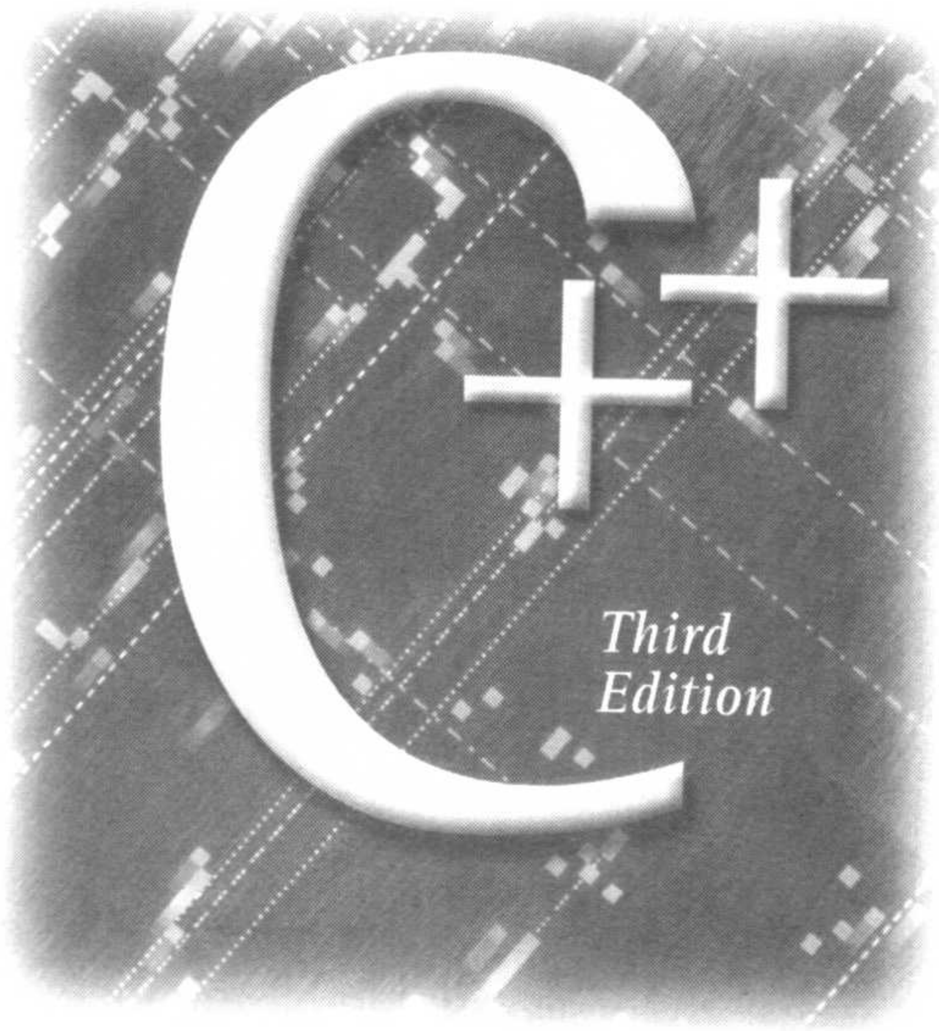
Problem Solving, Abstraction, & Design Using C++

Third Edition

Frank L. Friedman
Elliot B. Koffman

Problem Solving, Abstraction, Design Using



Frank L. Friedman
Temple University

Elliot B. Koffman
Temple University

Addison Wesley Longman

Reading, Massachusetts • Menlo Park, California
New York • Harlow, England • Don Mills, Ontario
Sydney • Mexico City • Madrid • Amsterdam

Senior Acquisitions Editor:	Susan Hartman
Assistant Editor:	Elinor Actipis
Project Managers:	The Publisher's Group
	Trillium Project Management
Executive Marketing Manager:	Michael Hirsch
Composition:	Michael and Sigrid Wile
Text Design:	Delgado Design, Inc.
Design Supervisor:	Gina Hagen
Copyeditor:	Stephanie Magean
Technical Artist:	Delgado Design, Inc.
Proofreading:	Trillium Project Management
Cover Design:	Gina Hagen

Library of Congress Cataloging-in-Publication Data

Friedman, Frank L.

Problem solving, abstraction, and design using C++ / Frank L. Friedman, Elliot B. Koffman. —3rd ed.

p. cm.

Includes bibliographical references and index.

ISBN 0–201–61277–1

1. C++ (Computer program language) I. Koffman, Elliot B. II. Title.

QA76.73.C153F75 2000

005.13'13—dc21 99–055315

CIP

Cover image © Steven Hunt / The Image Bank / PNI

Access the latest information about Addison-Wesley titles from our World Wide Web site: http://www.awlonline.com

This book was typeset in Quark 4.1 on a Macintosh G4. The font families used were Berkeley and Rotis. It was printed on New Era Matte.

1 2 3 4 5 6 7 8 9 10–MA–03020100

To my wife, Martha
my children, Dara and Shelly
and my parents, George and Sylvia
you've made this, and everything else, possible.

FLF

To my wife, Caryn
my children, Richard, Deborah, and Robin
with much thanks for your love and support.

EBK

Preface

This is a textbook for a one- or two-semester course in problem solving and program design. It is suitable for use by students with no programming background as well as those who may have had the equivalent of up to a one-semester course in another programming language.

The earlier editions of this book represented the culmination of an eight-year effort, partially sponsored by the National Science Foundation,[1] to define an introductory-level course combining the presentation of rudimentary principles of software engineering and object-oriented programming based on the C++ programming language. Our primary goal is to motivate and introduce sound principles of program design and abstraction in a first programming course. Topics such as program style, documentation, algorithm and data structuring, procedure- and data-oriented modularization, component reuse, and program testing are introduced early. The focus throughout is on the problem solving/software design process, from problem analysis to program design and coding.

Features of the Third Edition

The third edition has been revised to be totally compatible with the ANSI standard for C++. It introduces data type bool and the string class in Chapter 2. There is a new appendix (Appendix H) on standard containers and iterators that introduces vectors, stacks, queues, and lists. The section on vectors could

[1]NSF Instrumentation and Laboratory Improvement (ILI) Grant Number USE-9250254 and NSF Undergraduate Curriculum Course Development (UCCD) Grant Number USE-9156079.

be introduced right after arrays and subscripts (Section 9.2) if desired. There are also new appendices on two popular Integrated Development Environments (IDEs): Microsoft Visual C++ and Borland C++ Builder.

We have improved the writing style in this new edition and simplified the presentation whenever possible. The new design and the use of color also help to make the text more accessible to students.

The second edition contained a separate chapter on Software Engineering. Most of the material in this chapter has been distributed throughout the book where appropriate. The rest of the chapters follow the same order as in the second edition.

Balancing Object-Oriented and Procedural Approaches

Object-oriented concepts and the use of classes are introduced early in the book starting in Chapter 1. Chapters 2 and 3 discuss the use of two standard classes, `iostream` and `string`, and we refer to the use of classes and objects throughout most of the text. We also introduce a user-defined money class in Chapter 3.

We have tried to follow a balanced path between the strictly objects-first and totally procedure-focused programming metaphors. We agree with the objects-first concept, but not at the expense of the fundamentals of algorithm organization and design. Students in a first course can and should be taught the basic elements of procedural design. Our task is to do so within the context of an early focus on the importance of data modeling, reuse, and other fundamental principles of good software development.

An issue of concern to faculty is the relative order of arrays, structs, and classes. As in the last edition, we introduce arrays and structs first (Chapter 9) and then introduce the definition and coding of classes (Chapter 10). Some faculty may prefer to reverse the order and this is entirely possible. The chapter on classes uses arrays only in the implementation of class `simpleString` which can be omitted or deferred until after arrays are covered.

We continue to emphasize the design of classes and data modeling in Chapter 11, which introduces template classes, an indexed-list class, a stack class, friend functions, and operator overloading. We also use template classes in Chapter 13 where we discuss dynamic data structures: lists, stacks, queues, and trees. Appendix E discusses inheritance and virtual functions. Appendix H describes the use of the standard container classes: vector, list, stack, and queue.

Software Engineering and Object-Oriented Concepts

Many fundamental software engineering and object-oriented concepts are illustrated in the text: user-defined types, modeling problem domain entities and their relationships, minimal interfaces, high-level cohesion, information hiding, separation of concerns, parameterized components, and inheritance. Abstraction is stressed from the start. Numerous complete case studies are provided throughout the text, which follow a standard software development method, from the specification and analysis of a problem to the first stage of design to the final coding.

Issues of program style are presented throughout in special displays. The concept of a program as a sequence of control structures is discussed in Chapters 4 (on selection structures) and 5 (repetition structures). We introduced functions and classes as early as possible at the introductory level—functions in Chapters 3 and 6, and the use and definition of classes in Chapters 3 and 10 respectively. We also provide several sections that discuss testing and debugging.

Outline of Contents

Conceptually, the text may be partitioned into three sections. Chapters 1 through 6 provide introductory material on functions and top-down design and detailed coverage of selection and repetition structures. The connection between good problem-solving skills and effective software development is established early in the first three chapters. The problem-solving approach outlined in these chapters is applied consistently to all other case studies in the text. Chapter 2 also contains an introduction to the basic elements of C++, including a section on data types and abstraction. In Chapter 3, we continue the emphasis on basic problem-solving skills with a discussion of top-down design. The reuse of program components is discussed and additional detail is provided on the `money` and `string` classes and their member functions.

Top-down procedural decomposition is further illustrated throughout Chapters 4 through 6. Decision structures are introduced in Chapter 4, and repetition structures are presented in Chapter 5. In Chapter 6, we revisit the C++ function, introducing functions with output arguments and providing a complete case study illustrating much of what has been learned to this point. An optional section on recursion is also included at the end of Chapter 6.

Chapters 7 through 9 cover simple data types, input and output, and structured data types (arrays and structs). Chapter 7 contains a more detailed

discussion of simple data types, including additional commentary on data abstraction as well as a description of the internal and external distinctions among the simple types. In Chapter 9, the structured types (arrays and structs) are first introduced. Simple searching and sorting algorithms are discussed and the use of structured types as function arguments is illustrated.

Chapter 8 provides an introduction to external file input/output. Although studying external files may seem premature at this point, we believe it is appropriate. Programs do not exist in a vacuum; they manipulate data that often come from external sources and they produce results that may subsequently be manipulated by other programs. It is therefore important for students to gain a relatively early exposure to some fundamental concepts related to file input and output, as long as this exposure does not disrupt the presentation of other essential ideas. Of course, by the time Chapter 8 is reached, students will have already been introduced to the basics of stream input and output, including a minimal use of input/output manipulators (Chapter 5).

For students with the equivalent of a one-semester programming course in another language, Chapters 1 through 9 can be covered fairly quickly, perhaps in as little as five or six weeks. For students with little or no background, this coverage may take ten to twelve weeks.

Chapters 10 and 11 cover intermediate-level concepts which would normally be covered at the end of CS1 or the beginning of CS2. These chapters describe the definition and use of classes and class instances (objects). Chapter 11 focuses on data modeling. We begin with a discussion of multidimensional arrays and arrays of structs and classes, and then extend our modeling capability with illustrations of the use of class templates.

Chapters 12 and 13 cover more advanced topics in some depth: recursion (Chapter 12), and dynamic data structures (linked lists, stacks, queues, and trees) in Chapter 13. This material will be covered in the second semester of the first-year sequence.

Coverage of Pointers

Pointers are introduced only where they really belong—in the discussion of dynamic data structures (Chapter 13). The pointer is one of the more dangerous, relatively unprotected aspects of the C++ language and need not be an essential part of an introductory text. Use of the `new` and `delete` operators and the allocation and deallocation of memory cells in the heap are discussed at the beginning of Chapter 13. We illustrate the manipulation of dynamic data structures such as simple linked lists, stacks and queues, and binary trees.

Pedagogical Features

Several pedagogical features also enhance the usefulness of the text as an instructional tool. These include the following:

- Consistent use of analysis and design aids such as data requirements tables and program structure charts
- End-of-section self-check and programming exercises (answers to the odd numbered self-check exercises are provided in the text)
- End-of-chapter self-check exercises (answers are provided)
- End-of-chapter programming projects
- Numerous examples and case studies carried through from analysis and design to implementation
- Syntax displays containing the syntax and semantics of each new C++ feature introduced
- Program style and design guideline displays
- Detailed syntax and run-time error discussions at the end of each chapter
- Chapter reviews and review questions

Appendices and Special Supplements

Separate appendices are provided, summarizing information about character sets, C++ reserved words, C++ operators, and function libraries (with descriptions and specific section numbers). There is an appendix illustrating inheritance and virtual functions, an appendix on Visual C++, an appendix on Borland C++ Builder, and an appendix on standard containers and iterators.

Additional supplements available to instructors who use this textbook are:

- PowerPoint slides
- Laboratory assignments keyed to the textbook
- Instructor's manual

The instructor's manual includes the following features:

- A statement of objectives for each chapter.
- Answers to even-numbered quick-check exercises
- Answers to review questions
- Commentary on the analysis and design of selected programming projects

To order the IM, please contact your local A-W sales representative.

The following can be obtained electronically through Addison-Wesley's web site: http://www.aw.com/cseng/authors/friedman/probsol3e/probsol3e.html

- All programs, functions, and classes from the text
- Answers to all end-of-section programming exercises
- The implementation of selected programming projects
- Sample exam questions
- A `money` class

Acknowledgements

Many people helped with the development of this book. Primary contributors to the first edition included Paul LaFollette, Paul Wolfgang, and Rajiv Tewari of Temple University. Temple graduate students Donna Chrupcala, Bruce Weiner, and Judith Wilson also contributed significantly to the development of the first edition. Steve Vinoski provided detailed comments concerning the C++ material in many of the later chapters. Robin Koffman contributed significantly to the development of this edition. Jeri Hanly very graciously allowed us to adapt material from *C: Problem Solving and Program Design, 3rd Edition*, co-authored by herself and Elliot Koffman.

The principal reviewers and class testers were enormously helpful in suggesting improvements and finding errors. For the first edition, these included Allen Alexander (Delaware Technical and Community College), Ruth Barton and Richard Reid (Michigan State University), Larry Cottrell (University of Central Florida), H. E. Dunsmore and Russell Quong (Purdue University), Donna Krabbe (College of Mount St. Joseph), Sally Kyvernitis (Neumann College), Xiaoping Jia (DePaul University), Xiannong Meng and Rick Zaccone (Bucknell), Jeff Buckwalter and Kim Summerhays (University of San Francisco), and Jo Ellen Perry (University of North Carolina). Valuable proofreading and editing assistance were provided by Sally Kyvernitis, Donna Skalski, and Frank Friedman's daughters Dara and Shelley.

We are also very grateful to the principal reviewers of the second edition for their hard work and timely responses. They include: William E. Bulley (Merit Network, Inc.), Greg Comeau (Comeau Computing), Bruce Gilland (University of Colorado at Boulder), William I. Grosky (Wayne State University), Bina Ramamurthy (SUNY at Buffalo), and W. Brent Seales (University of Kentucky). Our thanks, also, to Temple student Niv Hartman, who helped proofread the text and helped with the exercise solutions.

We would also like to thank Conrad Weisert (Information Disciplines, Inc.) for permission to use the `money` class and for providing the code for this class for users of the text (see special supplements). Thanks also to Lynn Lambert,

Christopher Newport College, for preparing the laboratory manual as well as to Michael R. Hudock, Muhlenberg College, who prepared the PowerPoint slides.

As always, it has been a pleasure working with the people of Addison-Wesley throughout this endeavor. Susan Hartman, senior acquisitions editor, was closely involved in all phases of the development of the manuscript, and provided friendship, guidance, and encouragement. Elinor Actipis, assistant editor, provided timely assistance at a moment's notice. Amy Rose coordinated the conversion of the manuscript to a finished book, Stephanie Magean thoroughly copyedited the manuscript, Brooke Albright proofread the page proofs, and Mike Wile handled the production of the book.

Philadelphia, PA
E. B. K.
F. L. F.

Contents