# [PDF] Pointers On C

## Kenneth Reek - pdf download free book

---


**Books Details:**
Title: Pointers on C
Author: Kenneth Reek
Released: 1997-08-04
Language:
Pages: 636
ISBN: 0673999866
ISBN13: 978-0673999863
ASIN: 0673999866

# CLICK HERE FOR DOWNLOAD

---

**pdf, mobi, epub, azw, kindle**

### Description:

**From the Inside Flap** Why Another C Book?

There are many good C books on the market; why do we need another one? In my ten years of teaching a college-level course on C programming, I have yet to find a book that covers pointers the way I would like. Many books cover pointers in a single chapter dedicated to the topic, often late in the book.

It is not enough to describe the syntax of pointers and show only simple examples of their use. I discuss pointers early and often. I describe their effective use in various contexts and show

programming idioms in which they appear. I discuss related issues of program efficiency versus maintainability. Pointers are a thread that is woven throughout this book. Why are pointers so important? My belief is that pointers are what gives C its power. Pointers allow the C programmer to implement many tasks more efficiently than is possible in other languages, and to perform some tasks, such as accessing the hardware directly, that are impossible in other languages. A thorough knowledge of pointers, then, is a prerequisite to becoming a good C programmer.

However, the power of pointers comes with a price. You can cut wood faster with a chain saw than with a nail file, but the chain saw can injure you a lot more seriously, and more quickly. Pointers are like the chain saw. Used correctly, they can simplify the implementation of an algorithm as well as make it more efficient. Used incorrectly, they can be the cause of errors that exhibit subtle and confusing symptoms and are thus extremely difficult to find. An incomplete understanding of pointers is dangerous because it invariably leads to pain rather than pleasure. This book gives you the depth of knowledge in pointers that you need to avoid the pain. Why Learn C?

Why is the C language still so popular? Historically, industry has embraced C for a number of reasons. Among these are its efficiency; good C programs can be nearly as efficient as assembly language programs, but they are considerably easier to develop. C gives programmers more control over where data is stored and how it is initialized than many languages do. C's lack of "safety net" features also contributes to its efficiency, but increases the likelihood of errors. For example, subscripts to arrays and accesses through pointers are not checked for validity, which saves time but makes it much more important that these features be used correctly. If the language is used with discipline, the potential problems can be avoided.

The rich collection of operators provided in C give the programmer power to efficiently perform low-level computations, such as shifting and masking, without resorting to assembly language. This ability has prompted many to characterize C as being a "high-level" assembly language. However, when needed, C programs can interface easily with assembly language. These characteristics make C a good choice for implementing operating systems and software for embedded controllers.

Another reason for its popularity is its ubiquity. C compilers are widely available for a great number of machines. In addition, the ANSI Standard improves the portability of C programs among different machines.

Finally, C is the foundation upon which C++ is built. C++ provides a different view of program design and implementation than C. Nevertheless, a thorough knowledge of C skills and techniques, such as using pointers and the standard library, will also be useful for the C++ programmer. Who Should Use this Book?

This book is not an introductory text on programming. It is intended for people who already have some programming experience and wish to learn C without being held back by discussions of why loops are important or when to use an if statement.

On the other hand, I assume that the reader has no prior knowledge of C. I cover all of its many aspects. This broad coverage makes the book useful for both students and professionals, that is, for programmers first learning C and more experienced users wishing to improve their command of the language.

The better C++ textbooks concentrate on issues relating to the Object-Oriented (OO) paradigm, such as class design, rather than fundamental C techniques, and rightly so. But C++ is built upon C□the fundamental skills are still important, particularly for those implementing reusable classes. While C++ programmers using this book will be able to skip some familiar material, they will also

find a wealth of useful C tools and techniques. Organization of the Book

The book is organized as a tutorial for people with prior programming experience. It is written in the style of a mentor looking over your shoulder and giving you tips and advice. My goal is to pass along the kind of knowledge and insight that ordinarily takes years of experience with the language to attain. This organization influences the ordering of the material—topics are generally introduced and explained completely in one place. Thus, the book is also useful as a reference.

There are two notable exceptions to this organization. The first is pointers, which are discussed in many different contexts throughout the book. The second is Chapter 1, which gives a quick introduction to the basics of the language. The introduction helps get you started writing simple programs quickly. The topics it presents are covered more thoroughly in subsequent chapters.

The book is more verbose in many areas than other texts, usually to provide the depth in a topic that you would otherwise get only from experience. In addition, I use a few examples that are not often seen in real programs. Though they may be obscure, these examples shed light on interesting aspects of the language. ANSI C

This text describes ANSI C, as defined by ANSI/ISO 9899-1990 ANSI 90 and described by KERN 89. I choose this dialect of C for two reasons: it is the successor to and has essentially replaced the older C (sometimes referred to as Kernighan and Ritchie KERN 78 or K&R C), and it is the foundation upon which C++ is built. All of the examples in this book are written in ANSI C. I will often refer to the ANSI C standards document as simply "the Standard." Typography

Syntax descriptions, such as the following example, and function prototypes are shown on a light gray background to help make them easy to find when you need them later.

if( expression )

statement else

statement

Four typefaces are used in the syntax descriptions. Code that must be written exactly as shown (such as the keyword if in this example) is set in Courier. Abstract descriptions of required code (such as expression above) appear in Courier Italic. Some statements also have optional parts. Code that you must write exactly as shown if you decide to use the optional part (for example, the else keyword) is shown in Courier Bold, and abstract descriptions of optional parts (the second statement above) appear in Courier Bold-Italic. New terms are introduced with Helvetica Italic.

Complete programs are numbered and displayed in the format shown in Program 0.1. The caption gives the title of the program. The filename in which the source code can be found appears beneath the right-hand corner—these files are available from the Addison Wesley Longman web site.

This margin symbol indicates a programming tip. Many of these tips are discussions of good programming techniques—ways to make programs easier to write and easier to read and understand later. Often a little extra effort when a program is first written can yield large time savings later when the program must be modified. Other tips will help you write code that is more compact or efficient.

Other tips deal with software engineering issues. C was designed long before modern principles of software engineering evolved. Thus, some language features and common techniques are now

discouraged by these principles. The issue is often the tradeoffs between the efficiency of a certain construct and its effect on the readability and maintainability of the code. The discussions will give you the background you need to help you decide whether the gain in efficiency justifies the loss of these other qualities.

void function(){

} Program 0.1  Sample program listing  filename.c

Pay close attention when you see this symbol: I am pointing out one of the mistakes that beginning (and sometimes experienced) C programmers often make or something that does not work as you might expect it to. The caution sign makes these hints hard to miss and easy to find later.

This symbol indicates a discussion of an important difference between ANSI C and K&R C. Although most programs written in K&R C should run with only minor modification in ANSI C environments, you might some day run into a pre-ANSI compiler or encounter an older program written for one. The differences will then be important.
Finally, I typeset the book myself using lwroff, a troff clone that I wrote, and I uploaded the resulting PostScript files to the publisher. Thus, the final responsibility for any errors in the book is mine. I would appreciate receiving mail about errors or other correspondence at kar@cs.rit.edu. Chapter Questions and Programming Exercises

Each chapter ends with a selection of questions and programming exercises. The questions range from simple syntax problems to discussions of more complex issues such as tradeoffs between efficiency and maintainability. The programming exercises have been rated for their difficulty:  is easiest and  is most difficult. Many of these exercises have been class tested for many years. The symbol  before the number of a question or exercise means that a solution for it appears in the Appendix. The remaining solutions are found in the Instructor's Guide.

Supplementary Materials

Addison Wesley Longman maintains a World Wide Web site for this book. Its URL is awl/cseng/titles/0-673-99986-6/. The site includes copies of the numbered programs in the book, organized by chapter. The latest errata list is also available. Contact your Addison Wesley Longman representative to get the Instructor's Guide, which contains the answers to the remaining questions and programming exercises.

Software for automated submission and testing of student programs on UNIX systems REEK 89, REEK 96 is available free for educational users via anonymous ftp from ftp.cs.rit in the directory pub/kar/try. Acknowledgments

I cannot possibly list all the people who contributed to this book, yet I would like to acknowledge and thank all of them. My wife Margaret provided abundant encouragement and moral support, and she patiently put up with the disruptions to our lives that resulted from this work.

I would like to thank Professor Warren Carithers, one of my colleagues at RIT, for proofreading the first draft. His careful critique helped me produce a clear, coherent manuscript from a binder full of lecture notes and examples.

Many thanks to the students in my C Programming Seminar for their assistance in finding typos and suggesting improvements and for putting up with a textbook in draft form. You were my guinea pigs, and your reactions to what I wrote provided valuable feedback that helped me improve the text.

I am indebted to Steve Allan, Bill Appelbe, Richard C. Detmer, Roger Eggen, Joanne Goldenberg, Dan Hinton, Dan Hirschberg, Keith E. Jolly, Joseph F. Kent, Masoud Milani, Steve Summit, and Kanupriya Tewary, who reviewed the book before publication. Their suggestions and insights were a great help in refining my presentation.

Finally, I'd like to express my gratitude to my Editor at Addison-Wesley, Ms. Deborah Lafferty, and my Production Editor, Ms. Amy Willcutt. It is because of these people that this text is a book rather than a computer manual. They both gave me many valuable suggestions and encouraged me to change a lot of typography that I thought was fine. Now that I have seen the result, I realize that they were right. Now it is time to begin. Above all, I hope you have fun learning C!

Churchville, NY Kenneth A. Reek
kar@cs.rit.edu
References

ANSI 90 1990. American National Standard for Programming Languages□C. ANSI/ISO 9899-1990. New York, NY: American National Standards Institute. KERN 89 Kernighan, Brian and Dennis Ritchie. 1989. The C Programming Language, Second Edition. Englewood Cliffs, NJ: Prentice Hall. KERN 78 Kernighan, Brian and Dennis Ritchie. 1978. The C Programming Language. Englewood Cliffs, NJ: Prentice Hall. REEK 89 Reek, Kenneth A. 1989. "The TRY System□or□How to Avoid Testing Student Programs." Proceedings of the Twentieth SIGCSE Technical Symposium on Computer Science Education Volume 21: 112-116. REEK 96 Reek, Kenneth A. 1996. "A Software Infrastructure to Support Introductory Computer Science Courses." Proceedings of the Twenty Seventh SIGCSE Technical Symposium on Computer Science Education Volume 28: 125-129.
0673999866P04062001

**From the Back Cover**

**Pointers On C** brings the power of pointers to your C programs.

Designed for professionals and advanced students, **Pointers on C** provides a comprehensive resource for those needing in-depth coverage of the C programming language. An extensive explanation of pointer basics and a thorough exploration of their advanced features allows programmers to incorporate the power of pointers into their C programs. Complete coverage, detailed explanations of C programming idioms, and thorough discussion of advanced topics makes **Pointers on C** a valuable tutorial and reference for students and professionals alike.

**Highlights:**

- Provides complete background information needed for a thorough understanding of C.
- Covers pointers thoroughly, including syntax, techniques for their effective use and common programming idioms in which they appear.
- Compares different methods for implementing common abstract data structures.
- Offers an easy, conversant writing style to clearly explain difficult topics, and contains numerous illustrations and diagrams to help visualize complex concepts.
- Includes Programming Tips, discussing efficiency, portability, and software engineering issues, and warns of common pitfalls using Caution! Sections.
- Describes every function on the standard C library.

0673999866B04062001

- Title: Pointers on C
- Author: Kenneth Reek
- Released: 1997-08-04
- Language:
- Pages: 636
- ISBN: 0673999866
- ISBN13: 978-0673999863
- ASIN: 0673999866