教育部高等教育司推荐
国外优秀信息科学与技术系列教学用书

数据结构与程序设计

——C++语言描述

（影印版）

DATA STRUCTURES AND PROGRAM DESIGN IN C++

Robert L. Kruse
Alexander J. Ryba

高等教育出版社
Higher Education Press
Pearson Education
出版集团

教育部高等教育司推荐
国外优秀信息科学与技术系列教学用书

数据结构与程序设计

——C++语言描述

（影印版）

DATA STRUCTURES AND PROGRAM DESIGN IN C++

Robert L. Kruse
Alexander J. Ryba

高等教育出版社
Pearson Education 出版集团

图字： 01-2001-1043 号

Data Structures and Program Design in C++

By Robert L . Kruse & Alexander J. Ryba

图书在版编目(CIP)数据

数据结构与程序设计：C++语言描述：英文/(美)克鲁斯(Kruse，R.L.)等.—北京：高等教育出版社，2001(2002 重印)
ISBN 7-04-010039-8

Ⅰ.数… Ⅱ.克… Ⅲ.① 数据结构—英文 ② C 语言—程序设计—英文 Ⅳ.TP311.12

中国版本图书馆 CIP 数据核字(2001)第 19797 号

数据结构与程序设计——C++语言描述
Robert L .Kruse 等

出版发行	高等教育出版社	**购书热线**	010-64054588
社　　址	北京市东城区沙滩后街 55 号	**免费咨询**	800-810-0598
邮政编码	100009	**网　　址**	http://www.hep.edu.cn
传　　真	010-64014048		http://www.hep.com.cn
经　　销	新华书店北京发行所		
印　　刷	北京民族印刷厂		
开　　本	787×1092　1/16	**版　　次**	2001 年 5 月第 1 版
印　　张	46.00	**印　　次**	2002 年 10 月第 3 次印刷
字　　数	1 026 000	**定　　价**	39.00 元

本书如有缺页、倒页、脱页等质量问题，请到所购图书销售部门联系调换。

前　言

20 世纪末，以计算机和通信技术为代表的信息科学和技术，对世界的经济、军事、科技、教育、文化、卫生等方面的发展产生了深刻的影响，由此而兴起的信息产业已经成为世界经济发展的支柱。进入 21 世纪，各国为了加快本国的信息产业，加大了资金投入和政策扶持。

为了加快我国信息产业的进程，在我国《国民经济和社会发展第十个五年计划纲要》中，明确提出“以信息化带动工业化，发挥后发优势，实现社会生产力的跨越式发展。”信息产业的国际竞争将日趋激烈。在我国加入 WTO 后，我国信息产业将面临国外竞争对手的严峻挑战。竞争成败最终将取决于信息科学和技术人才的多少与优劣。

在 20 世纪末，我国信息产业虽然得到迅猛发展，但与国际先进国家相比，差距还很大。为了赶上并超过国际先进水平，我国必须加快信息技术人才的培养，特别要培养一大批具有国际竞争能力的高水平的信息技术人才，促进我国信息产业和国家信息化水平的全面提高。为此，教育部高等教育司根据教育部吕福源副部长的意见，在长期重视推动高等学校信息科学和技术的教学的基础上，将实施超前发展战略，采取一些重要举措，加快推动高等学校的信息科学和技术等相关专业的教学工作。在大力宣传、推荐我国专家编著的面向 21 世纪和“九五”重点的信息科学和技术课程教材的基础上，在有条件的高等学校的某些信息科学和技术课程中推动使用国外优秀教材的影印版进行英语或双语教学，以缩短我国在计算机教学上与国际先进水平的差距，同时也有助于强化我国大学生的英语水平。

为了达到上述目的，在分析一些出版社已影印相关教材，一些学校已试用影印教材进行教学的基础上，教育部高等教育司组织并委托高等教育出版社开展国外优秀信息科学和技术优秀教材及其教学辅助材料的引进研究与影印出版的试点工作。为推动用影印版教材进行教学创造条件。

本次引进的系列教材的影印出版工作，是在对我国高校的信息科学和技术专业的课程与美国高校的进行对比分析的基础上展开的；所影印出版的教材均由我国主要高

校的信息科学和技术专家组成的专家组，从国外近两年出版的大量最新教材中精心筛选评审通过的内容新、有影响的优秀教材；影印教材的定价原则上应与我国大学教材价格相当。

教育部高等教育司将此影印系列教材推荐给高等学校，希望有关教师选用，使用后有什么意见和建议请及时反馈。也希望有条件的出版社，根据影印教材的要求，积极参加此项工作，以便引进更多、更新、更好的外国教材和教学辅助材料。

同时，感谢国外有关出版公司对此项引进工作的配合，欢迎更多的国外公司关心并参与此项工作。

教育部高等教育司

二〇〇一年四月

Preface

THE APPRENTICE CARPENTER may want only a hammer and a saw, but a master builder employs many precision tools. Computer programming likewise requires sophisticated tools to cope with the complexity of real applications, and only practice with these tools will build skill in their use. This book treats structured problem solving, object-oriented programming, data abstraction, and the comparative analysis of algorithms as fundamental tools of program design. Several case studies of substantial size are worked out in detail, to show how all the tools are used together to build complete programs.

Many of the algorithms and data structures we study possess an intrinsic elegance, a simplicity that cloaks the range and power of their applicability. Before long the student discovers that vast improvements can be made over the naïve methods usually used in introductory courses. Yet this elegance of method is tempered with uncertainty. The student soon finds that it can be far from obvious which of several approaches will prove best in particular applications. Hence comes an early opportunity to introduce truly difficult problems of both intrinsic interest and practical importance and to exhibit the applicability of mathematical methods to algorithm verification and analysis.

Many students find difficulty in translating abstract ideas into practice. This book, therefore, takes special care in the formulation of ideas into algorithms and in the refinement of algorithms into concrete programs that can be applied to practical problems. The process of data specification and abstraction, similarly, comes before the selection of data structures and their implementations.

We believe in progressing from the concrete to the abstract, in the careful development of motivating examples, followed by the presentation of ideas in a more general form. At an early stage of their careers most students need reinforcement from seeing the immediate application of the ideas that they study, and they require the practice of writing and running programs to illustrate each important concept that they learn. This book therefore contains many sample programs, both short

functions and complete programs of substantial length. The exercises and programming projects, moreover, constitute an indispensable part of the book. Many of these are immediate applications of the topic under study, often requesting that programs be written and run, so that algorithms may be tested and compared. Some are larger projects, and a few are suitable for use by a small group of students working together.

Our programs are written in the popular object-oriented language C++. We take the view that many object-oriented techniques provide natural implementations for basic principles of data-structure design. In this way, C++ allows us to construct safe, efficient, and simple implementations of data-structures. We recognize that C++ is sufficiently complex that students will need to use the experience of a data structures courses to develop and refine their understanding of the language. We strive to support this development by carefully introducing and explaining various object-oriented features of C++ as we progress through the book. Thus, we begin Chapter 1 assuming that the reader is comfortable with the elementary parts of C++ (essentially, with the C subset), and gradually we add in such object-oriented elements of C++ as classes, methods, constructors, inheritance, dynamic memory management, destructors, copy constructors, overloaded functions and operations, templates, virtual functions, and the STL. Of course, our primary focus is on the data structures themselves, and therefore students with relatively little familiarity with C++ will need to supplement this text with a C++ programming text.

SYNOPSIS

Programming Principles

By working through the first large project (CONWAY's game of Life), Chapter 1 expounds principles of object-oriented program design, top-down refinement, review, and testing, principles that the student will see demonstrated and is expected to follow throughout the sequel. At the same time, this project provides an opportunity for the student to review the syntax of elementary features of C++, the programming language used throughout the book.

Introduction to Stacks

Chapter 2 introduces the first data structure we study, the stack. The chapter applies stacks to the development of programs for reversing input, for modelling a desk calculator, and for checking the nesting of brackets. We begin by utilizing the STL stack implementation, and later develop and use our own stack implementation. A major goal of Chapter 2 is to bring the student to appreciate the ideas behind information hiding, encapsulation and data abstraction and to apply methods of top-down design to data as well as to algorithms. The chapter closes with an introduction to abstract data types.

Queues

Queues are the central topic of Chapter 3. The chapter expounds several different implementations of the abstract data type and develops a large application program showing the relative advantages of different implementations. In this chapter we introduce the important object-oriented technique of inheritance.

Linked Stacks and Queues

Chapter 4 presents linked implementations of stacks and queues. The chapter begins with a thorough introduction to pointers and dynamic memory management in C++. After exhibiting a simple linked stack implementation, we discuss

destructors, copy constructors, and overloaded assignment operators, all of which are needed in the safe C++ implementation of linked structures.

Recursion

Chapter 5 continues to elucidate stacks by studying their relationship to problem solving and programming with recursion. These ideas are reinforced by exploring several substantial applications of recursion, including backtracking and tree-structured programs. This chapter can, if desired, be studied earlier in a course than its placement in the book, at any time after the completion of Chapter 2.

Lists and Strings

More general lists with their linked and contiguous implementations provide the theme for Chapter 6. The chapter also includes an encapsulated string implementation, an introduction to C++ templates, and an introduction to algorithm analysis in a very informal way.

Searching

Sorting

Chapter 7, Chapter 8, and Chapter 9 present algorithms for searching, sorting, and table access (including hashing), respectively. These chapters illustrate the interplay between algorithms and the associated abstract data types, data structures, and implementations. The text introduces the "big-O" and related notations for elementary algorithm analysis and highlights the crucial choices to be made regarding best use of space, time, and programming effort. These choices require that we find analytical methods to assess algorithms, and producing such analyses is a battle for which combinatorial mathematics must provide the arsenal. At an elementary level we can expect students neither to be well armed nor to possess the mathematical maturity needed to hone their skills to perfection. Our goal, therefore, is to help students recognize the importance of such skills in anticipation of later chances to study mathematics.

Tables and Information Retrieval

Binary Trees

Binary trees are surely among the most elegant and useful of data structures. Their study, which occupies Chapter 10, ties together concepts from lists, searching, and sorting. As recursively defined data structures, binary trees afford an excellent opportunity for the student to become comfortable with recursion applied both to data structures and algorithms. The chapter begins with elementary topics and progresses as far as such advanced topics as splay trees and amortized algorithm analysis.

Multiway Trees

Chapter 11 continues the study of more sophisticated data structures, including tries, B-trees, and red-black trees.

Graphs

Chapter 12 introduces graphs as more general structures useful for problem solving, and introduces some of the classical algorithms for shortest paths and minimal spanning trees in graphs.

Case Study: The Polish Notation

The case study in Chapter 13 examines the Polish notation in considerable detail, exploring the interplay of recursion, trees, and stacks as vehicles for problem solving and algorithm development. Some of the questions addressed can serve as an informal introduction to compiler design. As usual, the algorithms are fully developed within a functioning C++ program. This program accepts as input an expression in ordinary (infix) form, translates the expression into postfix form, and evaluates the expression for specified values of the variable(s). Chapter 13 may be studied anytime after the completion of Section 10.1.

The appendices discuss several topics that are not properly part of the book's subject but that are often missing from the student's preparation.

Mathematical Methods

Appendix A presents several topics from discrete mathematics. Its final two sections, Fibonacci numbers amd Catalan numbers, are more advanced and not

needed for any vital purpose in the text, but are included to encourage combinatorial interest in the more mathematically inclined.

Random Numbers

Appendix B discusses pseudorandom numbers, generators, and applications, a topic that many students find interesting, but which often does not fit anywhere in the curriculum.

Packages and Utility Functions

Appendix C catalogues the various utility and data-structure packages that are developed and used many times throughout this book. Appendix C discusses declaration and definition files, translation units, the utility package used throughout the book, and a package for calculating CPU times.

Programming Precepts, Pointers, and Pitfalls

Appendix D, finally, collects all the Programming Precepts and all the Pointers and Pitfalls scattered through the book and organizes them by subject for convenience of reference.

COURSE STRUCTURE

prerequisite

The prerequisite for this book is a first course in programming, with experience using the elementary features of C++. However, since we are careful to introduce sophisticated C++ techniques only gradually, we believe that, used in conjunction with a supplementary C++ textbook and extra instruction and emphasis on C++ language issues, this text provides a data structures course in C++ that remains suitable even for students whose programming background is in another language such as C, Pascal, or Java.

A good knowledge of high school mathematics will suffice for almost all the algorithm analyses, but further (perhaps concurrent) preparation in discrete mathematics will prove valuable. Appendix A reviews all required mathematics.

content

This book is intended for courses such as the ACM Course CS2 (*Program Design and Implementation*), ACM Course CS7 (*Data Structures and Algorithm Analysis*), or a course combining these. Thorough coverage is given to most of the ACM/IEEE knowledge units[1] on data structures and algorithms. These include:

AL1 Basic data structures, such as arrays, tables, stacks, queues, trees, and graphs;

AL2 Abstract data types;

AL3 Recursion and recursive algorithms;

AL4 Complexity analysis using the big Oh notation;

AL6 Sorting and searching; and

AL8 Practical problem-solving strategies, with large case studies.

The three most advanced knowledge units, AL5 (complexity classes, NP-complete problems), AL7 (computability and undecidability), and AL9 (parallel and distributed algorithms) are not treated in this book.

[1] See *Computing Curricula 1991: Report of the ACM/IEEE-CS Joint Curriculum Task Force*, ACM Press, New York, 1990.

Most chapters of this book are structured so that the core topics are presented first, followed by examples, applications, and larger case studies. Hence, if time allows only a brief study of a topic, it is possible, with no loss of continuity, to move rapidly from chapter to chapter covering only the core topics. When time permits, however, both students and instructor will enjoy the occasional excursion into the supplementary topics and worked-out projects.

two-term course

A two-term course can cover nearly the entire book, thereby attaining a satisfying integration of many topics from the areas of problem solving, data structures, program development, and algorithm analysis. Students need time and practice to understand general methods. By combining the studies of data abstraction, data structures, and algorithms with their implementations in projects of realistic size, an integrated course can build a solid foundation on which, later, more theoretical courses can be built. Even if it is not covered in its entirety, this book will provide enough depth to enable interested students to continue using it as a reference in later work. It is important in any case to assign major programming projects and to allow adequate time for their completion.

SUPPLEMENTARY MATERIALS

A CD-ROM version of this book is anticipated that, in addition to the entire contents of the book, will include:

- All packages, programs, and other C++ code segments from the text, in a form ready to incorporate as needed into other programs;
- Executable versions (for DOS or Windows) of several demonstration programs and nearly all programming projects from the text;
- Brief outlines or summaries of each section of the text, suitable for use as a study guide.

These materials will also be available from the publisher's internet site. To reach these files with `ftp`, log in as user `anonymous` to the site `ftp.prenhall.com` and change to the directory

```
pub/esm/computer_science.s-041/kruse/cpp
```

Instructors teaching from this book may obtain, at no charge, an instructor's version on CD-ROM which, in addition to all the foregoing materials, includes:

- Brief teaching notes on each chapter;
- Full solutions to nearly all exercises in the textbook;
- Full source code to nearly all programming projects in the textbook;
- Transparency masters.

BOOK PRODUCTION

This book and its supplements were written and produced with software called PreTEX, a preprocessor and macro package for the TEX typesetting system.[2] PreTEX, by exploiting context dependency, automatically supplies much of the typesetting markup required by TEX. PreTEX also supplies several tools that greatly simplify some aspects of an author's work. These tools include a powerful cross-reference system, simplified typesetting of mathematics and computer-program listings, and automatic generation of the index and table of contents, while allowing the processing of the book in conveniently small files at every stage. Solutions, placed with exercises and projects, are automatically removed from the text and placed in a separate document.

For a book such as this, PreTEX's treatment of computer programs is its most important feature. Computer programs are not included with the main body of the text; instead, they are placed in separate, secondary files, along with any desired explanatory text, and with any desired typesetting markup in place. By placing tags at appropriate places in the secondary files, PreTEX can extract arbitrary parts of a secondary file, in any desired order, for typesetting with the text. Another utility removes all the tags, text, and markup, producing as its output a program ready to be compiled. The same input file thus automatically produces both typeset program listings and compiled program code. In this way, the reader gains increased confidence in the accuracy of the computer program listings appearing in the text. In fact, with just two exceptions, all of the programs developed in this book have been compiled and succesfully tested under the g++ and Borland C++ compilers (versions 2.7.2.1 and 5.0, respectively). The two exceptions are the first program in Chapter 2 (which requires a compiler with a full ANSI C++ standard library) and the last program of Chapter 13 (which requires a compiler with certain Borland graphics routines).

ACKNOWLEDGMENTS

Over the years, the Pascal and C antecedents of this book have benefitted greatly from the contributions of many people: family, friends, colleagues, and students, some of whom are noted in the previous books. Many other people, while studying these books or their translations into various languages, have kindly forwarded their comments and suggestions, all of which have helped to make this a better book.

We are happy to acknowledge the suggestions of the following reviewers, who have helped in many ways to improve the presentation in this book: KEITH VANDER LINDEN (Calvin College), JENS GREGOR (University of Tennessee), VICTOR BERRY (Boston University), JEFFERY LEON (University of Illinois at Chicago), SUSAN

[2] TEX was developed by DONALD E. KNUTH, who has also made many important research contributions to data structures and algorithms. (See the entries under his name in the index.)

HUTT (University of Missouri–Columbia), FRED HARRIS (University of Nevada), ZHI-LI ZHANG (University of Minnesota), and ANDREW SUNG (New Mexico Institute of Technology).

ALEX RYBA especially acknowledges the helpful suggestions and encouraging advice he has received over the years from WIM RUITENBURG and JOHN SIMMS of Marquette University, as well as comments from former students RICK VOGEL and JUN WANG.

It is a special pleasure for ROBERT KRUSE to acknowledge the continuing advice and help of PAUL MAILHOT of PreTEX, Inc., who was from the first an outstanding student, then worked as a dependable research assistant, and who has now become a valued colleague making substantial contributions in software development for book production, in project management, in problem solving for the publisher, the printer, and the authors, and in providing advice and encouragement in all aspects of this work. The CD-ROM versions of this book, with all their hypertext features (such as extensive cross-reference links and execution of demonstration programs from the text), are entirely his accomplishment.

Without the continuing enthusiastic support, faithful encouragement, and patience of the editorial staff of Prentice Hall, especially ALAN APT, Publisher, LAURA STEELE, Acquisitions Editor, and MARCIA HORTON, Editor in Chief, this project would never have been started and certainly could never have been brought to completion. Their help, as well as that of the production staff named on the copyright page, has been invaluable.

ROBERT L. KRUSE
ALEXANDER J. RYBA

Contents