

## Programming in Haskell

Haskell is one of the leading languages for teaching functional programming, enabling students to write simpler and cleaner code, and to learn how to structure and reason about programs.

This introduction is ideal for beginners: it requires no previous programming experience and all concepts are explained from first principles with the aid of carefully chosen examples. Each chapter includes a series of exercises, ranging from the straightforward to extended projects, along with suggestions for further reading on more advanced topics. The presentation is clear and simple, and benefits from having been refined and class-tested over several years.

### Features:

- Powerpoint slides for each chapter freely available for instructors and students from the book's website
- Solutions to exercises, and examination questions (with solutions) available to instructors
- All the code in the book is fully compliant with the latest release of Haskell, and can be downloaded from the web.
- Written by a leading Haskell researcher and instructor, well known for his teaching skills
- Can be used with courses, or as a stand-alone text for self-learning

Graham Hutton has worked in four of the leading centres for research and teaching on functional programming. He has more than 15 years of experience in functional programming research, during which time he has published more than 30 research articles, chaired the Haskell Workshop, and edited a special issue on Haskell of the *Journal of Functional Programming*. He also has more than 10 years' experience in teaching Haskell, and in promoting the use of functional programming in the curriculum.

Cambridge University Press  
978-0-521-87172-3 - Programming in Haskell  
Graham Hutton  
Frontmatter  
[More information](#)

---

---

# Programming in Haskell

Graham Hutton  
*University of Nottingham*



Cambridge University Press  
978-0-521-87172-3 - Programming in Haskell  
Graham Hutton  
Frontmatter  
[More information](#)

---

CAMBRIDGE UNIVERSITY PRESS

Cambridge, New York, Melbourne, Madrid, Cape Town, Singapore, São Paulo

Cambridge University Press

The Edinburgh Building, Cambridge CB2 2RU, UK

Published in the United States of America by Cambridge University Press, New York

[www.cambridge.org](http://www.cambridge.org)

Information on this title: [www.cambridge.org/9780521871723](http://www.cambridge.org/9780521871723)

© G. Hutton 2007

This publication is in copyright. Subject to statutory exception and to the provisions of relevant collective licensing agreements, no reproduction of any part may take place without the written permission of Cambridge University Press.

First published 2007

Printed in the United Kingdom at the University Press, Cambridge

*A catalogue record for this publication is available from the British Library*

*Library of Congress Cataloguing in Publication data*

ISBN-13 978-0-521-87172-3 hardback

ISBN-10 0-521-87172-7 hardback

ISBN-13 978-0-521-69269-4 paperback

ISBN-10 0-521-69269-5 paperback

---

Cambridge University Press has no responsibility for the persistence or accuracy of URLs for external or third-party internet websites referred to in this publication, and does not guarantee that any content on such websites is, or will remain, accurate or appropriate.

---

Cambridge University Press  
978-0-521-87172-3 - Programming in Haskell  
Graham Hutton  
Frontmatter  
[More information](#)

---

*For Annette, Callum and Tom*

## Contents

Preface	<i>page xi</i>
<b>Chapter 1</b>   Introduction	1
1.1 Functions	1
1.2 Functional programming	2
1.3 Features of Haskell	4
1.4 Historical background	6
1.5 A taste of Haskell	6
1.6 Chapter remarks	9
1.7 Exercises	9
<b>Chapter 2</b>   First steps	10
2.1 The Hugs system	10
2.2 The standard prelude	10
2.3 Function application	12
2.4 Haskell scripts	13
2.5 Chapter remarks	16
2.6 Exercises	16
<b>Chapter 3</b>   Types and classes	17
3.1 Basic concepts	17
3.2 Basic types	18
3.3 List types	20
3.4 Tuple types	20
3.5 Function types	21
3.6 Curried functions	21
3.7 Polymorphic types	23
3.8 Overloaded types	23
3.9 Basic classes	24
3.10 Chapter remarks	28
3.11 Exercises	28
<b>Chapter 4</b>   Defining functions	30
4.1 New from old	30
4.2 Conditional expressions	31
4.3 Guarded equations	31
4.4 Pattern matching	32
4.5 Lambda expressions	34
4.6 Sections	36
4.7 Chapter remarks	36
4.8 Exercises	37

<b>Chapter 5</b>	<b>List comprehensions</b>	38
5.1	Generators	38
5.2	Guards	39
5.3	The <i>zip</i> function	40
5.4	String comprehensions	41
5.5	The Caesar cipher	42
5.6	Chapter remarks	46
5.7	Exercises	46
<b>Chapter 6</b>	<b>Recursive functions</b>	48
6.1	Basic concepts	48
6.2	Recursion on lists	49
6.3	Multiple arguments	52
6.4	Multiple recursion	53
6.5	Mutual recursion	53
6.6	Advice on recursion	55
6.7	Chapter remarks	59
6.8	Exercises	59
<b>Chapter 7</b>	<b>Higher-order functions</b>	61
7.1	Basic concepts	61
7.2	Processing lists	62
7.3	The <i>foldr</i> function	64
7.4	The <i>foldl</i> function	66
7.5	The composition operator	68
7.6	String transmitter	69
7.7	Chapter remarks	72
7.8	Exercises	72
<b>Chapter 8</b>	<b>Functional parsers</b>	74
8.1	Parsers	74
8.2	The parser type	75
8.3	Basic parsers	75
8.4	Sequencing	76
8.5	Choice	78
8.6	Derived primitives	78
8.7	Handling spacing	81
8.8	Arithmetic expressions	82
8.9	Chapter remarks	85
8.10	Exercises	85
<b>Chapter 9</b>	<b>Interactive programs</b>	87
9.1	Interaction	87
9.2	The input/output type	88
9.3	Basic actions	88

9.4 Sequencing	89
9.5 Derived primitives	90
9.6 Calculator	91
9.7 Game of life	94
9.8 Chapter remarks	97
9.9 Exercises	97
<hr/>	
<b>Chapter 10</b>   Declaring types and classes	99
10.1 Type declarations	99
10.2 Data declarations	100
10.3 Recursive types	102
10.4 Tautology checker	105
10.5 Abstract machine	109
10.6 Class and instance declarations	111
10.7 Chapter remarks	114
10.8 Exercises	114
<hr/>	
<b>Chapter 11</b>   The countdown problem	116
11.1 Introduction	116
11.2 Formalising the problem	117
11.3 Brute force solution	119
11.4 Combining generation and evaluation	120
11.5 Exploiting algebraic properties	121
11.6 Chapter remarks	123
11.7 Exercises	123
<hr/>	
<b>Chapter 12</b>   Lazy evaluation	124
12.1 Introduction	124
12.2 Evaluation strategies	125
12.3 Termination	128
12.4 Number of reductions	129
12.5 Infinite structures	130
12.6 Modular programming	132
12.7 Strict application	134
12.8 Chapter remarks	137
12.9 Exercises	137
<hr/>	
<b>Chapter 13</b>   Reasoning about programs	139
13.1 Equational reasoning	139
13.2 Reasoning about Haskell	140
13.3 Simple examples	141
13.4 Induction on numbers	142
13.5 Induction on lists	145
13.6 Making append vanish	146
13.7 Compiler correctness	150
13.8 Chapter remarks	154

13.9 Exercises	154
<b>Appendix A</b>   Standard prelude	156
A.1 Classes	156
A.2 Logical values	157
A.3 Characters and strings	158
A.4 Numbers	159
A.5 Tuples	160
A.6 Maybe	160
A.7 Lists	160
A.8 Functions	164
A.9 Input/output	164
<b>Appendix B</b>   Symbol table	166
<i>Bibliography</i>	167
<i>Index</i>	169

---

## Preface

... there are two ways of constructing a software design: One way is to make it so simple that there are *obviously* no deficiencies and the other way is to make it so complicated that there are no *obvious* deficiencies. The first method is far more difficult.

*Tony Hoare, 1980 ACM Turing Award Lecture*

This book is about an approach to programming in which simplicity, clarity, and elegance are the key goals. More specifically, it is an introduction to the functional style of programming, using the language Haskell.

The functional style is quite different to that promoted by most current languages, such as Java, C++, C, and Visual Basic. In particular, most current languages are closely linked to the underlying hardware, in the sense that programming is based upon the idea of changing stored values. In contrast, Haskell promotes a more abstract style of programming, based upon the idea of applying functions to arguments. As we shall see, moving to this higher-level leads to considerably simpler programs, and supports a number of powerful new ways to structure and reason about programs.

The book is primarily aimed at students studying computing science at university level, but is also appropriate for a broader spectrum of readers who would like to learn about programming in Haskell. No previous programming experience is required or assumed, and all the concepts are explained from first principles, with the aid of carefully chosen examples.

The version of Haskell used in the book is Haskell 98, the standard version of the language, for which the recently published definition is the culmination of fifteen years of work by its designers. As this is an introductory text, we do not attempt to cover all aspects of Haskell and its associated libraries. Around half of the book is dedicated to introducing the main features of the language, while the other half comprises examples and case studies of programming of Haskell. Each chapter includes a series of exercises, and suggestions for further reading on more advanced and specialist topics.

The book is based upon course material that has been refined and class-tested over many years at the University of Nottingham. Most of the material

from the book can be covered in twenty hours of lectures, supported by approximately forty hours of private study, practical sessions in a supervised laboratory, and take-home programming courseworks. However, additional time would be required to cover some of the later chapters in more detail, along with some of the later programming examples.

The website for the book provides a range of supporting material, including Powerpoint slides for each chapter, and Haskell code for each of the extended examples. Instructors can also obtain model answers to the exercises for each chapter, together with a large collection of exam questions and their model answers, by emailing [solutions@cambridge.org](mailto:solutions@cambridge.org).

### Acknowledgements

The Foundations of Programming group at the University of Nottingham is an excellent environment in which to do research and teaching on functional programming. I am grateful to the University for providing a sabbatical to start work on this book; all the students and tutors on my Haskell courses for their feedback; and Thorsten Altenkirch, Neil Ghani, Mark Jones (now at Portland State), Conor McBride, and Henrik Nilsson in the FOP group for our many discussions about functional ideas and how to present them.

I would also like to thank David Tranah and Dawn Preston for their editorial work at Cambridge University Press; Mark Jones for the Hugs interpreter for Haskell; Ralf Hinze and Andres Löh for the lhs2TeX system for typesetting Haskell; Rik van Geldrop and Jaap van der Woude for their feedback on using drafts of the book; Kees van den Broek, Frank Heitmann, and Bill Tonkin for pointing out errors; Ian Bayley and the anonymous reviewers for useful comments; and Joel Wright for timing the countdown programs.

Graham Hutton  
Nottingham, 2006