Preface

The C++ programming language is derived from the C programming language, with added features to support object-oriented programming through the use of classes and programmer-defined types. The features of the C programming language that make it attractive for system-level operations and embedded programming are also supported by C++, making C++ one of the most powerful and versatile programming languages available—and a good choice for an introduction to computing course for scientists and engineers. This text was written to introduce engineering problem solving with C++ and also the object-oriented features of the C++ programming language. Our objectives are the following:

- to develop a consistent methodology for solving engineering problems
- to present the object-oriented features of C++, while focusing on the fundamentals of programming and problem solving
- to illustrate a problem-solving process with C++ through a variety of engineering examples and applications
- to provide an easy-to-understand, integrated introduction to data types, functions, and container classes defined in the C++ Standard Template Library

To accomplish these objectives, Chapter 1 presents a five-step process that is used consistently in the rest of the text for solving engineering problems. Chapter 2 introduces the built-in data types supported by C++ and provides an introduction to classes, pre-defined objects, and member functions that support standard input and output. Chapters 3–6 present the fundamental capabilities of C++ for solving engineering problems, including control structures, data files, functions, and programmer-defined data types. Chapters 7 and 8 present arrays, vectors, and the string class. Chapter 9 introduces the use of pointers, dynamic memory allocation, and linked data structures. Chapter 10 provides a more in-depth look at some advanced topics, including function templates, class templates, recursive member functions, inheritance, and virtual functions. Throughout all these chapters, we present a large number of examples from many different engineering, science, and computer science disciplines. The solutions to these examples are developed using the five-step process and Standard C++.

Features of the Third Edition

The third edition of our text:

- Introduces students to three integrated development environments (IDEs)
  - NetBeans
  - MS Visual Studio
- Includes new engineering applications using global positioning system (GPS) data and data used with tsunami warning systems.
- Includes coverage of bitwise operators.

Preface

- Has expanded coverage of control structures.
- Introduces classes and the development of programmer-defined data types early in the text as optional chapter sections, for flexibility.
- Integrates coverage of classes throughout the text and offers a comparison of standard and object-based solutions.
- Includes additional Statement Boxes, Program Traces and Memory Snapshots, and flowcharts.

Student Resources and an Instructor’s Resource Center (IRC) are available online at www.pearsonhighered.com/etter.

Prerequisites

No prior experience with the computer is assumed. The mathematical prerequisites are college algebra and trigonometry. Of course, the initial material can be covered much faster if the student has used other computer languages or software tools.

Course Structure

The material in these chapters was selected to provide the basis for a one-term course in engineering and scientific computing. These chapters contain the essential topics of mathematical computations, character data, control structures, functions, arrays, classes, and pointers. Students with a background in another computer language should be able to complete this material in one semester. A minimal course that provides only an introduction to C++ can be designed using the nonoptional sections of the text. (Optional sections are indicated in the Contents with an asterisk.) Three ways to use the text, along with the recommended chapter sections, are

- Introduction to C++ Many freshman introductory courses introduce the student to several computer tools in addition to language. For these courses, we recommend covering the nonoptional sections of Chapters 1–8. This material presents to students the fundamental capabilities of C++, and they will then be able to write substantial programs using mathematical computations, character data, control structures, programmer-defined data types, functions, and arrays.

- Problem Solving with C++ In a semester course devoted specifically to teaching students to master the C++ language, we recommend covering all nonoptional sections of Chapters 1–10. This material covers all the fundamental concepts of the C++ language, including mathematical computations, character data, control structures, functions, arrays, classes, templates, and pointers.

- Problem Solving with C++ and Numerical Techniques Upper-level students or students who are already familiar with other high-level languages will be able to cover the material in this text very quickly. In addition, they will be able to apply the numerical-technique material to their other courses. Therefore, we recommend that these students cover all sections of Chapters 1–10, including the optional material.

The chapters in this text were designed to give the instructor flexibility in the ordering of topics. Coverage of programmer-defined types and classes is incorporated throughout the text, beginning with Chapter 2. However, coverage of classes is placed at the end of each chapter, in an optional section. A dependency chart is provided on the next page for illustration.
Preface

Problem-Solving Methodology

The emphasis on engineering and scientific problem solving is an integral part of the text. Chapter 1 introduces a five-step process for solving engineering problems using the computer:

1. State the problem clearly.
2. Describe the input and output information, and determine required data types.
3. Work a simple example by hand.
4. Develop an algorithm and convert it to a computer program.
5. Test the solution with a variety of data.

To reinforce the development of problem-solving skills, each of these five steps is clearly identified each time a complete engineering problem is solved. In addition, top-down design and stepwise refinement are presented with the use of decomposition outlines, pseudocode, and flowcharts.

Engineering and Scientific Applications

Throughout the text, emphasis is placed on incorporating real-world engineering and scientific examples and problems. This emphasis is centered around a theme of engineering challenges, which include

- prediction of weather, climate, and global change
- computerized speech understanding
- image processing
- artificial intelligence
- enhanced oil and gas recovery
- simulation

Each chapter begins with a photograph and a discussion of an aspect of one of these challenges that provides a glimpse of some of the exciting and interesting areas in which engineers might work. Later in the chapter, we solve a problem that not only relates to the introductory problem, but also has applications in other problem solutions.

Standard C++

The statements presented and all programs developed use C++ standards developed by the International Standards Organization and the American National Standards Institute (ISO/ANSI) C++ Standards committee. ISO and ANSI together have published the first international standard for the C++ programming language. By using Standard C++, students learn to write portable code that can be transferred from one computer platform to another. Many of the standard capabilities of the C++ programming language are discussed in the text. Additional components of the C++ standard library are discussed in Appendix A.
Software Engineering Concepts

Engineers and scientists are expected to develop and implement user-friendly and reusable computer solutions. Learning software engineering techniques is therefore crucial. Readability and documentation are stressed in the development of programs. Additional topics that relate to software engineering are discussed throughout the text and include issues such as software life cycle, portability, maintenance, modularity, recursion, abstraction, reusability, structured programming, validation, and verification.

Four Types of Problems

Learning any new skill requires practice at a number of different levels of difficulty. We have developed four types of exercises that are used throughout the text to develop problem-solving skills. The first set of exercises is Practice! problems. These are short-answer questions that relate to the section of material just presented. Most sections are immediately followed by a set of Practice! problems so that students can determine if they are ready to continue to the next section. Complete solutions to all the Practice! problems are included at the end of the text.

The Modify! problems are designed to provide hands-on experience with example programs and the programs developed in the Problem Solving Applied sections. In these sections, we develop a complete C++ program using the five-step process. The Modify! problems ask students to run the program with different sets of data, to test their understanding of how the program works and of the relationships among the engineering variables. These exercises also ask the students to make simple modifications to the program and then run the program to test their changes.

All chapters end with a set of Exam Practice! problems, and every chapter includes a set of Programming Problems. The Exam Practice! problems are short-answer questions that relate to the material covered in the chapter. These problems help students determine how well they understand the features of C++ presented in the chapter. The Programming Problems are new problems that relate to a variety of engineering applications, and the level of difficulty ranges from very straightforward to longer project assignments. Each programming problem requires that the student develop a complete C++ program or function. Engineering data sets for many of the problems are included within the Instructor’s Resource Center to use in testing. Also provided within the IRC are solutions to all of the Exam Practice! problems and Programming Problems.

Study and Programming Aids

Statement Boxes, UML diagrams, and Program Traces provide easily accessible visual illustrations of important concepts. Margin notes are used to help the reader not only identify the important concepts, but also easily locate specific topics. In addition, margin notes are used to identify programming style guidelines and debugging information. Style guidelines show students how to write C++ programs that incorporate good software discipline; debugging sections help students recognize common errors so that they can avoid them. The programming style notes are indicated with the margin note Style, and the debugging notes with a bug.
Preface

Object-oriented features of C++ display an OOP icon to help students recognize these features early in the text. Each Chapter Summary contains a summary of the style notes and debugging notes, plus a list of the Key Terms from the chapter and a C++ Statement Reference of the new statements, to make the book easy to use as a reference.

Optional Numerical Techniques

Numerical techniques that are commonly used in solving engineering problems are also discussed in optional sections in the chapters, and include interpolation, linear modeling (regression), root finding, numerical integration, and the solution to simultaneous equations. The concept of a matrix is also introduced and then illustrated using a number of examples. All of these topics are presented assuming only a trigonometry and college algebra background.

Appendices

To further enhance reference use, the appendices include a number of important topics. Appendix A contains a discussion of components in the C++ standard library. Appendix B presents the ASCII character codes. Appendix D contains a list of references used throughout the text. A MATLAB reference is also included as Appendix C, and solutions to Practice! problems make up Appendix E.

Additional Resources

All instructor and student resources can be accessed at www.pearsonhighered.com/etter. Here, students can access all source code for the book, and instructors can register for the password-protected Instructor's Resource Center. The IRC contains all the example programs used in the text, complete solutions to all the Programming Problems found at the end of each chapter, testbank questions, as well as data files to use with application problems and a complete set of Lecture PowerPoint slides.

Acknowledgments

We would like to thank our outstanding team of reviewers—Roman Tankelevich, Colorado School of Mines; John Sustersic, Penn State University; Tanya L. Crenshaw, University of Portland; Daniel McCracken, City College of New York; Deborah L. Pollio, Virginia Tech; Keith Hellman, Colorado School of Mines; Tammy VanDeGrift, University of Portland; Melanie Ford, Penn State University—Behrend Campus; Amar Raheja, California State Polytechnic University for their detailed and constructive comments and their valuable insights. We would also like to acknowledge and thank our excellent editorial staff, including Tracy Dunkelberger, Stephanie Sellinger, and Emma Snider, for their help in keeping everything on task. Finally, we would like to thank our outstanding production team, including Eric Arima, Kayla Smith-Tarbox, and Lily Ferguson, for their insight and attention to detail.