深入浅出C语言（影印版）

Head First C

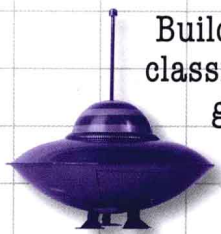
Discover the secrets of the C coding gurus

Learn how make can change your life

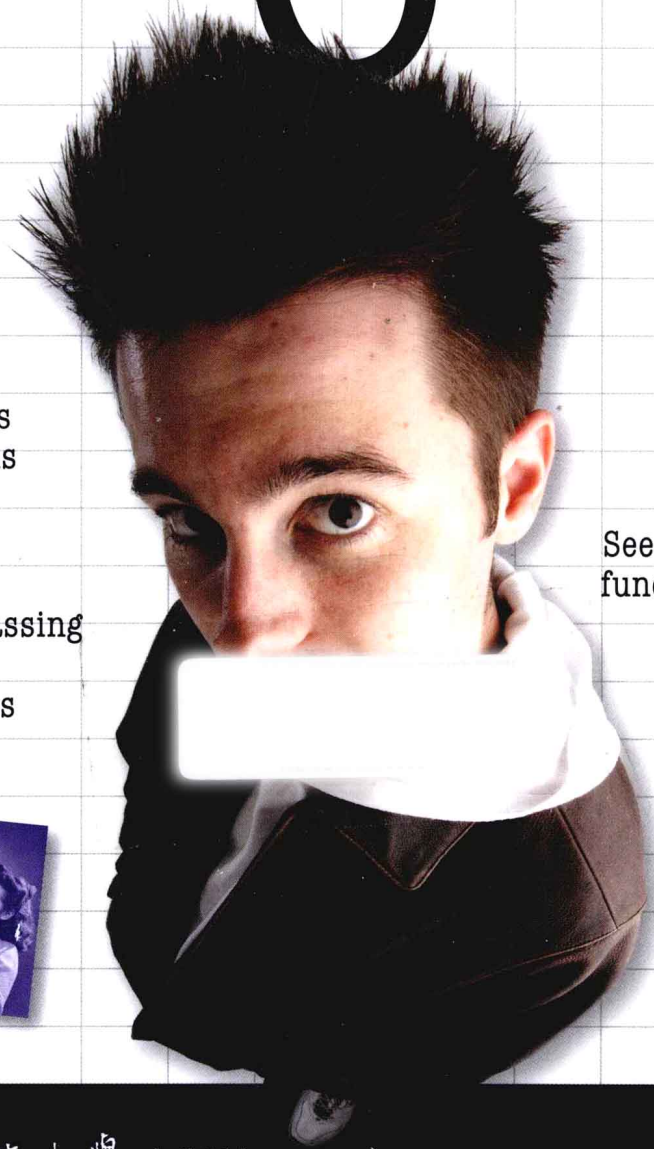
Avoid embarrassing pointer mistakes

See how variadic functions helped Sue be more flexible

Fool around in the C Standard Library



Build a retro classic arcade game



O'REILLY® 東南大學出版社

David Griffiths &
Dawn Griffiths 著

深入浅出C语言（影印版）

Head First C

David Griffiths
Dawn Griffiths

O'REILLY®

Beijing • Cambridge • Farnham • Köln • Sebastopol • Tokyo

O'Reilly Media, Inc. 授权东南大学出版社出版

东南大学出版社

图书在版编目（CIP）数据

深入浅出C语言：英文/（美）格里菲思（Griffiths, D.）著.—影印本.—南京：东南大学出版社，2013.1

书名原文：Head First C

ISBN 978-7-5641-3895-0

I. ①深… II. ①格… III. ①C语言－程序设计－英文 IV. ①TP312

中国版本图书馆CIP数据核字（2012）第273572号

江苏省版权局著作权合同登记

图字：10-2012-5号

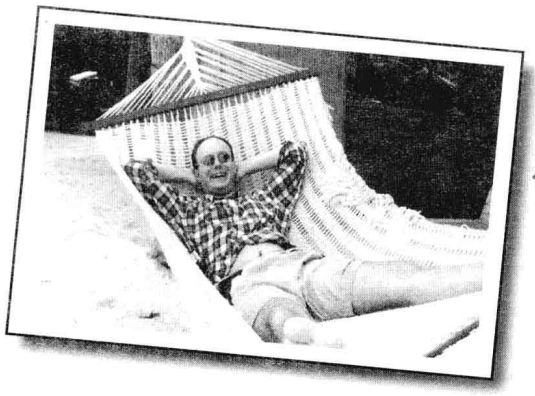
深入浅出C语言（影印版）

出版发行：东南大学出版社
地　　址：南京四牌楼2号　　邮编：210096
出 版 人：江建中
网　　址：http://www.seupress.com
电子邮件：press@seupress.com
印　　刷：扬中市印刷有限公司
开　　本：787毫米×980毫米　12开本
印　　张：52.5
字　　数：878千字
版　　次：2013年1月第1版
印　　次：2013年1月第1次印刷
书　　号：ISBN 978-7-5641-3895-0
定　　价：98.00元（册）

本社图书若有印装质量问题，请直接与营销部联系。电话（传真）：025-83791830

To Dennis Ritchie (1941–2011), the father of C.

Authors of Head First C



David Griffiths

Dawn Griffiths

David Griffiths began programming at age 12, when he saw a documentary on the work of Seymour Papert. At age 15, he wrote an implementation of Papert's computer language LOGO. After studying pure mathematics at university, he began writing code for computers and magazine articles for humans. He's worked as an agile coach, a developer, and a garage attendant, but not in that order. He can write code in over 10 languages and prose in just one, and when not writing, coding, or coaching, he spends much of his spare time traveling with his lovely wife—and coauthor—Dawn.

Before writing *Head First C*, David wrote two other Head First books: *Head First Rails* and *Head First Programming*.

You can follow David on Twitter at *http://twitter.com/dogriffiths*.

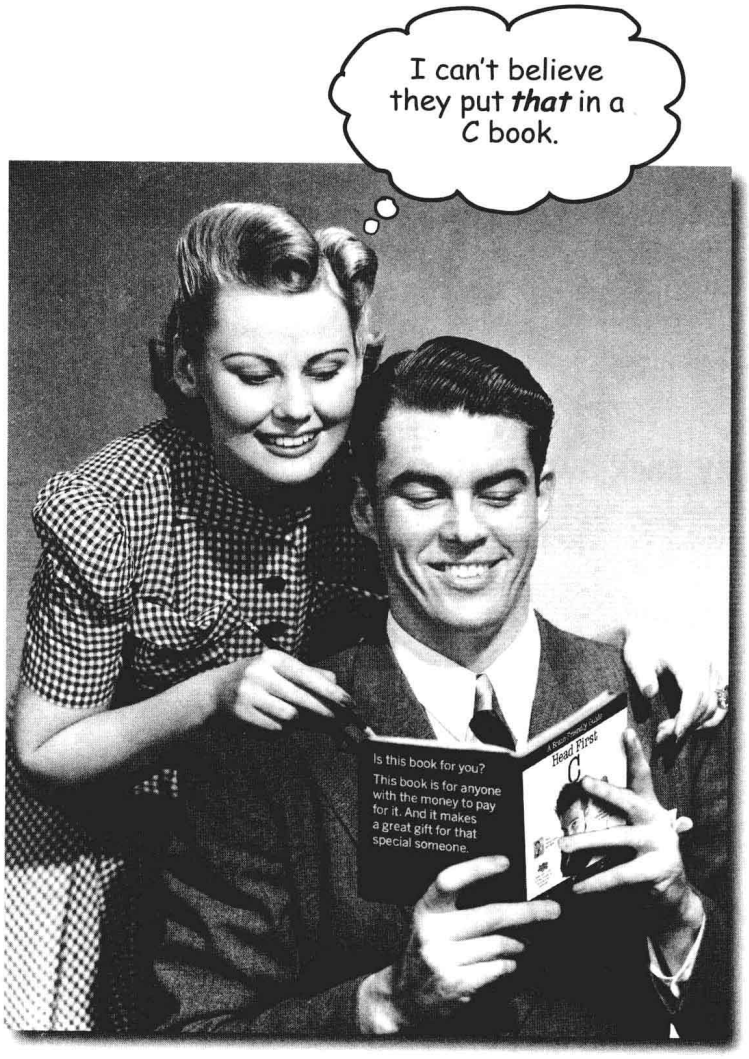
Dawn Griffiths started life as a mathematician at a top UK university, where she was awarded a first-class honors degree in mathematics. She went on to pursue a career in software development and has over 15 years experience working in the IT industry.

Before joining forces with David on *Head First C*, Dawn wrote two other Head First books (*Head First Statistics* and *Head First 2D Geometry*) and has also worked on a host of other books in the series.

When Dawn's not working on Head First books, you'll find her honing her Tai Chi skills, running, making bobbin lace, or cooking. She also enjoys traveling and spending time with her husband, David.

how to use this book

Intro



In this section, we answer the burning question:
"So why DID they put that in a C book?"

Who is this book for?

If you can answer "yes" to all of these:

1. Do you already know how to program in another programming language?
2. Do you want to master C, create the next big thing in software, make a small fortune, and retire to your own private island?
3. Do you prefer actually doing things and applying the stuff you learn over listening to someone in a lecture rattle on for hours on end?

OK, maybe that one's a little far-fetched. But, you gotta start somewhere, right?

this book is for you.

Who should probably back away from this book?

If you can answer "yes" to any of these:

1. Are you looking for a quick introduction or reference book to C?
2. Would you rather have your toenails pulled out by 15 screaming monkeys than learn something new? Do you believe a C book should cover *everything* and if it bores the reader to tears in the process, then so much the better?

this book is ***not*** for you.

[Note from Marketing: this book is for anyone with a credit card... we'll accept a check, too.]

We know what you're thinking

"How can *this* be a serious C book?"

"What's with all the graphics?"

"Can I actually *learn* it this way?"

We know what your *brain* is thinking

Your brain craves novelty. It's always searching, scanning, *waiting* for something unusual. It was built that way, and it helps you stay alive.

So what does your brain do with all the routine, ordinary, normal things you encounter? Everything it *can* to stop them from interfering with the brain's *real* job—recording things that *matter*. It doesn't bother saving the boring things; they never make it past the "this is obviously not important" filter.

How does your brain *know* what's important? Suppose you're out for a day hike and a tiger jumps in front of you—what happens inside your head and body?

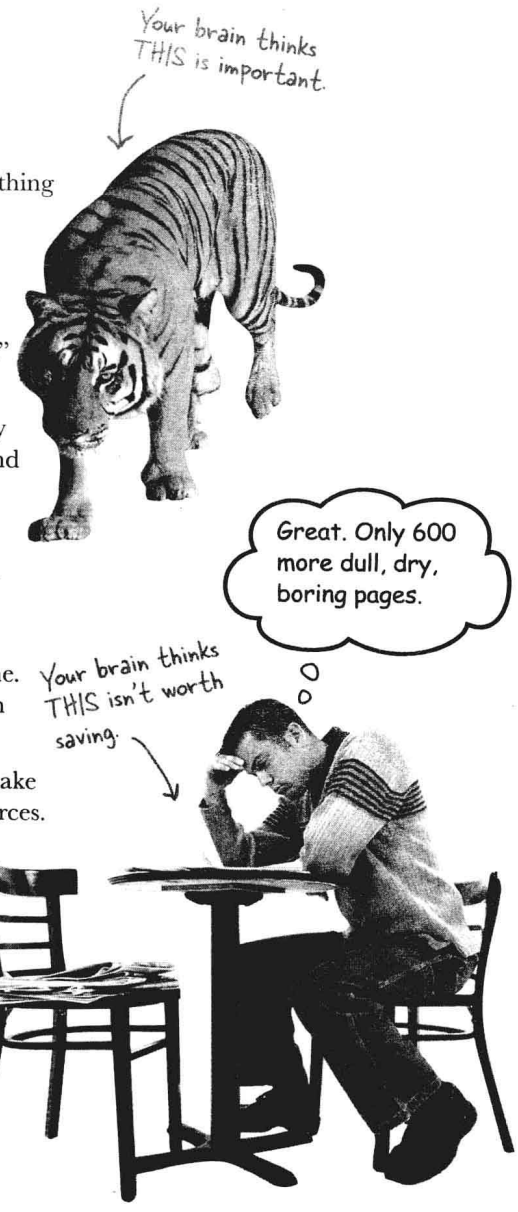
Neurons fire. Emotions crank up. *Chemicals surge.*

And that's how your brain knows...

This must be important! Don't forget it!

But imagine you're at home or in a library. It's a safe, warm, tiger-free zone. You're studying. Getting ready for an exam. Or trying to learn some tough technical topic your boss thinks will take a week, ten days at the most.

Just one problem. Your brain's trying to do you a big favor. It's trying to make sure that this *obviously* unimportant content doesn't clutter up scarce resources. Resources that are better spent storing the really *big* things. Like tigers. Like the danger of fire. Like how you should never have posted those party photos on your Facebook page. And there's no simple way to tell your brain, "Hey brain, thank you very much, but no matter how dull this book is, and how little I'm registering on the emotional Richter scale right now, I really *do* want you to keep this stuff around."



We think of a "Head First" reader as a learner.

So what does it take to *learn* something? First, you have to *get* it, then make sure you don't *forget* it. It's not about pushing facts into your head. Based on the latest research in cognitive science, neurobiology, and educational psychology, *learning* takes a lot more than text on a page. We know what turns your brain on.

Some of the Head First learning principles:

Make it visual. Images are far more memorable than words alone, and make learning much more effective (up to 89% improvement in recall and transfer studies). It also makes things more understandable. **Put the words within or near the graphics** they relate to, rather than on the bottom or on another page, and learners will be up to *twice* as likely to solve problems related to the content.

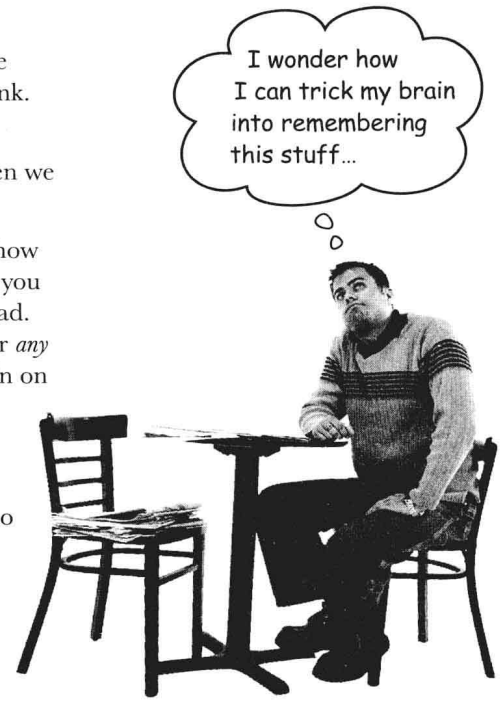
Use a conversational and personalized style. In recent studies, students performed up to 40% better on post-learning tests if the content spoke directly to the reader, using a first-person, conversational style rather than taking a formal tone. Tell stories instead of lecturing. Use casual language. Don't take yourself too seriously. Which would *you* pay more attention to: a stimulating dinner-party companion, or a lecture?

Get the learner to think more deeply. In other words, unless you actively flex your neurons, nothing much happens in your head. A reader has to be motivated, engaged, curious, and inspired to solve problems, draw conclusions, and generate new knowledge. And for that, you need challenges, exercises, and thought-provoking questions, and activities that involve both sides of the brain and multiple senses.

Get—and keep—the reader's attention. We've all had the "I really want to learn this, but I can't stay awake past page one" experience. Your brain pays attention to things that are out of the ordinary, interesting, strange, eye-catching, unexpected. Learning a new, tough, technical topic doesn't have to be boring. Your brain will learn much more quickly if it's not.

Touch their emotions. We now know that your ability to remember something is largely dependent on its emotional content. You remember what you care about. You remember when you *feel* something. No, we're not talking heart-wrenching stories about a boy and his dog. We're talking emotions like surprise, curiosity, fun, "what the…?", and the feeling of "I rule!" that comes when you solve a puzzle, learn something everybody else thinks is hard, or realize you know something that "I'm more technical than thou" Bob from Engineering *doesn't*.

Metacognition: thinking about thinking



If you really want to learn, and you want to learn more quickly and more deeply, pay attention to how you pay attention. Think about how you think. Learn how you learn.

Most of us did not take courses on metacognition or learning theory when we were growing up. We were *expected* to learn, but rarely *taught* to learn.

But we assume that if you're holding this book, you really want to learn how to program in C. And you probably don't want to spend a lot of time. If you want to use what you read in this book, you need to *remember* what you read. And for that, you've got to *understand* it. To get the most from this book, or *any* book or learning experience, take responsibility for your brain. Your brain on *this* content.

The trick is to get your brain to see the new material you're learning as Really Important. Crucial to your well-being. As important as a tiger. Otherwise, you're in for a constant battle, with your brain doing its best to keep the new content from sticking.

So just how *DO* you get your brain to treat programming like it was a hungry tiger?

There's the slow, tedious way, or the faster, more effective way. The slow way is about sheer repetition. You obviously know that you *are* able to learn and remember even the dullest of topics if you keep pounding the same thing into your brain. With enough repetition, your brain says, "This doesn't *feel* important to him, but he keeps looking at the same thing *over* and *over* and *over*, so I suppose it must be."

The faster way is to do ***anything that increases brain activity,*** especially different *types* of brain activity. The things on the previous page are a big part of the solution, and they're all things that have been proven to help your brain work in your favor. For example, studies show that putting words *within* the pictures they describe (as opposed to somewhere else in the page, like a caption or in the body text) causes your brain to try to makes sense of how the words and picture relate, and this causes more neurons to fire. More neurons firing = more chances for your brain to *get* that this is something worth paying attention to, and possibly recording.

A conversational style helps because people tend to pay more attention when they perceive that they're in a conversation, since they're expected to follow along and hold up their end. The amazing thing is, your brain doesn't necessarily *care* that the "conversation" is between you and a book! On the other hand, if the writing style is formal and dry, your brain perceives it the same way you experience being lectured to while sitting in a roomful of passive attendees. No need to stay awake.

But pictures and conversational style are just the beginning...

Here's what WE did:

We used ***pictures***, because your brain is tuned for visuals, not text. As far as your brain's concerned, a picture really *is* worth a thousand words. And when text and pictures work together, we embedded the text *in* the pictures because your brain works more effectively when the text is *within* the thing it refers to, as opposed to in a caption or buried in the body text somewhere.

We used ***redundancy***, saying the same thing in *different* ways and with different media types, and *multiple senses*, to increase the chance that the content gets coded into more than one area of your brain.

We used concepts and pictures in ***unexpected*** ways because your brain is tuned for novelty, and we used pictures and ideas with at least *some* ***emotional*** *content*, because your brain is tuned to pay attention to the biochemistry of emotions. That which causes you to *feel* something is more likely to be remembered, even if that feeling is nothing more than a little ***humor***, ***surprise***, or ***interest.***

We used a personalized, ***conversational style***, because your brain is tuned to pay more attention when it believes you're in a conversation than if it thinks you're passively listening to a presentation. Your brain does this even when you're *reading*.

We included more than 80 ***activities***, because your brain is tuned to learn and remember more when you ***do*** things than when you *read* about things. And we made the exercises challenging-yet-doable, because that's what most people prefer.

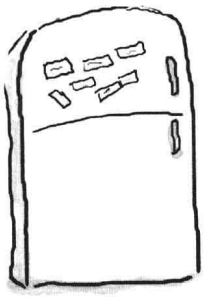
We used ***multiple learning styles***, because *you* might prefer step-by-step procedures, while someone else wants to understand the big picture first, and someone else just wants to see an example. But regardless of your own learning preference, *everyone* benefits from seeing the same content represented in multiple ways.

We include content for ***both sides of your brain***, because the more of your brain you engage, the more likely you are to learn and remember, and the longer you can stay focused. Since working one side of the brain often means giving the other side a chance to rest, you can be more productive at learning for a longer period of time.

And we included ***stories*** and exercises that present ***more than one point of view,*** because your brain is tuned to learn more deeply when it's forced to make evaluations and judgments.

We included ***challenges***, with exercises, and by asking ***questions*** that don't always have a straight answer, because your brain is tuned to learn and remember when it has to *work* at something. Think about it—you can't get your *body* in shape just by *watching* people at the gym. But we did our best to make sure that when you're working hard, it's on the *right* things. That ***you're not spending one extra dendrite*** processing a hard-to-understand example, or parsing difficult, jargon-laden, or overly terse text.

We used ***people***. In stories, examples, pictures, etc., because, well, *you're* a person. And your brain pays more attention to *people* than it does to *things*.



Cut this out and stick it on your refrigerator.

Here's what YOU can do to bend your brain into submission

So, we did our part. The rest is up to you. These tips are a starting point; listen to your brain and figure out what works for you and what doesn't. Try new things.

1. **Slow down. The more you understand, the less you have to memorize.**

   Don't just *read*. Stop and think. When the book asks you a question, don't just skip to the answer. Imagine that someone really *is* asking the question. The more deeply you force your brain to think, the better chance you have of learning and remembering.

2. **Do the exercises. Write your own notes.**

   We put them in, but if we did them for you, that would be like having someone else do your workouts for you. And don't just *look* at the exercises. **Use a pencil.** There's plenty of evidence that physical activity *while* learning can increase the learning.

3. **Read "There Are No Dumb Questions."**

   That means all of them. They're not optional sidebars, ***they're part of the core content!*** Don't skip them.

4. **Make this the last thing you read before bed. Or at least the last challenging thing.**

   Part of the learning (especially the transfer to long-term memory) happens *after* you put the book down. Your brain needs time on its own, to do more processing. If you put in something new during that processing time, some of what you just learned will be lost.

5. **Talk about it. Out loud.**

   Speaking activates a different part of the brain. If you're trying to understand something, or increase your chance of remembering it later, say it out loud. Better still, try to explain it out loud to someone else. You'll learn more quickly, and you might uncover ideas you hadn't known were there when you were reading about it.

6. **Drink water. Lots of it.**

   Your brain works best in a nice bath of fluid. Dehydration (which can happen before you ever feel thirsty) decreases cognitive function.

7. **Listen to your brain.**

   Pay attention to whether your brain is getting overloaded. If you find yourself starting to skim the surface or forget what you just read, it's time for a break. Once you go past a certain point, you won't learn faster by trying to shove more in, and you might even hurt the process.

8. **Feel something.**

   Your brain needs to know that this *matters*. Get involved with the stories. Make up your own captions for the photos. Groaning over a bad joke is *still* better than feeling nothing at all.

9. **Write a lot of code!**

   There's only one way to learn to program in C: **write a lot of code**. And that's what you're going to do throughout this book. Coding is a skill, and the only way to get good at it is to practice. We're going to give you a lot of practice: every chapter has exercises that pose a problem for you to solve. Don't just skip over them—a lot of the learning happens when you solve the exercises. We included a solution to each exercise—don't be afraid to **peek at the solution** if you get stuck! (It's easy to get snagged on something small.) But try to solve the problem before you look at the solution. And definitely get it working before you move on to the next part of the book.

Read me

This is a learning experience, not a reference book. We deliberately stripped out everything that might get in the way of learning whatever it is we're working on at that point in the book. And the first time through, you need to begin at the beginning, because the book makes assumptions about what you've already seen and learned.

We assume you're new to C, but not to programming.

We assume that you've already done some programming. Not a lot, but we'll assume you've already seen things like loops and variables in some other language, like JavaScript. C is actually a pretty advanced language, so if you've never done any programming *at all*, then you might want to read some other book before you start on this one. We'd suggest starting with *Head First Programming*.

You need to install a C compiler on your computer.

Throughout the book, we'll be using the *Gnu Compiler Collection* (`gcc`) because it's free and, well, we think it's just a pretty darned good compiler. You'll need to make sure you have `gcc` installed on your machine. The good news is, if you have a *Linux* computer, then you should already have `gcc`. If you're using a Mac, you'll need to install the Xcode/Developer tools. You can either download these from the Apple *App Store* or by downloading them from Apple. If you're on a Windows machine, you have a couple options. *Cygwin* (*http://www.cygwin.com*) gives you a complete simulation of a *UNIX* environment, including `gcc`. But if you want to create programs that will work on Windows plain-and-simple, then you might want to install the *Minimalist GNU for Windows* (MingW) from *http://www.mingw.org*.

All the code in this book is intended to run across *all* these operating systems, and we've tried hard not to write anything that will only work on one type of computer. Occasionally, there will be some differences, but we'll make sure to point those out to you.

We begin by teaching some basic C concepts, and then we start putting C to work for you right away.

We cover the fundamentals of C in Chapter 1. That way, by the time you make it all the way to Chapter 2, you are creating programs that actually do something real, useful, and—gulp!—fun. The rest of the book then builds on your C skills, turning you from *C newbie* to *coding ninja master* in no time.

The activities are NOT optional.

The exercises and activities are not add-ons; they're part of the core content of the book. Some of them are to help with memory, some are for understanding, and some will help you apply what you've learned. ***Don't skip the exercises.***

The redundancy is intentional and important.

One distinct difference in a Head First book is that we want you to *really* get it. And we want you to finish the book remembering what you've learned. Most reference books don't have retention and recall as a goal, but this book is about *learning*, so you'll see some of the same concepts come up more than once.

The examples are as lean as possible.

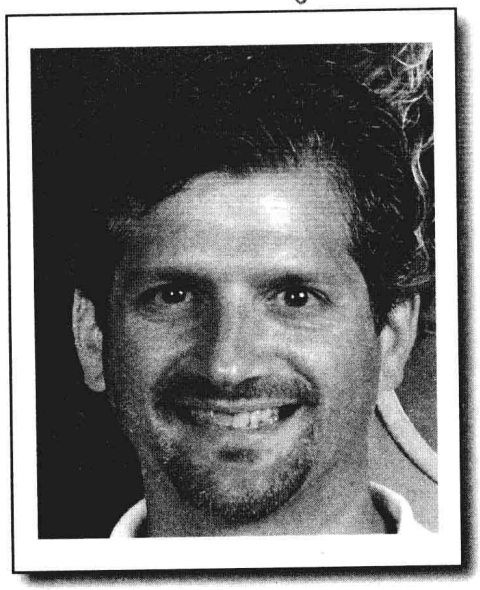
Our readers tell us that it's frustrating to wade through 200 lines of an example looking for the two lines they need to understand. Most examples in this book are shown within the smallest possible context, so that the part you're trying to learn is clear and simple. Don't expect all of the examples to be robust, or even complete—they are written specifically for learning, and aren't always fully functional.

The Brain Power exercises don't have answers.

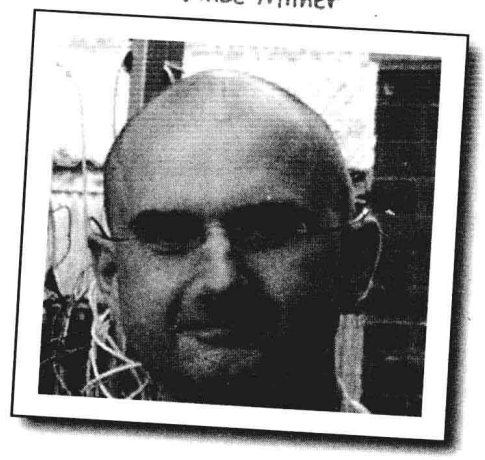
For some of them, there is no right answer, and for others, part of the learning experience of the Brain Power activities is for you to decide if and when your answers are right. In some of the Brain Power exercises, you will find hints to point you in the right direction.

The technical review team

Dave Kitabjian



Vince Milner



Technical reviewers:

Dave Kitabjian has two degrees in electrical and computer engineering and about 20 years of experience consulting, integrating, architecting, and building information system solutions for clients from Fortune 500 firms to high-tech startups. Outside of work, Dave likes to play guitar and piano and spend time with his wife and three kids.

Vince Milner has been developing in C (and many other languages) on a wide variety of platforms for over 20 years. When not studying for his master's degree in mathematics, he can be found being beaten at board games by six-year-olds and failing to move house.

Acknowledgments

Our editor:

Many thanks to **Brian Sawyer** for asking us to write this book in the first place. Brian believed in us every step of the way, gave us the freedom to try out new ideas, and didn't panic too much when deadlines loomed.

Brian Sawyer

The O'Reilly team:

A big thank you goes to the following people who helped us out along the way: **Karen Shaner** for her expert image-hunting skills and for generally keeping the wheels oiled; **Laurie Petrycki** for keeping us well fed and well motivated while in Boston; **Brian Jepson** for introducing us to the wonderful world of the Arduino; and the **early release team** for making early versions of the book available for download. Finally, thanks go to **Rachel Monaghan** and the production team for expertly steering the book through the production process and for working so hard behind the scenes. You guys are awesome.

Family, friends, and colleagues:

We've made a lot of friends on our Head First journey. A special thanks goes to **Lou Barr**, **Brett McLaughlin**, and **Sanders Kleinfeld** for teaching us so much.

David: My thanks to **Andy Parker**, **Joe Broughton**, **Carl Jacques**, and **Simon Jones** and the many other friends who have heard so little from me whilst I was busy scribbling away.

Dawn: Work on this book would have been a lot harder without my amazing support network of family and friends. Special thanks go to **Mum and Dad**, **Carl**, **Steve**, **Gill**, **Jacqui**, **Joyce**, and **Paul**. I've truly appreciated all your support and encouragement.

The without-whom list:

Our technical review team did a truly excellent job of keeping us straight and making sure what we covered was spot on. We're also incredibly grateful to all the people who gave us feedback on early releases of the book. We think the book's much, much better as a result.

Finally, our thanks to **Kathy Sierra** and **Bert Bates** for creating this extraordinary series of books.

Safari® Books Online

Safari Books Online (*www.safaribooksonline.com*) is an on-demand digital library that delivers expert content in both book and video form from the world's leading authors in technology and business. Technology professionals, software developers, web designers, and business and creative professionals use Safari Books Online as their primary resource for research, problem solving, learning, and certification training.

Safari Books Online offers a range of product mixes and pricing programs for organizations, government agencies, and individuals. Subscribers have access to thousands of books, training videos, and prepublication manuscripts in one fully searchable database from publishers like O'Reilly Media, Prentice Hall Professional, Addison-Wesley Professional, Microsoft Press, Sams, Que, Peachpit Press, Focal Press, Cisco Press, John Wiley & Sons, Syngress, Morgan Kaufmann, IBM Redbooks, Packt, Adobe Press, FT Press, Apress, Manning, New Riders, McGraw-Hill, Jones & Bartlett, Course Technology, and dozens more. For more information about Safari Books Online, please visit us online.