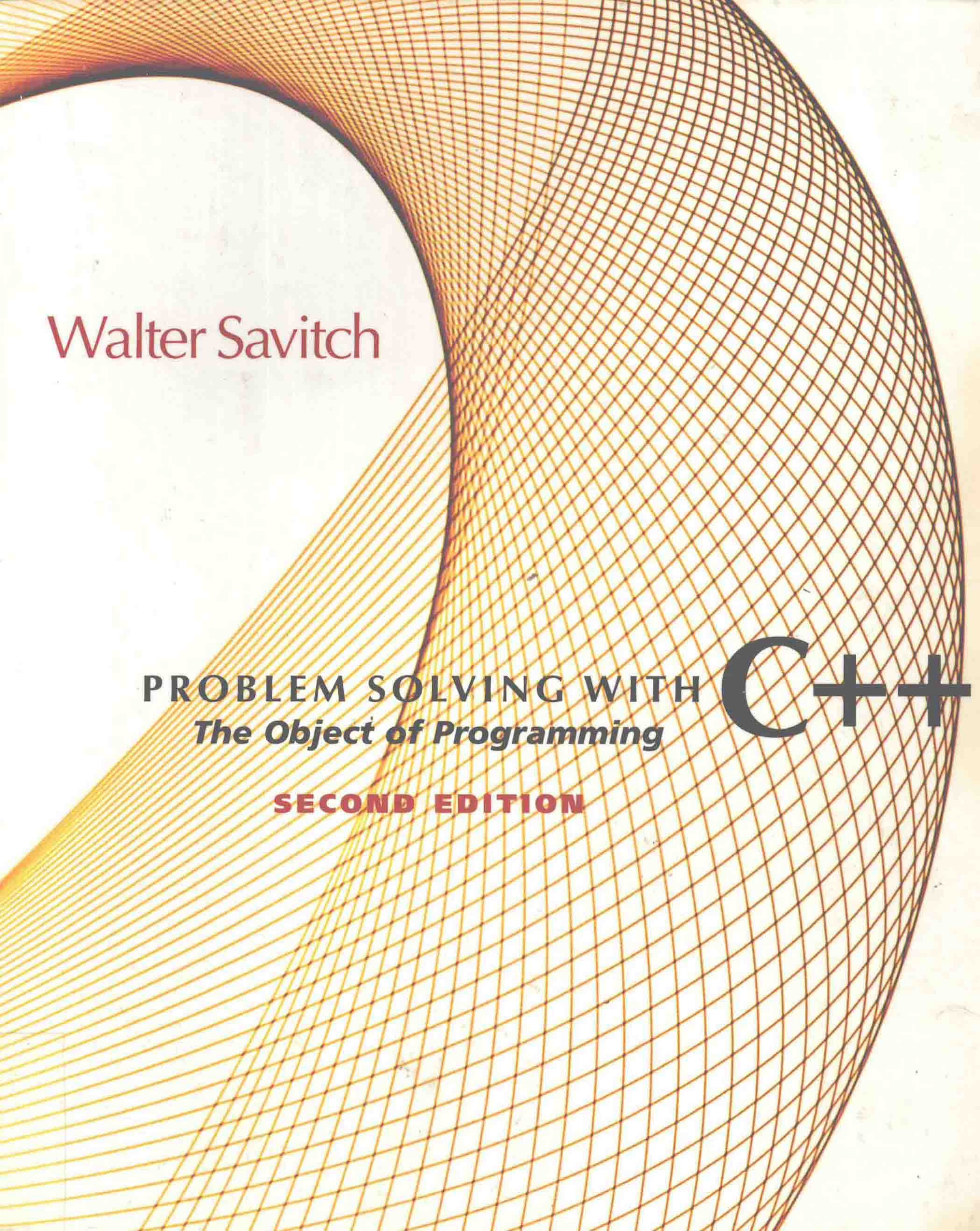
Walter Savitch

PROBLEM SOLVING WITH C++
The Object of Programming

SECOND EDITION



Walter Savitch
University of California at San Diego

PROBLEM SOLVING WITH C++
The Object of Programming

SECOND EDITION

Addison-Wesley
An imprint of Addison Wesley Longman, Inc.

Reading, Massachusetts • Harlow, England • Menlo Park, California
Berkeley, California • Don Mills, Ontario • Sydney • Bonn • Amsterdam
Tokyo • Mexico City

Acquisitions Editor: Susan Hartman
Production Editor: Amy Willcutt
Assistant Editor: Julie Dunn
Production Assistant: Brooke D. Albright
Composition: Michael and Sigrid Wile
Copyeditor: Roberta Lewis
Proofreader: Phyllis Coyne et al.
Cover Design: Arisman Design

Cover image © Akira Inoue/Photonica

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

Library of Congress Cataloging-in-Publication Data

Savitch, Walter J., 1943–
  Problem solving with C++ : the object of programming / Walter
 Savitch. — 2nd ed.
   p. cm.
  Includes index.
  ISBN 0-201-35749-6
  1. C++ (Computer program language) I. Title.
 QA76.73.C153S29 1999
 005.13'3—dc21 98–27261
  CIP

This book was typeset in QuarkXpress 3.32 on a Power Macintosh 7500. The font used was Utopia. It was printed on New Era Matte.

Reprinted with corrections, March 1999.

7 8 9 10-MA-02010099

Preface

This book is meant to be used in a first course in programming and computer science using the C++ language. It assumes no previous programming experience and no mathematics beyond high school algebra. It could also be used as a text for a course designed to teach C++ to students who have already had some other programming course, in which case the first few chapters can be assigned as outside reading.

If you have used the first edition of this book, you should read the following section, which explains the changes to this second edition, and can skip the rest of this preface. If you are new to this book, the rest of this preface will give you an overview of the book.

Changes to the Second Edition

This second edition presents the same programming philosophy and uses the same outline of topics as the first edition. If you are an instructor already using the first edition, you can continue to teach your course almost without change. However, this second edition does offer an opportunity for adding topics to a course. This second edition includes extra exercises, reworking of some sections to improve clarity and correctness, and the introduction of some important new topics. The main additional topics are the use of the type *bool* for Boolean values, the introduction of the string class from the Standard Template Library (STL), and an additional concluding chapter on inheritance. The type *bool* is integrated through the book. If you are using a compiler that does not yet support the type *bool*, then Appendix 10 explains how you can easily simulate the type *bool* on your system. Both the use of the string class from the Standard Template Library and the additional material on inheritance can be considered optional. However, it would be good to consider adding one or both of these topics to your course, if they are not already included.

A Resource—Not a Straightjacket

Most introductory textbooks that use C++ have a very detailed agenda that an instructor must follow in order to use the book in class. If you are an instructor, this book adapts to the way you teach, rather than making you adapt to the book. This book explains C++ and basic programming techniques in a way suitable for beginning students, but it does not tightly prescribe the order in which your course must cover topics and does not prescribe the specialized libraries, if any, that must be used in your course. You can easily change the order in which chapters and sections are covered without loss of continuity in reading the book. The details about rearranging material are explained in the section of this preface on flexibility. Although this book uses libraries and teaches students the importance of libraries, it requires no special libraries. It was designed to be used with a "standard" C++ implementation and only uses libraries that are provided with essentially all C++ implementations. Instructors who wish to do so may use additional libraries of their own choosing.

Early Classes

There are at least two ways that a book can introduce classes early. A book can teach students how to *design* their own classes early in the book or it can merely teach them how to *use* classes early without defining them. This book teaches students to define their own classes early and does not merely teach them how to use classes early. In order to effectively design classes, a student needs some basic tools such as some simple control structures and function definitions. This book thus starts out covering these basics in Chapters 2, 3, and 4. It then moves immediately to classes. In Chapter 5 file I/O streams are used to teach students how to use classes. In Chapter 6 students learn how to write their own classes.

This book uses a measured approach to classes. It teaches students to write some very simple classes, then adds constructors, then overloading simple operators, then overloading the I/O operators << and >>, and so forth. This measured approach keeps the student from being overwhelmed with a long list of complicated constructions and concepts. However, one goal of this book is to get students writing realistic class definitions as soon as possible, and not to have them spending time writing classes that are artificially simple. By the end of Chapter 8, students are writing essentially the same kinds of classes that they will be writing when they finish the course.

There are a few topics related to classes that are not introduced at the beginning. Destructors, templates, and sophisticated use of inheritance are not introduced early.

Destructors are not introduced until dynamic data structures are covered. Introducing them before dynamic data structures would be pointless, since they would have no purpose.

Templates are not introduced early for three reasons: First, many compilers still do not handle them gracefully. With many compilers, class templates require special care if they are to be compiled separately. Often compilation of class templates has restrictions that make separate compilation impractical. Second, a template is a schema that generates numerous nontemplate classes. Thus, before one can understand what a template class is, one needs to know what a nontemplate class is. Once a student is comfortable with nontemplate classes, template classes are an easy and natural generalization. Third, many of the examples that benefit most from templates are data structures, like stacks and queues, which are usually not covered until the second programming course. However, if an instructor wishes to cover templates earlier, the material on templates was written so that it can be moved to an earlier place in the course.

Inheritance is covered briefly in Chapter 5 so that students become aware of the concept. However, this book does not teach students how to write their own derived classes until the final chapter of the book. The reason for this is that the examples that strongly motivate inheritance and derived classes often do not arise naturally at the beginning of a first course. One can make a point of introducing examples that use derived classes early in the first course, but it is difficult for students to get much realistic utility out of derived classes until they are more sophisticated programmers. These more advanced topics can easily be introduced later, and even without these topics students will have a sophisticated and realistic concept of classes. Chapter 15 does teach students how to define and use derived classes including the use of virtual functions. Some instructors may choose to leave that material for a second course. Other instructors will want to integrate this inheritance coverage into their course.

Even though we postpone the discussion of destructors, templates, and most of inheritance, the classes that are used early in the book are very sophisticated, and some would argue that they are too complicated to be covered so early in a first programming course. After all, in addition to the basic notions of member variables and member functions, beginning students reading this book will learn all of the following topics very early: public and private members, function overloading, operator overloading, friend functions, returning a reference so that they can overload the I/O operators << and >>, constructors for automatic initialization, constructors for type conversion, and a number of smaller issues. Some would argue that this is too much to give students so early. We have class tested this material, however, and found that the examples become unrealistic or poorly behaved if we omit any of these topics. Moreover, in class testing we found that students respond to early classes in basically the same way that they respond to early functions. Certainly the material presents some problems. However, students are as capable of learning the material early in the course as they will be later in the course. Moreover, covering classes early leaves students with a better working knowledge of classes. Students have a strong loyalty to the first technique they learn for solving a problem. So, if you want them to really use a technique, then you need to teach it early. Moreover, classes are not the hardest topic the students encounter. For example, classes are more intuitive and better behaved than ordinary (C style) arrays, which they learn later in the course.

Having made the case for early classes, we are still aware that not everybody wants to introduce classes as early as we do and so we have written the book to allow instructors to move coverage of classes to later in the course. This is discussed in the section of this preface on flexibility.

Summary Boxes

Each major point is summarized in a boxed section. These boxed sections are spread throughout each chapter. This allows the book to be read in many different ways by students with different backgrounds: Students with a good deal of programming experience can use the boxed sections to learn the material quickly. Students who do not adapt well to reading a full-length text can use the boxed sections to obtain the major points and then read more of the text for points that still need clarification after they have read the boxed sections and attended lectures. All students can use the boxed sections to preview the chapter, to review for exams, and as a short reference to check points they may have forgotten.

Self-Test Exercises

Each chapter contains numerous Self-Test Exercises at strategic points in the chapter. Complete answers for all the Self-Test Exercises are given at the end of each chapter. This second edition includes a number of new additional Self-Test Exercises that did not appear in the first edition.

Class Tested

The material has been fully class tested and revised in response to student and instructor reactions. Among other things, this class testing helped in determining our choice of which C++ concepts to include, which to omit, and which to place in optional sections. Preliminary versions of this material have been used in classes using a GNU compiler, in classes using a Sun CC compiler, and in classes using a Borland C++ compiler. The program code was designed to be portable and should work without modification on almost any C++ compiler.

Flexibility in Topic Ordering

This book was written to allow instructors wide latitude in reordering the material. To illustrate this flexibility we give a number of alternative orderings of topics after this paragraph. There is no loss of continuity when the book is read in any of these orders. In order to ensure this continuity when you rearrange material you do need to sometimes move sections rather than entire chapters. However, only large sections in convenient locations are moved. Under any of these orderings, whenever a chapter is first introduced, the chapter is covered without interruption up to some point. One cut point is inserted into the chapter and all the material before the cut is moved as a unit and all the material after the cut is moved as another unit. To help customize an ordering to any particular class's needs, the dependency chart, which follows this preface, describes many more possible orders in which the chapters and sections can be covered without loss of continuity.

Cut

Reordering 1: Late Classes

This version essentially covers all of an ANSI C course before going on to cover classes. The only thing that is very C++ like before the introduction of classes is the use of streams for I/O:

Basics: Chapters 1, 2, 3, 4, 5, and 7 (omitting Chapter 6 on defining classes). This material covers all of control structures, function definitions, and basic file I/O.

Arrays: Chapter 9, omitting the last section (Section 9.4), which uses classes. Chapter 10, omitting the last section (Section 10.2), which covers the STL string class.

Pointers and Dynamic Arrays: Chapter 11, omitting the last section (section 11.3), which covers material on classes.

Recursion: Chapter 12, omitting the programming example on classes that ends the chapter. (Alternatively, recursion may be moved to later in the course.)

Structures and Classes: Chapters 6, 8, and the last sections of Chapters 9, 10, and 11.

Pointers and Linked Lists: Chapter 14

Templates: Chapter 13

Inheritance: Chapter 15

Reordering 2: Classes Slightly Earlier

This version covers all control structures and the basic material on arrays before doing classes, but classes are covered a little earlier than the previous reordering.

Basics: Chapters 1, 2, 3, 4, 5, and 7 (omitting Chapter 6 on defining classes). This material covers all of control structures, function definitions, and basic file I/O.

One-Dimensional Arrays: Chapter 9, omitting the last section (Section 9.4), which uses classes.

Structures and Classes: Chapters 6, 8, and the last sections of Chapter 9.
Multidimensional Arrays and Strings: Chapter 10
Pointers and Dynamic Arrays: Chapter 11.
Recursion: Chapter 12.
Pointers and Linked Lists: Chapter 14
Templates: Chapter 13
Inheritance: Chapter 15

Reordering 3: Early Classes, but with All Control Structures Covered before Classes:

This is almost the regular ordering of the book. You only need to move Chapter 7 so that you cover Chapter 7 before Chapter 6.

Variations:

The first two sections of Chapter 12 (which cover basic recursion) can be covered any time after Chapter 4. The first two sections on pointers (in Chapter 11) can be covered before Chapter 10, which covers multidimensional arrays and strings. Most of the first section of Chapter 13, which covers function templates, can be covered anytime after Chapter 4. The first two, sections of Chapter 15 (15.1 and 15.2) which cover the basics of defining and using derived classes can be covered any time after Chapter 10. Other possible variations are shown in the dependency chart at the end of this preface.

Support Material

The following support material is available from the publisher:

Programs from the Text: All the programs in the text are available by anonymous ftp at `ftp.aw.com`. At the name prompt enter `anonymous`. The programs are in the directory `cseng/authors/savitch/psc++2e`.

Alternatively, you can find the programs by going to either the Addison Wesley Longman Web site or the author's Web site and following the links. The two Web sites are, respectively:

```
http://www.awl.com/cseng
http://www-cse.ucsd.edu/users/savitch/
```

Instructor's Resource Guide: A chapter-by-chapter instructor's guide including numerous teaching hints, quiz questions with solutions, and solutions to many programming exercises.

Contact your Addison Wesley Longman sales representative for details on obtaining the instructor's guide. If you wish to have an Addison Wesley Longman sales representative contact you, send e-mail to `aw.cse@awl.com`.

Email Contact

I would very much like to hear your comments so I can continue to improve this book to make it better fit your needs. Feel free to contact me at the following e-mail address:

`wsavitch@ucsd.edu`

Unfortunately, I am not able to provide students with solutions to exercises in this book or to other exercises provided by your instructor. I simply do not have enough time to answer the numerous requests I am getting for such detailed assistance. I also do not want to interfere with any instructor's plans for how students should go about solving programming problems. As at least a partial consolation to those who desire such help, let me point out that the book does include complete answers to all the Self-Test Exercises. Also, the Instructor's Guide does provide instructor's with some answers to Programming Projects, but that material is only available to college and university instructors who adopt the book, and it cannot be given out to students.

Acknowledgments

Numerous individuals and groups have provided me with suggestions, discussions, and other help in preparing this textbook. Much of the first edition of this book was written while I was visiting the Computer Science Department at the University of Colorado in Boulder. The remainder of the writing on the first edition and the work on this second edition were done in the Computer Science and Engineering Department at the University of California, San Diego (UCSD). I am grateful to these institutions for providing facilities and a conducive environment for teaching this material and writing this book.

David Teague deserves special acknowledgment. I very much appreciate his hard work, good insights, and careful rewriting of many sections of the book. This second edition could not have come out in a timely fashion without his help. He assisted in updating much of the text to reflect the new type *bool* and other updates

in the C++ language. He was, as a practical matter, a coauthor of the new Chapter 15, which covers inheritance.

The list of other individuals who have contributed critiques for earlier outlines and drafts of this book is too long to thank each contributor in the unique way that she or he deserves. So I must simply list them (in alphabetical order) and extend my deepest thanks to them all: Claire Bono, Andrew Burt, Karla Chaveau, Joel Cohen, Doug Cosman, Paulo Franca, Len Garrett, Jerrold Grossman, Dennis Heckman, Bob Holloway, Bruce Johnston, Thomas Judson, Michael Keenan, Barney MacCabe, Steve Mahaney, Michael Main, John Marsaglia, Nat Martin, Jesse Morehouse, Lt. Donald Needham, Dung Nguyen, Ken Rockwood, John Russo, and Jerry Weltman.

I extend a special thanks to the many students in my classes who tested and helped correct preliminary versions of this text. Their feedback, and the comments of other instructors who used early drafts of this book, provided some of the most helpful reviewing the book received. In particular, I'd like to offer a special thanks to Joe Faletti, Paul Kube, Susan Seitz, and David Teague for their valuable feedback after class testing material from the first and second editions of this book.

I again thank David Teague. This time for his excellent work in preparing the instructor's guide.

I extend a special thanks to Carter Shanklin, my editor for the first edition of this book. His support and advice was invaluable in shaping the basic design of the book. I thank Amy Willcutt and Julie Dunn for their excellent work in handling production and reviews under very tight time constraints. Finally, I thank my editor Susan Hartman who provided the reviewers, and the encouragement that allowed this second edition to be produced in a timely fashion. The various pieces would not have come together without her expert guidance.

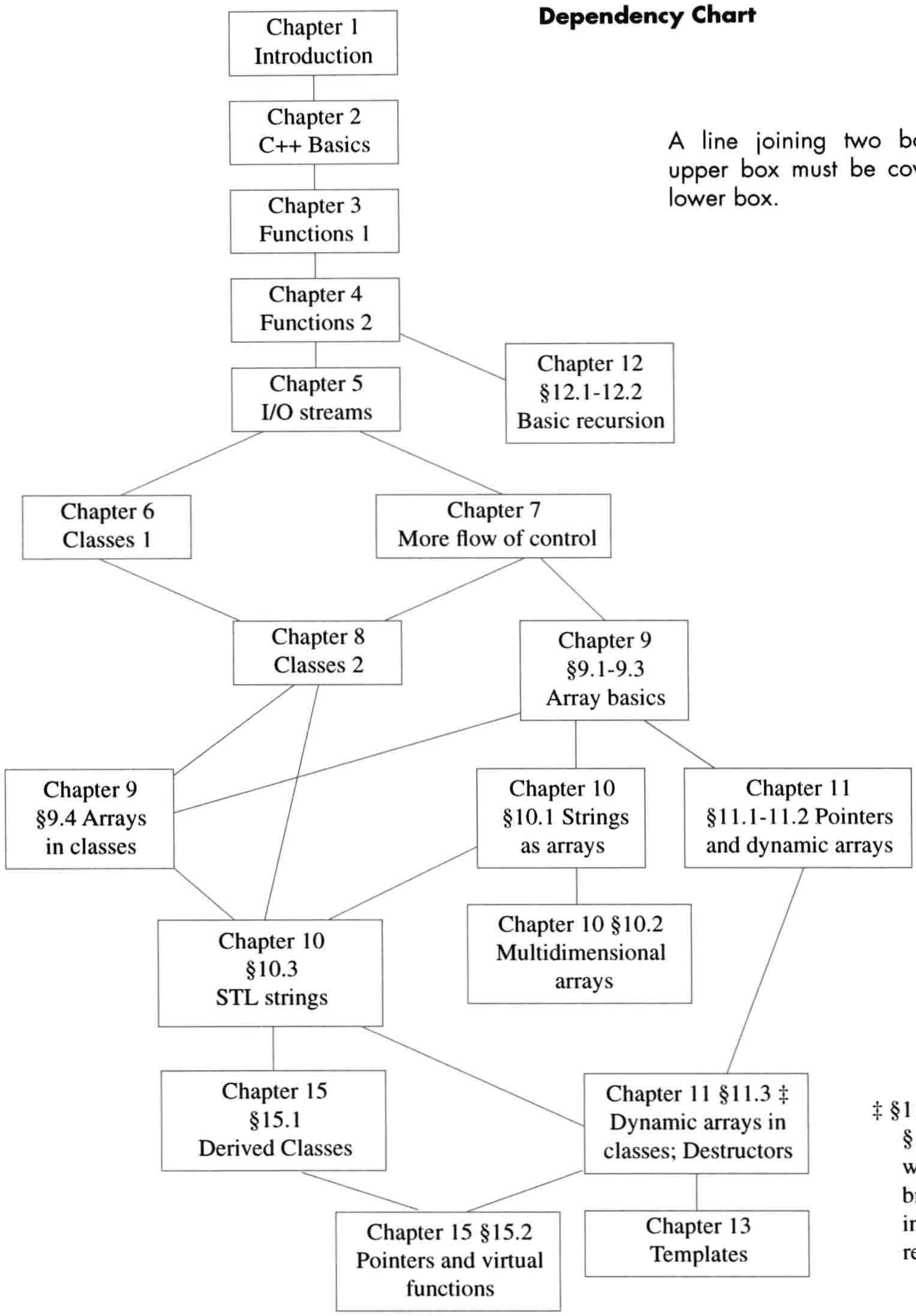
W.S.
`http://www-cse.ucsd.edu/users/savitch/`

Dependency Chart

The dependency chart on the next page shows possible orderings of chapters and subsections. A line joining two boxes means the upper box must be covered before the lower box. Any ordering that is consistent with this partial ordering can be read without loss of continuity. If a section number or section numbers are given in a box, then the box refers only to those sections and not to the entire chapter.

Dependency Chart

A line joining two boxes means the upper box must be covered before the lower box.


Chapter 1 Introduction
Chapter 2 C++ Basics
Chapter 3 Functions 1
Chapter 4 Functions 2
Chapter 12 §12.1-12.2 Basic recursion
Chapter 5 I/O streams
Chapter 6 Classes 1
Chapter 7 More flow of control
Chapter 8 Classes 2
Chapter 9 §9.1-9.3 Array basics
Chapter 9 §9.4 Arrays in classes
Chapter 10 §10.1 Strings as arrays
Chapter 11 §11.1-11.2 Pointers and dynamic arrays
Chapter 10 §10.3 STL strings
Chapter 10 §10.2 Multidimensional arrays
Chapter 15 §15.1 Derived Classes
Chapter 11 §11.3 ‡ Dynamic arrays in classes; Destructors
Chapter 15 §15.2 Pointers and virtual functions
Chapter 13 Templates

‡ §11.3 does not depend on §10.3 in any essential way, although it does briefly mention material in §10.3. §11.3 does require §9.4.

Contents