Quick answers to common problems

D Cookbook

Discover the advantages of programming in D with over 100 incredibly effective recipes

Foreword by Andrei Alexandrescu, Author of *The D Programming Language*

Adam D. Ruppe

[PACKT] open source*
PUBLISHING
community experience distilled

D Cookbook

Discover the advantages of programming in D with over 100 incredibly effective recipes

Adam D. Ruppe

PUBLISHING

open source
community experience distilled

BIRMINGHAM - MUMBAI

D Cookbook

First published: May 2014

Production Reference: 1190514

Published by Packt Publishing Ltd.
Livery Place
35 Livery Street
Birmingham B3 2PB, UK.

ISBN 978-1-78328-721-5

www.packtpub.com

Cover Image by Dennis Ruppe (dennis.ruppe@gmail.com)

Credits

Author
Adam D. Ruppe

Reviewers
Andrei Alexandrescu
Brad Anderson
Maxim Fomin
Kai Nacke

Commissioning Editor
Sam Birch

Acquisition Editor
Sam Birch

Content Development Editor
Sriram Neelakantan

Technical Editors
Krishnaveni Haridas
Monica John
Edwin Moses
Shiny Poojary

Copy Editors
Alisha Aranha
Roshni Banerjee
Mradula Hegde

Project Coordinator
Amey Sawant

Proofreaders
Simran Bhogal
Paul Hindle

Indexer
Priya Subramani

Production Coordinators
Manu Joseph
Nitesh Thakur

Cover Work
Manu Joseph

Foreword

There is an immediacy and a delicious sense of urgency running through Adam's book that makes the very notion of its foreword almost offensive. "Let's go implement some great ideas", the book seems to rejoice at every page; "I know you don't have the patience but read me first, this may help." I wouldn't want to hold you much with a fluffy, needless opener for a book that in turn frames itself as a prelude to many enjoyable hours of spinning code. I'll try to keep this short and to the point—much in the spirit of the book itself.

D Cookbook aims at enabling you to get work done using D, and it is written from the perspective of one who's clearly walking the walk. I know that Adam has leveraged D for years in his consulting gigs, but even if I didn't, I would have inferred this easily. He writes in the factual, no-nonsense tone of the senior engineer who wants to bring a n00b up to speed so they can get good work done together. Adam's use of "you" and "we" nicely orients himself and the reader toward solving a problem together. He's not coy to just tell the reader what to do to accomplish a task, but never comes across as patronizing. Simple explanations pepper the recipes, and there's always an implied "here's something I tried and works well, you may find that useful" lurking in the subtext.

The book covers a variety of topics that appear to be only loosely connected: what do (to quote a few consecutive chapter titles) "Ranges", "Integration" (with platforms and other languages), "Resource Management", and "Wrapped Types" have in common? Usefulness, that's what. Such topics, and everything else that the book sets out to explain, are likely to be important in real-world D applications. Of these, a few are "canon". At the other extreme there'd be borderline apocryphal stuff such as the *Kernel in D* chapter. Finally, the bulk of it is annotated folklore (idioms and patterns known by D's early adopters but not yet by the wider community), mixed with the author's own insights for good measure. Such a collection of relevant, high-impact topics is difficult to find collected, let alone in book format. You should read this book if you want to ramp up to using D in industrial-strength applications.

Adam's style is refreshing for someone like me; I've been involved in a mix of language design and language advocacy for years now, both fields of considerable subjectivity and fervor. Adam's dispassionate take on language advocacy is a breath of fresh air. His passion is expended on building great systems, and the language is but a means to that end. If Adam likes a language feature, he does primarily because he can use it to good effect, and proceeds to illustrate that. If, on the contrary, he finds a shortcoming, he simply discusses possible workarounds; that, and the missing lamentations, wonderfully imply that the point of it all is to get work done. "There is one disadvantage", Adam notes in a sidebar, "to operator overloading being implemented with templates, though: the operator overload functions cannot be virtual." Before even finishing that sentence, I've evoked in my mind enough pros and cons for a lively talk show debate. He's unfazed: "To work around this, write the overload implementation as a final method which merely forwards the request to a virtual method."

Last but not least, I took pleasure with the varying "zoom level" of the book. Like a philosopher who also knows his way around a welding machine, Adam can discuss esoteric code generation topics and show code disassembly, sometimes within the same chapter (see for example, "Code Generation") and all in style, while illustrating a good point. Wherever you dwell on the high-level/low-level continuum, it's likely you'll find ways to expand your range by reading *D Cookbook*.

Many years ago, while in the military, I learned to shoot the famed Kalashnikov AK47. I was bad at shooting from the hip (which is odd because everybody in the movies is great at it) until one day I learned a trick that was doing the rounds—wrap the weapon's strap tightly around the left arm at the elbow. The extra tension increases hand stability. That hack worked great; yet it was not to be found in any doctrine or manual, and in fact I couldn't find much about it today on the Internet. *D Cookbook* reminds me of that hack—it contains advice that's hard to find in the official documentation, and of immense practical utility. If you want to work in D, you'll find this book a great companion.

Andrei Alexandrescu, PhD

Research Scientist, Facebook
Author of *The D Programming Language*
San Francisco, CA, 12th May 2014

About the Author

Adam D. Ruppe is a professional software developer living in Watertown, New York. He started programming PCs in high school, writing assembly language, and later C and C++, using the Digital Mars compiler to build programs based on MS DOS on a hand-me-down computer. Programming in the DOS environment with the slow computer gave him early practical experience in low-level and efficient code—skills he carries on developing today.

After finishing school, he started doing web programming—initially with PHP. While he'd make it work, he often found himself longing for the good old days. One day, he decided to check back with the vendor of his old compiler and discovered the D programming language (well before it reached 1.0!).

He was enamored with it and used it to write some games, and then started writing web libraries to use it for work too, to replace PHP. He found success in this endeavor in early 2009.

Combining his pioneering spirit with his blend of low-level and high-level programming experience, he was able to forge ahead with D, taking it to places many people didn't believe possible.

About the Reviewers

Andrei Alexandrescu coined the colloquial term "modern C++", which is used today to describe a collection of important C++ styles and idioms. His book on the topic, *Modern C++ Design: Generic Programming and Design Patterns Applied* (Addison-Wesley, 2001), revolutionized C++ programming and produced a lasting influence not only on subsequent work on C++, but also on other languages and systems. With Herb Sutter, he is also the co-author of *C++ Coding Standards: 101 Rules, Guidelines, and Best Practices (*Addison-Wesley Professional, 2010*)*. He has garnered a solid reputation in both industrial and academic circles through his varied work on libraries and applications, as well as research in machine learning and natural language processing. From 2006, he worked on the D programming language together with Walter Bright, the inventor and initial implementer of the language. He co-designed many important features of D, authored a large part of D's standard library, and wrote the book *The D Programming Language* (Addison-Wesley Professional, 2010). Andrei holds a PhD in Computer Science from the University of Washington and a B.Sc. in Electrical Engineering from University Politehnica of Bucharest. He works as a research scientist for Facebook.

Brad Anderson is a computer programmer living in Salt Lake City. He has been writing software professionally for over 10 years and is currently a Lead Developer at Phoenix Project Management Systems.

Maxim Fomin is a programmist who is currently living and working in St. Petersburg, Russia. Coming with a background in other languages, he quickly recognized D programming language for its convenience, efficiency, and power synthesis. He helped a company to apply D language in writing software in an area of his professional interest—Finance.

I would like to thank my family for helping and encouraging me in times of difficulties and pessimism. I would also like to thank all the mentors that I've had over the years. Without their assistance, I would not have acquired knowledge and skills that I possess today.

Kai Nacke is the current maintainer of LDC, the LLVM-based D compiler. He has a strong interest in compiler construction and is also a contributor to the LLVM framework. In 1998, he received his Master of Computer Science degree. He is an IT architect at IBM and has over 10 years of experience in architecturing solutions and developing custom applications.

Table of Contents

Preface

The D programming language's popularity is growing rapidly. With its seamless blending of high-level convenience with low-level power and efficiency, D is suitable for tackling almost any programming task productively. This book comes out of years of experience of using D in the real world and closely following the language and libraries' development. It will also help you get up to speed with this exciting language and burgeoning ecosystem.

What this book covers

Chapter 1, Core Tasks, will get you started with D and cover the tasks you can perform with D's core language features that differ from other popular programming languages.

Chapter 2, Phobos – The Standard Library, introduces you to the standard D library to perform common tasks, including generating random numbers, writing a network client and server, and performing type conversions.

Chapter 3, Ranges, covers the range concept, which is central to D algorithms. Ranges allow you to write and consume generators, views on various collections, and perform generic transformations of data.

Chapter 4, Integration, explores integrating D with the outside world, including creating Windows-based applications, using C libraries, and extending C++ applications with D.

Chapter 5, Resource Management, discusses how to manage memory and other resources in D, including tips on why, when, and how to use the garbage collector effectively.

Chapter 6, Wrapped Types, dives into the world of user-defined types, showing you how to extend and restrict types via cheap wrapper abstractions.

Chapter 7, Correctness Checking, shows how to use D's bug-hunting features such as testing, assertions, and documentation, and the correct way to do conditional compilation.

Chapter 8, Reflection, teaches you about the rich introspection capabilities D provides, including tips learned through years of experience which stretch the limits of the language.

Chapter 9, Code Generation, demonstrates several techniques to automate the creation of new code to write efficient, generic, and specialized code, including a primer on creating your own mini languages inside D.

Chapter 10, Multitasking, introduces you to the options D offers for concurrency and parallelism.

Chapter 11, D for Kernel Coding, will get you started with writing bare metal code in D, stripping out the runtime library to say hello directly through the PC's video hardware and then handling interrupts sent back by the keyboard with D's low-level features.

Chapter 12, Web and GUI Programming, showcases some of the libraries I've written over the years that show how to make a dynamic website and desktop graphics windows while discussing my practical experience from writing these libraries, which will give you a leg up when you write your own code.

Appendix, Addendum, briefly shows how to use D on ARM processors, including systems without an operating system, and other small topics that didn't fit elsewhere in the book.

What you need for this book

You need to have a Windows or Mac PC that is capable of running the DMD compiler, which is available at `http://dlang.org/`.

Who this book is for

This book is for programmers who want to continue their professional development by learning more about D. Whether you are looking at D for the first time or have used it before and want to learn more, this book has something to offer you.

Conventions

In this book, you will find a number of styles of text that distinguish between different kinds of information. Here are some examples of these styles, and an explanation of their meaning.

Code words in text, database table names, folder names, filenames, file extensions, pathnames, dummy URLs, user input, and Twitter handles are shown as follows: "Add a struct to `test.d`, which uses `alias this` to activate subtyping."

A block of code is set as follows:

```
import project.foo;; // disambiguate with project.foo
import bar; // you can disambiguate calls with the name bar
```

Any command-line input or output is written as follows:

```
coffimplib myfile.lib
```

New terms and **important words** are shown in bold. Words that you see on the screen, in menus or dialog boxes for example, appear in the text like this: "Running the program will print **Hello, world!** in green text on a red background."

[Warnings or important notes appear in a box like this.]

[Tips and tricks appear like this.]

Reader feedback

Feedback from our readers is always welcome. Let us know what you think about this book—what you liked or may have disliked. Reader feedback is important for us to develop titles that you really get the most out of.

To send us general feedback, simply send an e-mail to `feedback@packtpub.com`, and mention the book title via the subject of your message.

If there is a topic that you have expertise in and you are interested in either writing or contributing to a book, see our author guide on `www.packtpub.com/authors`.

Customer support

Now that you are the proud owner of a Packt book, we have a number of things to help you to get the most from your purchase.

Downloading the example code

You can download the example code files for all Packt books you have purchased from your account at `http://www.packtpub.com`. If you purchased this book elsewhere, you can visit `http://www.packtpub.com/support` and register to have the files e-mailed directly to you.