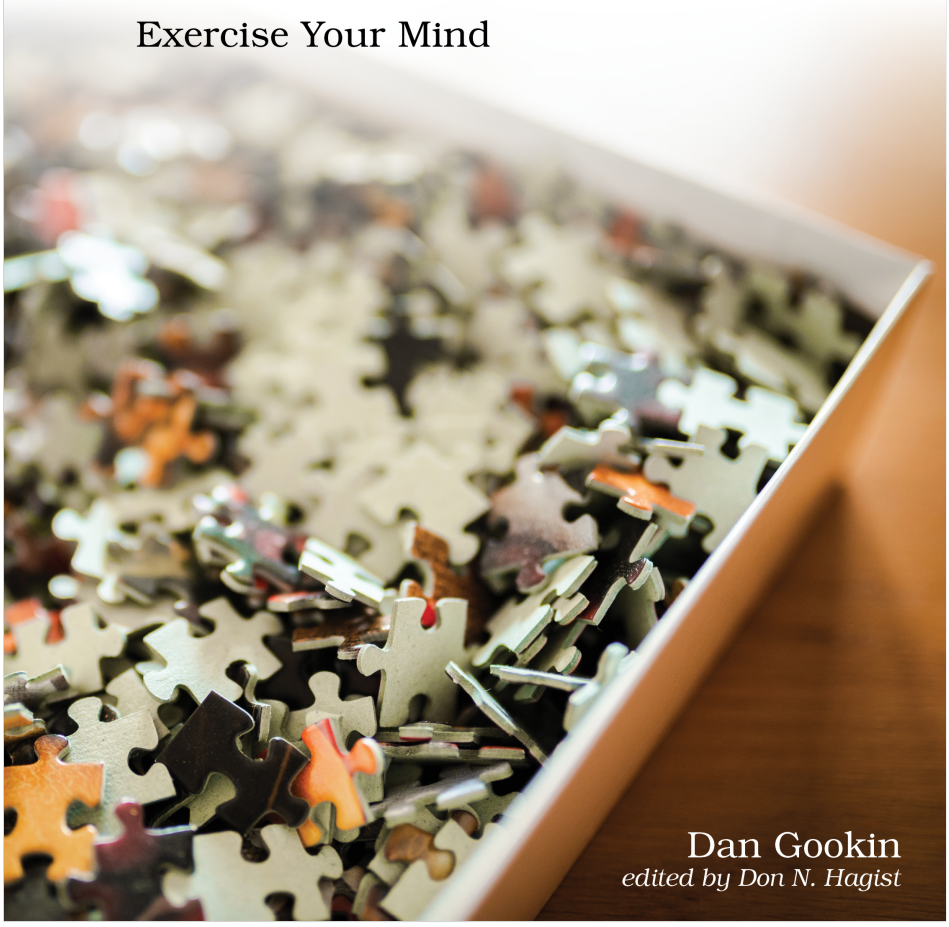
The Pragmatic Programmers

C Brain Teasers

Exercise Your Mind



Dan Gookin
edited by Don N. Hagist

This extract shows the online version of this title, and may contain features (such as hyperlinks and colors) that are not available in the print version.

For more information, or to purchase a paperback or ebook copy, please visit https://www.pragprog.com.

Preface

Shocking all the experts who continue to predict its demise, the C Programming language is stronger than ever. Taught in universities and used by developers around the world, C's syntax and structure are borrowed by other major languages. Often, those languages have their foundation in C. Today, C is used to maintain operating systems, create high-end graphics drivers, program microcontrollers, code for embedded systems, and more.

In this book, you'll find 25 puzzles that explore the potential and possibilities of the C language. These puzzles range from easy tasks to the complex and tricky. Some puzzles showcase how limited C can be—and how to get around these limitations. The goal is to showcase various aspects of C and programming in general.

C programmers, from beginners to advanced, will gain understanding from the puzzles presented in this book. Whether you're just starting out or have been coding for a while, the insights and surprises offered here will entertain and delight you. The goal is to make you a better programmer.

How to Use This Book

This book contains 25 programming puzzles. Some perform specific tasks, while others may attempt to do something that doesn't quite work. All of the code is written in C and adheres to the C11 standard. No additional files or libraries are required. The programs run in the text mode environment, specifically, under Linux in a terminal window.

For each puzzle, your job is to guess what happens. You can predict the output, guess what the program is trying to accomplish, or identify a potential problem. The answer is revealed on the pages that follow each puzzle, along with an insight into understanding the code, what it does, and how or why it fails to do what you might think it does. The point is to learn more about C programming, witness a few tricks, and put your new knowledge to work in your own programs.

Here's a sample puzzle:

```
int puts(const char *s);

#define lineout(a) puts(a)
#define end return(0)

int main()
{
        lineout("Hello there!");
        end;
}
```

Can you guess the output from this code? Yes, it's C source code, though I've taken some liberties with the way things are expressed. If you can guess the output, can you identify the liberties I've taken in the source code? When presented in the text, your task is to do so before you turn the page to see my explanation and further exploration of the puzzle.

Because I'm a nice guy, here are the answers:

- The output is the string "Hello there!" followed by a newline. This is the output generated by the puts() function, normally declared within the stdio.h header—but this header is missing in the source code!
- Instead of including the entire stdio.h header, I write the puts() function prototype. This is a legal move, as you can prototype any function you've created. But, here I just looked up the main page definition for puts() and copied it into my source code. Properly prototyped, the function works just fine. (Remember that the function's mechanics are stored in the library, not in the header file.)
- Two defines create the unusual statements found in the main() function. The first defines a function lineout() equivalent to the puts() function. The a in both represents the function's argument. So lineout() replaces puts() in the main() function, carrying out the same task.
- The second define assigns the return(0) statement to the word end. The result is the main() function's statements appear alien—very non-C-like. Still, the preprocessor replaces both lineout() and end with the proper C statements.

This funky construction is one of the things I adore about the C language. While as a programmer your goal should be to write easily readable text, you can add your own quirks to the language just to keep things interesting. But

on a grander scale, you can use these tricks to help simplify some complex operations. So, while my attempts obfuscated the code, you can use the same tools to make a program more readable. And hopefully, you'll have fun while doing so.

Required Tools

The code presented in this book is quick and to the point. You can copy it directly from the text (type it in) or you can obtain the puzzles from my GitHub account: github.com/dangookin/C_Brain_Teasers.

All programs run in the terminal window, which is where I do my coding. I use Vim to write the source code. I build (compile and link) with clang version 14.0.0. The command line code I use to build the program is clang -Wall followed by the source code filename. This command creates the program file named a.out, which you can run by typing ./a.out at the command prompt.

The terminal window is available in Linux and on the Macintosh under macOS. For Windows, you can obtain the Linux runtime environment, which provides the same functionality. See https://learn.microsoft.com/en-us/windows/wsl/install.

To install clang in Linux, use the package manager for your distro. For example, if the distro uses apt, the command is: sudo apt-get install clang.

If you're into Integrated Development Environments (IDEs), I recommend Visual Studio Code. It's available free at https://code.visualstudio.com/.

My coding style is close to the original K&R (Kernighan and Ritchie). I use only the traditional C comments. Remember that in C, whitespace is ignored. Feel free to use your own coding style if you desire to transcribe the examples.

Remember that some of the programs do not run properly as presented. Mistakes are made on purpose to drive home a point that's explained in this text. I understand that readers who are concerned about this approach probably aren't reading this Preface, yet I write this warning anyway.

Contacting the Author

I'm happy to provide feedback or offer advice related to my C programming books. You can contact me at mailto:dgookin@wambooli.com. That's my real email address, and I try to respond to all my mail, especially specific questions regarding my books. I cannot write code for you, though I can try to help with problems you encounter related to this book.

The support page for this book can be found on my C For Dummies website at https://c-for-dummies.com/cbrainteasers. I run a blog on that site, where you can learn more about C programming with weekly lessons and monthly exercises. The site has been up for over ten years, so it covers a lot of ground. Use the search box to locate topics of interest.

Have fun solving the puzzles!

Dan Gookin, March 2024