京:清华大学出版社, 1997.1

(大学计算机教育丛书;影印版)

ISBN 7-302-02413-8

I. 数… II. ①福… ②托… III. C语言-数据结构-高等学校-教材-英文  
IV. TP311.12

中国版本图书馆 CIP 数据核字(96)第 25164 号

出版者: 清华大学出版社(北京清华大学学研大厦, 邮编 100084)

<http://www.tup.tsinghua.edu.cn>

印刷者: 清华大学印刷厂

发行者: 新华书店总店北京发行所

开 本: 850×1168 1/32 印张: 28.75

版 次: 1997 年 3 月第 1 版 2001 年 6 月第 8 次印刷

书 号: ISBN 7-302-02413-8/TP·1215

印 数: 23001~26000

定 价: 54.00 元

## 出版前言

我们的大学生、研究生毕业后,面临的将是一个国际化的信息时代。他们将需要随时查阅大量的外文资料;会有更多的机会参加国际性学术交流活动;接待外国学者;走上国际会议的讲坛。作为科技工作者,他们不仅应有与国外同行进行口头和书面交流的能力,更为重要的是,他们必须具备极强的查阅外文资料获取信息的能力。有鉴于此,在国家教委所颁布的“大学英语教学大纲”中有一条规定:专业阅读应作为必修课程开设。同时,在大纲中还规定了这门课程的学时和教学要求。有些高校除开设“专业阅读”课之外,还在某些专业课拟进行英语授课。但教、学双方都苦于没有一定数量的合适的英文原版教材作为教学参考书。为满足这方面的需要,我们挑选了7本计算机科学方面最新版本的教材,进行影印出版。Prentice Hall 公司和清华大学出版社这次合作将国际先进水平的教材引入我国高等学校,为师生们提供了教学用书,相信会对高校教材改革产生积极的影响。

清华大学出版社  
Prentice Hall 公司

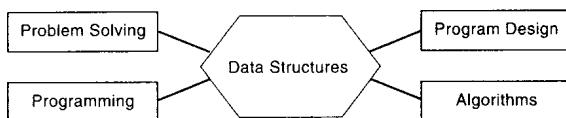
1996.11

---

## 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 heterogeneous 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

Preface xvii

## CHAPTER 1 INTRODUCTION 1

- 1.1 Abstract Data Types 2
  - ADT Format 3
- 1.2 C++ Classes and Abstract Types 6
  - Encapsulation and Information Hiding 7
  - Message Passing 7
- 1.3 Objects in C++ Applications 8
  - Application: The Circle Class 8
- 1.4 Object Design 11
  - Objects and Composition 11
  - C++ Geometric Classes 13
  - Objects and Inheritance 14
  - Inheritance in Programming 15
  - Ordered Lists and Inheritance 18
  - Software Reusability 19
  - SeqList and OrderedList Class Specifications 19
- 1.5 Applications with Class Inheritance 21
- 1.6 Object-Oriented Program Design 22
  - Problem Analysis/Program Definition 23
  - Design 23
  - Coding 24
  - Testing 24
  - Program Design Illustration: A Dice Graph 24
- 1.7 Program Testing and Maintenance 31
  - Object Testing 31
  - Control Module Testing 31
  - Program Maintenance and Documentation 32
- 1.8 The C++ Programming Language 32
- 1.9 Abstract Base Classes and Polymorphism 33
  - Polymorphism and Dynamic Binding 34
- Written Exercises 36

## CHAPTER 2 BASIC DATA TYPES 38

- 2.1 Integer Types 39
  - Computer Storage of Integers 41
  - Data in Memory 42
  - C++ Representation of Integers 43
- 2.2 Character Types 43
  - ASCII Characters 44
- 2.3 Real Data Types 45
  - Real Number Representations 46
- 2.4 Enumerated Types 48
  - Implementing C++ Enumerated Types 49
- 2.5 Pointers 49
  - Pointer ADT 49
  - Pointer Values 51
- 2.6 The Array Type 52
  - The Built-In C++ Array Type 52
  - Storage of One-Dimensional Arrays 53
  - Array Bounds 54
  - Two-Dimensional Arrays 55
  - Storage of Two-Dimensional Arrays 57
- 2.7 String Literals and Variables 58
  - C++ Strings 61
  - Application: Reversing Names 63
- 2.8 Records 65
  - C++ Structures 66
- 2.9 Files 66
  - C++ Stream Hierarchy 69
- 2.10 Array and Record Applications 72
  - Sequential Search 72
  - Exchange Sort 75
  - Counting C++ Reserved Words 77
- Written Exercises 80
- Programming Exercises 88

## CHAPTER 3 ABSTRACT DATA TYPES AND CLASSES 91

- 3.1 The User Type CLASS 92
  - Class Declaration 92
  - Constructor 94
  - Object Declaration 94
  - Class Implementation 95
  - Implementing a Constructor 96
  - Building Objects 97

3.2	Sample Classes	101
	The Temperature Class	101
	The Random Number Class	104
3.3	Objects and Information Passing	110
	An Object as a Return Value	110
	An Object as a Function Parameter	110
3.4	Arrays of Objects	111
	The Default Constructor	112
3.5	Multiple Constructors	113
3.6	Case Study: Triangular Matrices	116
	Upper Triangular Matrix Properties	117
	Written Exercises	126
	Programming Exercises	131

## **CHAPTER 4 COLLECTION CLASSES 141**

4.1	Describing Linear Collections	144
	Direct Access Collections	145
	Sequential Access Collections	146
	Generalized Indexing	150
4.2	Describing Nonlinear Collections	151
	Group Collections	152
4.3	Analysis of Algorithms	154
	Performance Criteria	154
	Common Orders of Magnitude	159
4.4	The Sequential and Binary Search	160
	Binary Search	161
4.5	The Basic Sequential List Class	167
	List Modification Methods	170
	Written Exercises	178
	Programming Exercises	181

## **CHAPTER 5 STACKS AND QUEUES 184**

5.1	Stacks	185
5.2	The Stack Class	188
5.3	Expression Evaluation	197
	Postfix Evaluation	198
	Application: A Postfix Calculator	199
5.4	Queues	204
5.5	The Queue Class	207
5.6	Priority Queues	221
	A Priority Queue Class	223
5.7	Case Study: Event-Driven Simulation	231

Written Exercises	245
Programming Exercises	249

## **CHAPTER 6 ABSTRACT OPERATORS 253**

6.1	Describing Operator Overloading	255
	Client-Defined External Functions	255
	Class Members	256
	Friend Functions	259
6.2	Rational Number System	260
	Representing Rational Numbers	261
	Rational Number Arithmetic	261
	Rational Number Conversion	262
6.3	The Rational Class	263
6.4	Rational Operators as Member Functions	265
	Implementing the Rational Operators	266
6.5	The Rational Stream Operators as Friends	267
	Implementing Rational Stream Operators	268
6.6	Converting Rational Numbers	269
	Conversion to Object Type	269
	Conversion from Object Type	271
6.7	Using Rational Numbers	272
	Written Exercises	277
	Programming Exercises	284

## **CHAPTER 7 GENERIC DATA TYPES 289**

7.1	Template Functions	290
	Template-Based Sort	294
7.2	Template Classes	294
	Defining a Template Class	294
	Declaring Template Class Objects	295
	Defining Template Class Methods	295
7.3	Template List Classes	297
7.4	Infix Expression Evaluation	299
	Written Exercises	308
	Programming Exercises	309

## **CHAPTER 8 CLASSES AND DYNAMIC MEMORY 313**

8.1	Pointers and Dynamic Data Structures	315
	The Memory Allocation Operator New	315

	Dynamic Array Allocation	316
	The Memory Deallocation Operator Delete	317
8.2	Dynamically Allocated Objects	318
	Deallocating Object Data: The Destructor	319
8.3	Assignment and Initialization	322
	Assignment Issues	322
	Overloading the Assignment Operator	324
	The This Pointer	325
	Initialization Issues	325
	Creating a Copy Constructor	326
8.4	Safe Arrays	329
	The Array Class	329
	Memory Allocation for the Array Class	331
	Array Bounds Checking and the Overloaded [] Operator	332
	Converting an Object to a Pointer	333
	Using the Array Class	335
8.5	A String Class	337
	String Class Implementation	343
8.6	Pattern Matching	349
	The Find Process	350
	Pattern Matching Algorithm	350
	Analysis of the Pattern Matching Algorithm	355
8.7	Integral Sets	356
	Sets of Integral Types	356
	C++ Bit Handling Operators	357
	Representing Set Elements	360
	The Sieve of Eratosthenes	363
	Set Class Implementation	366
	Written Exercises	369
	Programming Exercises	379

## **CHAPTER 9 LINKED LISTS 383**

	Describing a Linked List	386
	Chapter Overview	386
9.1	The Node Class	387
	Declaring a Node Type	387
	Implementing the Node Class	390
9.2	Building Linked Lists	393
	Creating a Node	393
	Inserting a Node: InsertFront	393
	Traversing a Linked List	394
	Inserting a Node: InsertRear	397
	Application: Student Graduation List	401

	Creating an Ordered List	404
	Application: Sorting with Linked Lists	406
9.3	Designing a Linked List Class	409
	Linked List Data Members	409
	Linked List Operations	410
9.4	The LinkedList Class	413
9.5	Implementing the LinkedList Class	421
9.6	Implementing Collections with Linked Lists	429
	Linked Queues	429
	Implementing Queue Methods	431
	Linked SeqList Class	432
	Implementing SeqList Data Access Methods	433
	Application: Comparing SeqList Implementations	434
9.7	Case Study: A Print Spooler	436
	Implementing the Spooler Update Method	439
	Spooler Evaluation Methods	440
9.8	Circular Lists	443
	Circular Node Class Implementation	445
	Application: Solving the Josephus Problem	447
9.9	Doubly Linked Lists	450
	Application: Doubly Linked List Sort	452
	DNode Class Implementation	455
9.10	Case Study: Window Management	457
	The Window List	458
	WindowList Class Implementation	461
	Written Exercises	465
	Programming Exercises	474

## CHAPTER 10 RECURSION 480

10.1	The Concept of Recursion	481
	Recursive Definitions	483
	Recursive Problems	484
10.2	Designing Recursive Functions	489
10.3	Recursive Code and the Runtime Stack	494
	The Runtime Stack	494
10.4	Problem-Solving with Recursion	497
	Binary Search	497
	Combinatorics: The Committee Problem	500
	Combinatorics: Permutations	503
	Maze Handling	514
	Maze Class Implementation	518
10.5	Evaluating Recursion	521



Written Exercises	527
Programming Exercises	530

## **CHAPTER 11 TREES 533**

Tree Terminology	535
Binary Trees	537
11.1 Binary Tree Structure	540
Designing a TreeNode Class	540
Building a Binary Tree	543
11.2 Designing TreeNode Functions	544
Recursive Tree Traversals	546
11.3 Using Tree Scan Algorithms	550
Application: Visiting Tree Nodes	550
Application: Tree Print	552
Application: Copying and Deleting Trees	553
Application: Upright Tree Printing	559
11.4 Binary Search Trees	564
The Key in a Binary Search Tree Node	566
Operations on a Binary Search Tree	567
Declaring a Binary Search Tree ADT	568
11.5 Using Binary Search Trees	573
Duplicate Nodes	575
11.6 The BinSTree Implementation	578
List Operations	579
11.7 Case Study: Concordance	590
Written Exercises	596
Programming Exercises	602

## **CHAPTER 12 INHERITANCE AND ABSTRACT CLASSES 606**

12.1 A View of Inheritance	607
Class Inheritance Terminology	609
12.2 Inheritance in C++	610
Constructors and Derived Classes	612
What Cannot Be Inherited	619
12.3 Polymorphism and Virtual Functions	620
Demonstrating Polymorphism	622
Application: Geometric Figures and Virtual Methods	626
Virtual Methods and the Destructor	629
12.4 Abstract Base Classes	630