

dependence on any particular machine, compiler, or operating system. All programs are designed to be portable to a wide variety of platforms.

- *Use illustrations to clarify key concepts.* I've tried to put in as many figures as I could, since I think these are crucial for understanding many aspects of C. In particular, I've tried to "animate" algorithms whenever possible by showing snapshots of data at different points in the computation.

What's So Modern about *A Modern Approach*?

One of my most important goals has been to take a "modern approach" to C. Here are some of the ways I've tried to achieve this goal:

- *Put C in perspective.* Instead of treating C as the only programming language worth knowing, I treat it as one of many useful languages. I discuss what kind of applications C is best suited for; I also show how to capitalize on C's strengths while minimizing its weaknesses.
- *Emphasize standard versions of C.* I pay minimal attention to versions of the language prior to the C89 standard. There are just a few scattered references to K&R C (the 1978 version of the language described in the first edition of Brian Kernighan and Dennis Ritchie's book, *The C Programming Language*). Appendix C lists the major differences between C89 and K&R C.
- *Debunk myths.* Today's compilers are often at odds with commonly held assumptions about C. I don't hesitate to debunk some of the myths about C or challenge beliefs that have long been part of the C folklore (for example, the belief that pointer arithmetic is always faster than array subscripting). I've re-examined the old conventions of C, keeping the ones that are still helpful.
- *Emphasize software engineering.* I treat C as a mature software engineering tool, emphasizing how to use it to cope with issues that arise during programming-in-the-large. I stress making programs readable, maintainable, reliable, and portable, and I put special emphasis on information hiding.
- *Postpone C's low-level features.* These features, although handy for the kind of systems programming originally done in C, are not as relevant now that C is used for a great variety of applications. Instead of introducing them in the early chapters, as many C books do, I postpone them until Chapter 20.
- *De-emphasize "manual optimization."* Many books teach the reader to write tricky code in order to gain small savings in program efficiency. With today's abundance of optimizing C compilers, these techniques are often no longer necessary; in fact, they can result in programs that are less efficient.

Q&A Sections

Each chapter ends with a "Q&A section"—a series of questions and answers related to material covered in the chapter. Topics addressed in these sections include: