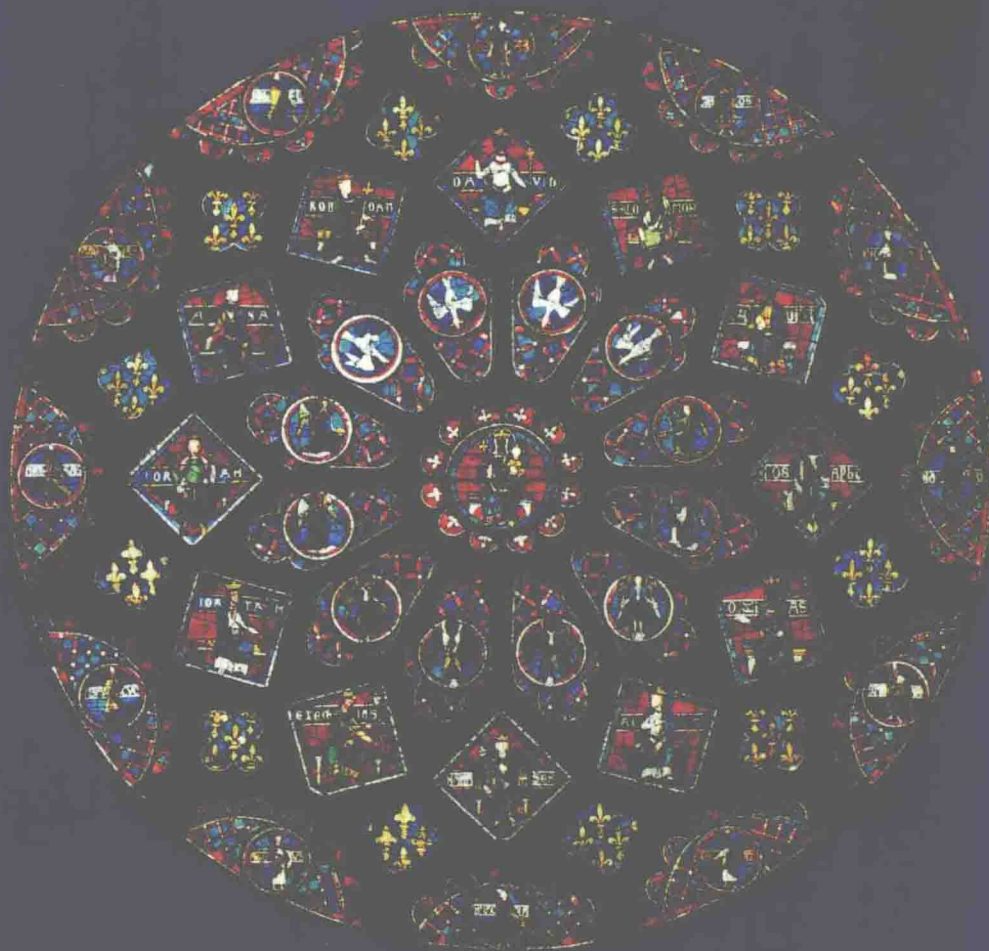


# Turbo C++

An Introduction to Computing



**JOEL ADAMS**  
**SANFORD LEESTMA**  
**LARRY NYHOFF**

# **Turbo C++: An Introduction to Computing**

**Joel Adams  
Sanford Leestma  
Larry Nyhoff**

Calvin College  
Grand Rapids, Michigan



Prentice Hall, Upper Saddle River, New Jersey 07458

Library of Congress Cataloging-in-Publication Data

Adams, Joel.

Turbo C++: an introduction to computing / Joel Adams, Sanford  
Leestma, Larry Nyhoff.

p. cm.  
Includes index.

ISBN 0-13-439928-5

1. C++ (Computer program language) 2. Turbo C++. I. Leestma,  
Sanford. II. Nyhoff, Larry R. III. Title.

QA76.73.C153A34 1996

005.13'3—dc20

95-41348

CIP

About the Cover: The rose window on the cover is the Rose de France (c. 1233) in the north transept of the Chartres cathedral. Like many of the other beautiful rose windows in French cathedrals, it is an early example of object-oriented design, in which objects of various shapes, sizes, colors, and meanings are fitted together according to certain basic principles. An interesting property of this window is how its components—the outer semicircles containing the last twelve Old Testament prophets; the quatrefoils containing the three-petaled fleur-de-lis, symbols of the Annunciation and of royalty; the twelve squares containing the kings of the Virgin Mary's ancestry as recorded by St. Matthew; the twelve circles containing doves, angels and other celestial beings; and the twelve-petaled central rosette containing the Virgin Mary—are arranged into spiral-shaped patterns based on the golden section and the Fibonacci sequence.

Acquisitions Editor: Alan Apt

Developmental Editor: Sondra Chavez

Production Manager: Judy Winthrop

Cover Design: Robert Freese

Cover Art: Giraudon/Art Resource LAC 58289. Rose Window, North Transept, Chartres Cathedral  
(13th C.).

Illustrations: York Graphic Services, Inc.

Manufacturing Buyer: Donna Sullivan



Copyright © 1996 by Prentice-Hall, Inc.

Simon & Schuster/A Viacom Company

Upper Saddle River, New Jersey 07458

The author and publisher of this book have used their best efforts in preparing this book. These efforts include the development, research, and testing of the theories and programs to determine their effectiveness. The author and publisher shall not be liable in any event for incidental or consequential damages in connection with, or arising out of, the furnishing, performance, or use of these programs.

All rights reserved. No part of this book may be reproduced, in any form or by any means, without permission in writing from the publisher.

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

ISBN 0-13-439928-5

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*

# PREFACE

---

In order to properly introduce students to computing, we believe that the first computer course for students should accomplish two goals:

1. The student should be introduced to the discipline, methodologies, and techniques of computer programming using a modern programming language.
2. The student should be introduced to the breadth of the discipline of computing, so that he or she comes to understand the role of programming in the broader context of computing.

The aim of this textbook is to accomplish both of these goals.

## The Programming Goal

In order to accomplish our first goal, we have chosen the language C++, whose features (we believe) make it the language of choice in the immediate future. A few of the reasons for this choice are as follows:

1. C++ provides a *strong type-checking system*.
2. C++ provides *reference parameters* for its functions.
3. C++ provides a *library* mechanism, whereby a programmer can store generally useful functions in a library, so that they can be reused by any program that needs them.
4. C++ provides function name and operator *overloading*, allowing a programmer to use the same name to define subprograms that perform similar operations on different data types.
5. C++ provides *the class*, whereby both an object's data members and its operations can be encapsulated within a single, protected structure.
6. C++ provides *derived classes*, a mechanism whereby one class can *inherit* the data members and operations from another class, allowing a *class hierarchy* to be built.

These latter features of C++ allow it to be characterized as an *object-oriented programming* (OOP) language. From windowing systems to graphic user interfaces to object-oriented databases, more and more of today's best computing appli-

cations are being developed in C++ using the OOP approach, making it today's language of choice.

**C++ Is Not C.** Many people erroneously believe that C++ is simply its parent language C with a few additional features and that C++ is therefore inappropriate as a first programming language. In fact, most of the design flaws of C have been corrected in C++, making it a suitable language for a first course in computing.

**The Importance of Modeling.** Another popular misconception is that because many C programs are cryptically written, C++ programs probably suffer the same drawback. We believe that cryptic programs are caused by *people*, not by a *language*, and an undisciplined programmer will write cryptic programs in any language, not just C. This is because:

- 1. Most people learn C from the examples they see in a book; and
- 2. Most C books are not introductory programming texts, but rather *language references* intended for professional programmers.

One of the aims of this text is to teach a disciplined programming style (to those with no programming experience) that results in well-documented, easy-to-read programs.

We believe that *what students learn depends on the models that they see*. That is, if they are presented with examples that are well-written, well-documented, and maintainable, then the programs they write will exhibit these same characteristics, regardless of the language used. To that end, this text contains a large number of examples that illustrate good programming style.

**Standard C++.** At the time of this writing, the American National Standards Institute (ANSI) Committee X3J16 had not completed a C++ standard. In the absence of such a standard, we have used the *Annotated C++ Reference Manual*<sup>1</sup> as our primary reference in preparing this text.

While written in the Turbo environment, the examples in this text are not restricted to that environment and have been successfully ported to the following environments:

Computing Environment	C++ Compiler
UNIX (Sun, Apollo, etc.)	GNU g++ (v. 2.4.5) <sup>2</sup>
DOS, Windows, OS-2 (IBM PC)	Turbo C++ (v. 3.0) <sup>3</sup>
MacOS (Apple Macintosh)	Symantec C++ For Macintosh (v. 6.01) <sup>4</sup>

<sup>1</sup> Margaret Ellis and Bjarne Stroustrup, *The Annotated C++ Reference Manual*. (Addison-Wesley, 1992).

<sup>2</sup> GNU g++ is a copylefted product of the Free Software Foundation, Inc., 675 Mass Ave., Cambridge, MA 02139; and is available for free via anonymous ftp from [prep.ai.mit.edu/pub/gnu](ftp://prep.ai.mit.edu/pub/gnu).

<sup>3</sup> Turbo C++ is a copyrighted product of Borland International Inc., 1800 Green Hills Rd., P.O. Box 660001, Scotts Valley, CA 95067; and is available from most software vendors for the IBM PC.

<sup>4</sup> Symantec C++ for Macintosh is a copyrighted product of Symantec Corporation, 10201 Torre Ave., Cupertino, CA 95014; and is available from most software vendors for the Apple Macintosh.

## The Breadth of Computing

In 1991, a new set of curriculum recommendations was published in *Computing Curricula 1991: Report of the ACM/IEEE-CS Joint Curriculum Task Force*. One theme of this report is that an introductory course in computing should introduce the various knowledge areas of the discipline:

- Architecture
- Artificial intelligence and robotics
- Database and information retrieval
- Human–computer communication
- Numerical and symbolic computation
- Operating systems
- Programming languages
- Software methodology and engineering
- Social, ethical, and professional context

In this text, we include a number of sections that illustrate these areas, trying to capture the spirit of these curriculum guidelines in a natural, unobtrusive way. These sections have been carefully selected in accordance with the *Computing Curricula 1991* report to provide an overview of computer science and to provide a foundation for further study in theoretical and/or applied computer science. They have been highlighted in special PART OF THE PICTURE sections, which are marked with an icon in the shape of a puzzle piece. These sections include:



- What Is Computer Science?
- The History of Computing
- Computer Organization
- Social, Professional, and Ethical Issues
- Syntax and Semantics
- Computer Architecture
- Computability Theory
- Introduction to Numeric Computation
- Introduction to Algorithm Analysis
- Simulation
- Artificial Intelligence
- Databases
- Data Encryption
- The Type Hierarchy
- Analysis of Algorithms
- Automata and Language Translation
- Numeric Computation
- Computer Graphics
- Expert Systems

A solid base is thus established for later courses in theoretical and/or applied computer science.

## About the Text

**Organization.** We have organized the text material into four parts:

- The text begins with an *Introduction* consisting of two chapters that present an overview of computing and programming.
- The second part of the text, *Computing with Simple Objects*, consists of five chapters that introduce the student to the basic ideas of computing, including types, variables, constants, functions, I/O, libraries, selective control, repetitive control, parameter passing mechanisms, and so on. Each of these topics is covered in the context of simple data types: the integer, real, character, and boolean types.
- The third part, *Computing with Class Objects*, consists of seven chapters that extend the ideas from the second part to problems involving more sophisticated data types, including files, character strings, enumerations, arrays, and sets. Where applicable, C++ class libraries are used to implement objects that can be easily reused and maintained.
- The final part, *Computing with Advanced Objects*, consists of three chapters that introduce advanced topics, including indirection, run-time allocation/deallocation, and linked structures, such as linked lists, stacks, queues, and trees, each implemented using C++ classes.

We think that most of the first three parts can be covered in a typical semester course. Some or all of the fourth part can be covered in accelerated courses or in a second course or can be used as enrichment material or for honors work.

**Features.** This first edition text breaks new ground in many ways, by providing a gentle introduction to new topics such as designing for reusability; the use, design, and implementation of class libraries; the overloading of operators and function names; and the OOP approach to program design. A few of the features of the text are:

- Each chapter begins with an example problem, whose solution is used to introduce the ideas of that chapter. Following this example, the concepts and theory behind these ideas are explored, and other examples are presented to reinforce the ideas. In this approach, students see the *practice* of a new topic before the *abstract* definitions and theory behind that topic, providing them with a framework in which those abstract aspects can be organized and understood.
- A wealth of examples illustrates each topic, allowing students to distinguish what is essential from what is optional. In the spirit of *Computing Curricula 1991*, these examples are chosen from a wide range of applications and have been written to model good structure and style. Those marked in the text with a disk icon are included on the data disk that accompanies the Instructor's Manual or can be downloaded from our ftp site as follows:



```
ftp to ftp.prenhall.com
login as anonymous
use your email address as the password
cd to pub/EMS/adams/turbo.c++
```

- Optional sections (marked with asterisks) delve into the more advanced topics, without requiring that they be covered in a normal introductory course.

- Each chapter ends with *Programming Pointers* that highlight important points, especially proper techniques of design and style, as well as common programming pitfalls.
- Color is used to emphasize and highlight important features.
- Exercise sets include short written exercises as well as a large number of programming exercises and projects drawn from a wide range of application areas.

## Supplementary Materials

A number of supplementary materials are available from the publisher. These include the following:

- A solutions manual that contains solutions to the exercises in the text, including many of the programming exercises.
- A lab manual and diskette prepared by Professor Joel Adams.
- Disk containing the sample programs and data files referenced in the text. In addition, this material can be downloaded from our ftp site as follows:  
ftp to ftp.prenhall.com  
login as anonymous  
use your email address as the password  
cd to pub/EMS/adams/turbo.c++
- Disks containing all of the text exercises and solutions to many of the programming exercises.

## Suggestions

The authors welcome feedback, both positive and negative. Comments about features of the text that work especially well, as well as about features of the text that need improvement, will aid us in the preparation of subsequent editions. We would also appreciate being notified of errors. Such comments can be directed to any of the authors at the following U.S. mail address:

Department of Mathematics and Computer Science  
Calvin College  
Grand Rapids, Michigan 49546  
USA

or to [adams@calvin.edu](mailto:adams@calvin.edu), [lees@calvin.edu](mailto:lees@calvin.edu), or [nyhl@calvin.edu](mailto:nyhl@calvin.edu) via the Internet.

## Acknowledgments

The comments and suggestions made by the following reviewers of the forerunner of this text, *C++: An Introduction to Computing*, were valuable, and their work is much appreciated: Jose Cisnaros, Metropolitan College of Denver; Ann Ford, University of Michigan; Mike Holland, Northern Virginia Community College; John Lowther, Michigan Technological University; Dick Reed, Michigan State University; and Peter Spoerri, Fairfield University. Our thanks also go to Evan Scott of

Borland International, Inc., for his work in checking our descriptions of Turbo C++ features in this text. We must also thank our wives, Barbara, Marjory, and Shar, for not complaining about the times that their needs and wants were slighted by our busyness. Above all, we give thanks to God for giving us the opportunity, ability, and stamina to prepare these texts.

J.C.A.  
S.C.L.  
L.R.N.

# CONTENTS

---

<b>PREFACE</b>	<b>xxi</b>
<b>PART I: INTRODUCTION TO COMPUTING</b>	<b>1</b>
<b>1 THE SCIENCE OF COMPUTING</b>	<b>3</b>
1.1 PART OF THE PICTURE: What Is Computer Science?	4
1.2 PART OF THE PICTURE: The History of Computing	5
The Mechanization of Arithmetic	6
The Stored Program Concept	9
Mechanical Computers	10
Early Electronic Computers	14
Modern Computers	15
Computer Software	17
A Brief History of C++	18
Summary	18
1.3 PART OF THE PICTURE: Computer Organization	19
Computing Systems	19
Memory Organization	20
Number Systems	20
Data Storage	22
Instruction Processing	26
1.4 PART OF THE PICTURE: Social, Professional, and Ethical Issues	29
Exercises	32
<b>2 PROGRAM DEVELOPMENT</b>	<b>37</b>
2.1 An Introduction to Software Development	39
Problem 1: Revenue Calculation	39
Stage 1: Problem Analysis and Specification	39

- Stage 2: Design 40
- Stage 3: Coding 42
- Stage 4: Verification and Validation 45
- Stage 5: Maintenance 46
- Exercises 47
- 2.2 Problem Analysis and Specification 49**
  - Problem 2: The Pollution Index Problem—Specification 49
  - Problem 3: The Mean Time to Failure Problem—Specification 50
  - A “Real-World” Problem: Payroll 50
- 2.3 Design 51**
  - Object Attributes 52
  - Operations 53
  - Algorithms 54
  - Object-Oriented Design 57
- 2.4 Coding 60**
  - Types 60
  - Variable Data Objects 60
  - Constant Data Objects 62
  - Numeric Operations 63
  - Input/Output 63
  - Comments 64
  - Programming Style 64
  - Coding in Turbo C++ 66
- 2.5 Verification and Validation in Turbo C++ 67**
  - Translating a Turbo C++ Program 68
  - Testing a Turbo C++ Program 70
  - Life-Critical Systems 73
- 2.6 Maintenance 74**
  - Exercises 74

**PART II: COMPUTING WITH SIMPLE OBJECTS 79**

- 3 GETTING STARTED WITH EXPRESSIONS 81**
  - 3.1 C++ Programs 82**
    - Examples of Programs 83
    - The Main Function 84
  - 3.2 Programming with Libraries—Not Reinventing the Wheel 85**
    - Libraries 86
    - Using a Library 86
    - Linking to a Library 87
  - 3.3 Declarations: Types of Objects 87**
    - Fundamental Types 88
    - Identifiers 93

Named Constants	94
Variables	97
Variable Initialization	100
<b>Exercises</b>	<b>100</b>
<b>3.4 Numeric Expressions</b>	<b>101</b>
Operators	102
Functions	106
<b>Exercises</b>	<b>108</b>
<b>3.5 Assignment Expressions</b>	<b>110</b>
Assignment as an Operation	114
Chaining Assignment Operators	114
The Increment and Decrement Operations	115
Other Assignment Shortcuts	117
Transforming Expressions into Statements—Semicolons	119
A Final Word	119
<b>Exercises</b>	<b>120</b>
<b>3.6 Input/Output Expressions</b>	<b>121</b>
I/O Streams	122
Input Expressions	123
Output Expressions	126
Example: Calculating Wages	128
Output Formatting	129
<b>Exercises</b>	<b>134</b>
<b>3.7 Example: Truck Fleet Accounting</b>	<b>136</b>
Problem	136
Specification	136
Design	136
Coding and Testing	137
<b>3.8 PART OF THE PICTURE: Syntax and Semantics</b>	<b>139</b>
<b>Exercises</b>	<b>142</b>
<b>Programming Pointers</b>	<b>143</b>
Program Design	143
Potential Problems	143
Program Style	147
<b>Programming Projects</b>	<b>147</b>
 <b>4 FUNCTIONS AND LIBRARIES</b>	 <b>151</b>
<b>4.1 Computing with Formulas</b>	<b>152</b>
Example: Temperature Conversion	152
<b>4.2 Computing with Functions</b>	<b>154</b>
Defining a Function	157
Declaring a Function	160

Calling a Function	161
Summary	164
Exercises	165
<b>4.3 Computing with Libraries</b>	<b>167</b>
Constructing a Library	167
Using a Library in a Program	171
Translating a Library	172
Summary	173
Exercises	175
<b>4.4 Example Program: Converting Days to Distance</b>	<b>175</b>
Problem	175
Specification	175
Design	176
Coding	178
Program Testing	183
Exercises	184
<b>4.5 Computing with Class Libraries</b>	<b>185</b>
Introduction to Classes	186
Summary	189
<b>Programming Pointers</b>	<b>189</b>
Program Design	189
Potential Problems	190
Program Style	193
<b>Programming Projects</b>	<b>194</b>
 <b>5 SELECTIVE EXECUTION</b>	 <b>195</b>
<b>5.1 Sequential Execution</b>	<b>196</b>
Compound Statements	196
<b>5.2 Introducing Selective Execution</b>	<b>198</b>
Problem: Computing a Reciprocal	198
A Better Algorithm	199
A Better Function	200
<b>5.3 Conditions</b>	<b>200</b>
Simple Boolean Expressions	200
Compound Boolean Expressions	202
Operator Precedence	204
Short-Circuit Evaluation	205
<b>5.4 PART OF THE PICTURE: Computer Architecture</b>	<b>205</b>
Exercises	208
<b>5.5 Selection: The if Statement</b>	<b>209</b>
Example: Pollution Index Problem	209
The Simple if Statement	211

The General <b>if</b> Statement	212
Example: Grade Computation	215
Example: Grade Computation—Version 2	218
The <b>if-else-if</b> Form	220
Example: Grade Computation—Version 3	222
Pitfall: Confusing = and ==	225
Exercises	227
<b>5.6 Selection: The switch Statement</b>	<b>228</b>
Example: Converting Year Names to Year Codes	229
Form of the <b>switch</b> Statement	231
The <b>break</b> Statement	232
Choosing the Selection Statement to Use	232
Drop-Through Behavior	234
Example: Grade Computation—Version 4	235
Exercises	238
<b>*5.7 Selection: Conditional Expressions</b>	<b>239</b>
Exercises	243
<b>5.8 Example: A Menu-Driven Temperature Converter</b>	<b>243</b>
Problem	243
Specification	244
Design	245
Coding	247
<b>5.9 PART OF THE PICTURE: Computability Theory</b>	<b>250</b>
<b>Programming Pointers</b>	<b>252</b>
Program Design	252
Potential Problems	255
Program Style	258
<b>Programming Projects</b>	<b>260</b>
 <b>6 REPETITION STATEMENTS</b>	 <b>263</b>
<b>6.1 Introduction to Repetition: The For Loop</b>	<b>264</b>
The Summation Problem	264
Counting Loops	268
Nested Loops: Displaying a Multiplication Table	272
Words of Warning	275
Exercises	276
<b>6.2 Repetition: The While Loop</b>	<b>278</b>
Example: Follow the Bouncing Ball	278
The <b>while</b> Statement	280
Words of Warning	282
Using Sentinel-Controlled While Loops to Input a List of Values	283
Using End-of-File-Controlled While Loops to Input a List of Values	287
Exercises	291

**6.3 Repetition: The Do-While Loop 292**

Example: How Many Digits? 292

A Posttest Loop 296

Words of Warning 297

Using a Do-While Loop to Input a List of Values 298

**Exercises 304**

**\*6.4 Repetition: The Forever Loop 305**

Example: The ATM Menu-Choice Problem 306

An Unrestricted Loop 310

Input Loops Using Sentinels 313

Input Loops Using End-of-File 316

Words of Warning 317

**Exercises 318**

**6.5 Guidelines for Using Loops 320**

Choosing the Right Loop 320

Confusing = and == 323

**Exercises 324**

**6.6 Techniques for Testing and Debugging Loops 324**

An Example 325

Trace Tables 328

Debugging 328

Modifying and Testing the Program 329

Summary 332

**Exercises 332**

**6.7 Example: Depreciation Tables 333**

Problem: Depreciation Tables 333

Using the Depreciation Functions 336

**Exercises 341**

**6.8 PART OF THE PICTURE: Introduction to Numeric Computation 341**

Curve Fitting: Least Squares Line 342

Solving Equations 347

**Exercises 351**

**6.9 PART OF THE PICTURE: Introduction to Algorithm Analysis 353**

**Programming Pointers 355**

Program Design 355

Potential Problems 356

Program Style 358

**Programming Projects 359**

**7 FUNCTIONS: AN IN-DEPTH LOOK**

**361**

**7.1 Returning Multiple Values from a Function 362**

Example: The Half-Adder Revisited 362

**7.2 Reference Parameters 366**

Value Parameters 366

Reference Parameters 368

Using Reference Parameters 371

**7.3 Examples Using Reference Parameters 374**

Example 1: Making Change 374

Example 2: Exchanging the Values of Two Variables 379

Example 3: Analyzing a List of Data Values 385

**Exercises 388****7.4 Default Values for Parameters 391**

Example: Evaluating Fourth-Order Polynomials 391

The Difficulty 392

The Solution 393

Limitations of Default Parameter Values 394

Summary 395

**\*7.5 Varying the Number of Arguments 396**

Example: Evaluating Polynomials of Any Degree 396

**7.6 Inline Functions 400****7.7 Object Lifetime and Storage Classes 402**

First Things First 403

Object Lifetime 403

Storage Classes 404

Example: Computing Sums 405

Using the **register** Specifier 407**7.8 PART OF THE PICTURE: Simulation 408**

Example: Modeling a Dice Roll 409

**Exercises 413****7.9 Introduction to Recursion 415**

Examples: Factorial, Exponentiation, and Number Reversal 416

Recursion versus Iteration 432

**Exercises 433****7.10 PART OF THE PICTURE: Artificial Intelligence 437**

Example: The Towers of Hanoi 438

**Exercises 441****Programming Pointers 441**

Program Design 441

Potential Problems 442

Program Style 443

Programming Projects 444

<b>PART III: COMPUTING WITH CLASS OBJECTS</b>	<b>447</b>
<b>8 FILES AND STREAMS</b>	<b>449</b>
8.1 Computing with Files	450
Example: Processing Meteorological Data	451
8.2 <b>fstream</b> Objects and Operations	456
Declaring <b>fstream</b> Objects	456
The Basic <b>fstream</b> Operations	457
Summary	469
Exercises	470
8.3 Examples: Employee File Processing and Payroll Computation	472
Problem 1: Processing a File of Employee Data	472
Problem 2: Payroll Computation ( <b>fstream</b> Objects as Parameters)	475
8.4 Additional <b>fstream</b> Operations	481
The <b>get()</b> and <b>put()</b> Members	482
The <b>seekg()</b> , <b>tellg()</b> , <b>seekp()</b> , and <b>tellp()</b> Members	485
Controlling I/O State Flags	490
The <b>peek()</b> , <b>putback()</b> , and <b>ignore()</b> Members	494
Example: A Goof-Proof Numeric Input Function	496
8.5 PART OF THE PICTURE: Databases	498
Exercises	500
Programming Pointers	501
Program Design	501
Potential Problems	502
Programming Projects	504
 <b>9 CHARACTER STRINGS</b>	 <b>507</b>
9.1 A Sample Problem	508
Problem: Reversing a String	508
9.2 Declaring Strings	511
9.3 String I/O	514
Input	514
Output	517
Example: Copying a File	518
Exercises	521
9.4 More Operations on Strings Objects	522
String Length	522
The Subscript Operator	523
Assignment	525
The Relational Operators	527
Concatenation	529