PREFACE

Engineers use computers to solve a variety of problems ranging from the evaluation of a simple function to solving a system of nonlinear equations. Thus, C has become the language of choice for many engineers and scientists, not only because it has powerful commands and data structures, but also because it can easily be used for system-level operations. Since C is a language that a new engineer is likely to encounter in a job, it is a good choice for an introduction to computing for engineers. Therefore, this text was written to introduce engineering problem solving with the following objectives:

- to develop a consistent methodology for solving engineering problems;
- to present the fundamental capabilities of C, the language of choice for many practicing engineers and scientists; and
- to illustrate the problem-solving process with C through a variety of interesting engineering examples and applications.

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. Chapters 2 through 7 present the fundamental capabilities of C for solving engineering problems. Chapter 8 is an introduction to object-oriented programming using C++. Object-oriented programming is gaining popularity in many fields of engineering and science, and is likely to be seen in the workplace. Throughout all these chapters, we present a large number of examples from many different engineering and scientific disciplines. The solutions to these examples are developed using the five-step process and ANSI C (and ANSI C++ in Chapter 8), which are the standards developed by the American National Standards Institute.

Changes to the Fourth Edition

- The new theme for this edition is Crime Scene Investigation (CSI). Learning about the technology behind crime scene investigation is not only very interesting, but it provides a number of problems for which we can develop C program solutions.
- Section 1.2 has been rewritten to include discussion on current topics such as cloud computing and kernels.
- A new four-color insert has been added to define an important area of crime scene investigation—biometrics. Biometrics is a term used to describe the physical or behavioral characteristics that can be used to identify a person. The insert includes discussion on fingerprints, face recognition, iris recognition, DNA, and speech recognition.
- Each chapter begins with a photo and a related discussion on a technology used in crime scene investigation. Then, within each chapter after Chapter 1, an associated application section has been added so that in addition to learning all the key features of C, you will also learn about forensic anthropology, face recognition and surveillance video, iris recognition, speech analysis and speech recognition, DNA analysis, fingerprint recognition, and hand recognition. In these application sections, we develop a C solution to a problem related to the crime scene technology.
Preface

- New Modify! problems have been added to each new application.
- The material in Chapter 8 on C++ has been updated to reflect the new C++ standards.

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 computing. These chapters contain the essential topics of mathematical computing, character data, control structures, functions, arrays, pointers, and structures. Students with a background in another computer language should be able to complete this material in less than a 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 table of contents.)

There are three ways to use the text, along with the recommended chapter sections:

- **Introduction to C.** Many freshman courses introduce the student to several computer tools in addition to a language. For these courses, we recommend covering the non-optional sections of Chapters 1 through 5. This material introduces students to the fundamental capabilities of C, and they will be able to write substantial programs using mathematical computations, character data, control structures, 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 non-optional sections of Chapters 1 through 7. This material covers all the fundamental concepts of the C language, including mathematical computations, character data, control structures, functions, arrays, pointers, and structures.

- **Problem solving with C and numerical techniques.** A number of sections included in the text cover common numerical techniques, such as linear interpolation, linear modeling, finding roots of polynomials, and solutions to simultaneous equations. Including these along with the sections on the C language provides a strong combination for students who may need to use numerical techniques in their course work. This coverage would include all sections of Chapters 1 through 7.

Many students may be interested in reading about some of the additional object-oriented features found in C++. We recommend that students cover all non-optional sections of Chapters 1 through 7 before reading Chapter 8.

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. This five-step problem-solving process was developed by the author of this text early in her academic career, and it has been successfully used by the many thousands of students who
were in her classes or used one of her textbooks. This successful process has also been adopted by a number of other authors. The five steps are:

1. **State the problem clearly.**
2. **Describe the input and output information.**
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 that 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. Some examples to illustrate this **wide variety of engineering applications are**

- salinity of sea water
- velocity computation
- amino acid molecular weights
- wind tunnels
- ocean wave interactions
- ozone measurements
- sounding rocket trajectory
- suture packaging
- timber regrowth
- critical path analysis
- weather balloons
- iceberg tracking
- instrumentation reliability
- system stability
- component reliability
- flight simulator wind speeds
- hurricane categories
- molecular weights
- speech signal analysis
- terrain navigation
- electrical circuit analysis
Preface

- power plant data
- cryptography
- temperature distribution
- El Niño–Southern Oscillation
- seismic event detection
- tsunami analysis
- surface wind directions

In addition, each chapter begins with a discussion of some aspect of the new theme. Later in the chapter, we solve a problem that relates to the introductory discussion on the technology behind crime scene investigation. These problems address the following applications:

- forensic anthropology
- face recognition and surveillance video
- iris recognition
- speech analysis
- DNA analysis
- fingerprint recognition
- hand recognition

ANSI C

The statements presented and all programs developed use the C standards developed by the American National Standards Institute. By using ANSI C, students learn to write portable code that can be transferred from one computer system to another.

Software Engineering Concepts

Engineers and scientists are expected to develop and implement user-friendly and reusable computer solutions. Learning software engineering techniques is crucial to successfully developing these computer solutions. Readability and documentation are stressed in the development of programs. Additional topics that relate to software engineering issues 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 several different levels of difficulty. Four types of exercises 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 the material just presented. Most sections are immediately followed by a set of Practice! problems so that students can determine whether 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 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. Selected solutions to some of the Modify! problems are included at the end of the text.

Each chapter ends with two sets of problems. The Short-Answer problems include true/false problems, multiple choice problems, matching problems, syntax problems, fill-in-the-blank problems, memory snapshot problems, program output problems, and program segment analysis problems. Complete solutions to all the Short-Answer problems are included at the end of the text.

The final set of problems in each chapter (except for Chapter 1) are Programming problems. These are new problems that relate to a variety of engineering applications. The level of difficulty ranges from very straightforward to longer project assignments. Each problem requires that the students develop a complete C program or function. Selected solutions to the programming problems are included at the end of the text. Complete solutions to the programming problems are available for instructors.

**Study and Programming Aids**

Margin notes are used to help the reader not only identify the important concepts, but also to 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 notes help students recognize common errors so that they can avoid them. The programming style notes are indicated with a margin note, and the debugging notes are indicated with a bug icon. 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 Summary of the new statements to make the book easier to use as a reference. The combined list of these key terms, along with their definitions, is included in a Glossary at the end of the text. In addition, the inside of the front cover contains common functions and the precedence table; the inside of the back cover contains examples of most of the C statements.

**Optional Numerical Techniques**

Numerical techniques that are commonly used in solving engineering problems are also discussed in the text, and they include interpolation, linear modeling (regression), root finding, 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.

**MATLAB and Visualization**

The visualization of the information related to a problem and its solution is a critical component in understanding and developing the intuition necessary to be a creative engineer. Therefore, we have included a number of plots of data throughout the text to illustrate the relationships of the information needed to solve specific problems. All the plots were
generated using MATLAB, a powerful environment for numerical computations, data analysis, and visualization. We have also included an appendix that shows how to generate a simple plot from data that have been stored in a text file; this text file could be generated with a word processor or it could be generated by a C program.

Appendices

To further enhance reference use, the appendices include a number of important topics. Appendix A contains a discussion of the components in the ANSI C Standard Library. Appendix B presents the ASCII character codes. Appendix C shows how to use MATLAB to plot data from ASCII files; this allows students to generate ASCII files with their C programs and to plot the values using MATLAB.

Nontechnical Skills

The engineer of the twenty-first century needs many skills and capabilities in addition to the technical ones learned in an engineering program. In Chapter 1, we present a brief discussion on some of these nontechnical skills that are so important to engineers. Specifically, we discuss developing both oral and written communications skills, understanding the design/process/ manufacture path that takes an idea and leads to a product, working in interdisciplinary teams, understanding the world marketplace, the importance of synthesis as well as analysis, and the importance of ethics and other societal concerns in engineering solutions. While this text is devoted primarily to teaching problem-solving skills and the C language, we have attempted to tie these other nontechnical topics into many of the problems and discussions in the text.

Additional Resources

All instructor and student resources can be accessed at www.pearsonhighered.com/etter. Here, students can access student data files for the book, and instructors can register for the password-protected Instructor's Resource Center. The IRC contains complete solutions to all of the Programming Projects found at the end of each chapter, and a complete set of PowerPoint lecture slides.

Acknowledgments

A number of people have made significant contributions to this text. Students are always the best judge of “what works” and “what doesn’t work.” I appreciate the feedback from students who had never used the computer when they started this text, to undergraduates who already knew another language, and to graduate students who wanted to use C to do their research analysis. The comments and suggestions from these students greatly improved the text.

A constructive, but critical, review is extremely important in improving a text. The many reviewers who provided this critical guidance included Murali Narayanan (Kansas State University), Kyle Squires (Arizona State University), Amelia Regan (University of California at Irvine), Hyeong-Ah Choi (George Washington University), George Friedman (University of Illinois, Champaign), D. Dandapani (University of Colorado, Colorado Springs), Karl Mathias (Auburn University), William Koffke (Villanova University), Paul Heinemann.
Preface

(Pennsylvania State University), A. S. Hodel (Auburn University), Armando Barreto (Florida International University), Arnold Robbins (Georgia Technology College of Computing), Avelino Gonzalez (University of Central Florida), Thomas Walker (Virginia Polytechnic Institute and State University), Christopher Skelly (Insight Resource Inc.), Betty Barr (The University of Houston), John Cordero (University of Southern California), A. R. Marundarajan (Cal Poly, Pomona), Lawrence Genalo (Iowa State University), Karen Davis (University of Cincinnati), Petros Gheresus (General Motors Institute), Leon Levine (UCLA), Harry Tyrer (University of Missouri, Columbia), Caleb Drake (University of Illinois, Chicago), John Miller (University of Michigan, Dearborn), Elden Heiden (New Mexico State University), Joe Hootman (University of North Dakota), Nazeih Botros (Southern Illinois University), Mark C. Petzold (St. Cloud State University), Ali Saman Tosun (University of Texas at San Antonio), Turgay Korkmaz (University of Texas at San Antonio), Billie Goldstein (Temple University), Mark S. Hutchenreuther (California Polytechnic State University), Frank Friedman (Temple University), and Harold Mitchell Jr. (University of Houston).

The outstanding team at Pearson Education continues to be a delight to work with on my book projects. They include Marcia Horton, Tracy (Dunkelberger) Johnson, Emma Snider, Kayla Smith-Tarbox, and Eric Arima. I want to thank Jeanine Ingber (University of New Mexico) for her contributions as a co-author of the second edition; many of her contributions remain in this fourth edition.

Delores M. Etter
Department of Electrical Engineering
Southern Methodist University
Dallas, TX