I've also taken the opportunity to improve wording and explanations throughout the book. The changes are extensive and painstaking: every sentence has been checked and—if necessary—rewritten.

Although much has changed in this edition, I've tried to retain the original chapter and section numbering as much as possible. Only one chapter (the last one) is entirely new, but many chapters have additional sections. In a few cases, existing sections have been renumbered. One appendix (C syntax) has been dropped, but a new appendix that compares C99 with C89 has been added.

Goals

The goals of this edition remain the same as those of the first edition:

- ***Be clear, readable, and possibly even entertaining.*** Many C books are too concise for the average reader. Others are badly written or just plain dull. I've tried to give clear, thorough explanations, leavened with enough humor to hold the reader's interest.
- ***Be accessible to a broad range of readers.*** I assume that the reader has at least a little previous programming experience, but I don't assume knowledge of a particular language. I've tried to keep jargon to a minimum and to define the terms that I use. I've also attempted to separate advanced material from more elementary topics, so that the beginner won't get discouraged.
- ***Be authoritative without being pedantic.*** To avoid arbitrarily deciding what to include and what not to include, I've tried to cover all the features of the C language and library. At the same time, I've tried to avoid burdening the reader with unnecessary detail.
- ***Be organized for easy learning.*** My experience in teaching C underscores the importance of presenting the features of C gradually. I use a spiral approach, in which difficult topics are introduced briefly, then revisited one or more times later in the book with details added each time. Pacing is deliberate, with each chapter building gradually on what has come before. For most students, this is probably the best approach: it avoids the extremes of boredom on the one hand, or "information overload" on the other.
- ***Motivate language features.*** Instead of just describing each feature of the language and giving a few simple examples of how the feature is used, I've tried to motivate each feature and discuss how it's used in practical situations.
- ***Emphasize style.*** It's important for every C programmer to develop a consistent style. Rather than dictating what this style should be, though, I usually describe a few possibilities and let the reader choose the one that's most appealing. Knowing alternative styles is a big help when reading other people's programs (which programmers often spend a great deal of time doing).
- ***Avoid dependence on a particular machine, compiler, or operating system.*** Since C is available on such a wide variety of platforms, I've tried to avoid