

Undergraduate Topics in Computer Science


Series Editor

Ian Mackie, University of Sussex, Brighton, UK


Advisory Editors

Samson Abramsky , Department of Computer Science, University of Oxford, Oxford, UK

Chris Hankin , Department of Computing, Imperial College London, London, UK

Mike Hinchey , Lero – The Irish Software Research Centre, University of Limerick, Limerick, Ireland

Dexter C. Kozen, Department of Computer Science, Cornell University, Ithaca, NY, USA

Andrew Pitts , Department of Computer Science and Technology, University of Cambridge, Cambridge, UK

Hanne Riis Nielson , Department of Applied Mathematics and Computer Science, Technical University of Denmark, Kongens Lyngby, Denmark

Steven S. Skiena, Department of Computer Science, Stony Brook University, Stony Brook, NY, USA

Iain Stewart , Department of Computer Science, Durham University, Durham, UK

‘Undergraduate Topics in Computer Science’ (UTiCS) delivers high-quality instructional content for undergraduates studying in all areas of computing and information science. From core foundational and theoretical material to final-year topics and applications, UTiCS books take a fresh, concise, and modern approach and are ideal for self-study or for a one- or two-semester course. The texts are all authored by established experts in their fields, reviewed by an international advisory board, and contain numerous examples and problems, many of which include fully worked solutions.

The UTiCS concept relies on high-quality, concise books in softback format, and generally a maximum of 275–300 pages. For undergraduate textbooks that are likely to be longer, more expository, Springer continues to offer the highly regarded Texts in Computer Science series, to which we refer potential authors.

More information about this series at <http://www.springer.com/series/7592>

Noel Kalicharan

Julia - Bit by Bit

Programming for Beginners

Noel Kalicharan
Department of Computing and IT
University of the West Indies
St. Augustine, Trinidad and Tobago

ISSN 1863-7310 ISSN 2197-1781 (electronic)
Undergraduate Topics in Computer Science
ISBN 978-3-030-73935-5 ISBN 978-3-030-73936-2 (eBook)
<https://doi.org/10.1007/978-3-030-73936-2>

© The Editor(s) (if applicable) and The Author(s), under exclusive license to Springer Nature Switzerland AG 2021
This work is subject to copyright. All rights are solely and exclusively licensed by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors, and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, expressed or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Dedication

James & Clara

Claudette & Samuel

Jeff & Jenny

Margaret & Stephen

Kenrick & Debbie

Jennifer & Andrew

Anushka & Michael

Saskia & Vaishnavi

Special Thanks

Hubert Dupont

Shellyann Sooklal

For their meticulous, insightful and helpful comments on the manuscript. Their eye for detail was truly impressive. Each brought their special, but different, strengths to bear, making this a better book than it would have been without their input.

Preface

Julia—Bit by Bit attempts to teach computer programming to the complete beginner using *Julia*—a relatively new programming language. Created in 2009 by Jeff Bezanson, Stefan Karpinski, Viral B. Shah and Alan Edelman, *Julia* was launched in 2012. Their goal? "To create a free language that was both high-level and fast." Since its launch, *Julia* has undergone several version changes. As of November 9, 2020, it had matured to Version 1.5.3.

The book assumes you have no knowledge whatsoever about programming. And if you are worried that you are not good at high-school mathematics, don't be. It is a myth that you must be good at mathematics to learn programming. In this book, a knowledge of primary school mathematics is all that is required—basic addition, subtraction, multiplication, division, finding the percentage of some quantity, finding an average or the larger of two quantities.

Some of our most outstanding students over the last forty years have been people with little mathematics background from all walks of life—politicians, civil servants, sports people, housewives, secretaries, clerical assistants, artists, musicians and teachers. On the other hand, we've had professionals like engineers and scientists who didn't do as well as might be expected. So it's not about how "qualified" you are.

What *will* be an asset is the ability to think logically or to follow a logical argument. If you are good at presenting convincing arguments, you will probably be a good programmer. Even if you aren't, programming is the perfect vehicle for learning logical thinking skills. You should learn it for these skills even if you never intend to become a serious programmer.

The main goal of this book is to teach fundamental programming principles using *Julia*, one of the fastest growing programming languages in the world today. *Julia* can be classified as a "modern" language, possessing many features not available in more popular languages like C and Java.

Best of all, *Julia* is easy to learn. In fact, I would go so far as to say that, of all the many languages I have learnt and taught over the last forty years, *Julia* is the easiest to learn. This is particularly important for someone learning programming for the first time. You can concentrate on acquiring problem-solving skills without being overwhelmed by the language. I've known many students who got turned off learning programming because they found the basics of the language too difficult to grasp.

Julia strips away the "fluff" of most languages, the "overhead" you need to write even the simplest programs. It's not fussy about things like semi-colons or having to "declare" the type of every variable you need to use. You just use it the way you want—*Julia* will figure out the type for you. But if you really *want* *Julia* to enforce "typing", it can do that as well.

Nevertheless, this book is as much about teaching basic problem-solving principles as it is about teaching *Julia*. Remember, a language is useless if you can't use it to solve a problem. But once you learn the *principles* well, they can be applied to any language.

Chapter 1 gives an overview of the programming process. It shows you how to write your first *Julia* program and introduces some of the basic building blocks needed to write programs.

Chapter 2 is all about *numbers*—integers, floating-point, operators, expressions—how to work with them and how to print them. It also explains how to write programs that use *sequence logic*—statements are executed one after the other, from first to last.

Chapter 3 shows how to write programs which can make decisions. It explains how to use `if` and `if...else` statements.

Chapter 4 explains the notion of ‘looping’ and how to use this powerful programming idea to solve more interesting problems. Looping is implemented using `for` and `while` statements. We also explain how to read data from a file and write results to a file.

Chapter 5 formally treats with functions. These enable a (large) program to be broken up into smaller manageable units but which work together to solve a given problem.

Chapter 6 is devoted to Characters and Strings. These present some difficulty in other languages but, in *Julia*, we can work with them as seamlessly as we do numbers.

Chapter 7 tackles the nemesis of many would-be programmers—array processing. However, this is significantly easier in *Julia* than other languages. Master array processing and you would add to your repertoire a tool that will significantly increase the range of problems you can solve.

Chapter 8 is mainly about sorting and searching techniques. Sorting puts data in an order that can be searched more quickly/easily, and makes it more palatable for human consumption.

Chapter 9 introduces *structures*. These enable us to group data in a form that can be manipulated more easily than a unit.

Chapter 10 deals with two useful data structures—dictionaries and sets. These enable us to solve certain kinds of problems more easily and conveniently than we can without them.

The first step in becoming a good programmer is learning the syntax rules of the programming language. This is the easy part and many people mistakenly believe that this makes them a programmer. They get carried away by the cosmetics— they learn the *features* of a language without learning how to use them to solve problems. Of course, you must learn *some* features. But it is far better to learn a few features and be able to use them to solve many problems rather than learn many features but can’t use them to solve anything. For this reason, this book emphasizes solving many problems from just a few features.

This book is intended for anyone who is learning programming for the first time, regardless of age or institution. The presentation is based on our experience that many people (though not all) have difficulty learning programming. To try and overcome this, we use an approach which provides clear examples, detailed explanations of very basic concepts and numerous interesting problems (not just artificial exercises whose only purpose is to illustrate some language feature).

While computer programming is essentially a mental activity and you *can* learn a fair amount of programming from just *reading* the book, it is important that you “get your hands dirty” by writing and running programs. One of life’s thrills is to write your first program and get it to run successfully on a computer. Don’t miss out on it.

But do not stop there. The only way to learn programming well is to write programs to solve new problems. The end-of-chapter exercises are a very rich source of problems, a result of the author’s more than 40 years in the teaching of programming.

Thank you for taking the time to read this book. I hope your venture into programming is a successful and enjoyable one.

Noel Kalicharan

Contents

Chapter 1	Elementary Concepts	1
	1.1 Programs, Languages and Compilers	1
	1.2 How a Computer Solves a Problem	3
	1.2.1 Define the Problem	3
	1.2.2 Analyze the Problem	3
	1.2.3 Develop an Algorithm to Solve the Problem	4
	1.2.3.1 Data and Variables	4
	1.2.3.2 Example – Develop the Algorithm	5
	1.2.4 Write the Program for the Algorithm	5
	1.2.5 Test and Debug the Program	7
	1.2.6 Document the Program	8
	1.2.7 Maintain the Program	8
	1.3 How a Computer Executes a Program	8
	1.4 Data Types	9
	1.5 Characters	10
	1.6 Welcome to Julia Programming	11
	1.7 A Program With Input	12
	1.8 Writing Output with <code>print/println</code>	13
	1.8.1 The Newline Character, <code>\n</code> (backslash n)	14
	1.8.2 <code>println()</code>	15
	1.8.3 Escape Sequences	15
	1.9 Print the Value of a Variable	16
	1.10 Comments	17
	1.11 Julia Basics	18
	1.11.1 The Julia Alphabet	18
	1.11.2 Julia Tokens	18
	1.11.3 Reserved Words	20
	1.11.4 Identifiers	20
	1.11.5 Some Naming Conventions	21
	Exercises 1	22
Chapter 2	Numbers	23
	2.1 Introduction	23
	2.2 How to Read Integers	23
	2.3 How to Read Floating-Point Numbers	25
	2.4 Example - Average	26
	2.5 Example - Square a Number	27
	2.6 Example - Banking	28
	2.7 Example - Football Tickets	31
	2.8 Integers - Int	34
	2.8.1 Integer Expressions	36
	2.8.2 Precedence of Operators	37
	2.8.3 Print an Integer Using a <i>Field Width</i>	38
	2.9 Floating-point Numbers	40
	2.9.1 Print Float64 and Float32 Variables	41
	2.9.2 Assignment Between Float64 and Float32	42
	2 9 3 Floating-point Expressions	43

	2.9.4 Mixed Expressions	44
	2.10 Assignment Operator	45
	2.11 Updating Operators	45
	2.12 trunc, ceil, floor, round	46
	Exercises 2	51
Chapter 3	Selection Logic	53
	3.1 Introduction	53
	3.2 Boolean Expressions	53
	3.2.1 AND, &&	54
	3.2.2 OR,	55
	3.2.3 NOT, !	55
	3.3 The type Bool	56
	3.4 The if Statement	56
	3.4.1 Find the Sum of Two Lengths	59
	3.5 The if...else Statement	61
	3.5.1 Calculate Pay	63
	3.6 On Program Testing	64
	3.7 Symbolic Constants	65
	3.8 The if...elseif...else Statement	66
	3.8.1 Print a Letter Grade	67
	3.8.2 Classify a Triangle	68
	Exercises 3	69
Chapter 4	The for and while Statements	72
	4.1 Introduction	72
	4.2 The for Statement	72
	4.2.1 Multiplication Tables	75
	4.2.2 Temperature Conversion	78
	4.3 The Expressive Power of for	80
	4.4 break/continue in for	82
	4.5 Read Data From File	83
	4.5.1 Keep a Count	85
	4.5.2 Find Average	85
	4.6 Find Largest Number	86
	4.6.1 Find 'Largest' Word	88
	4.6.2 Find Longest Word	89
	4.6.3 Find Smallest Number	89
	4.7 Nested for Statement	90
	4.8 Read Data From File, Cont'd	92
	4.9 The while Statement	95
	4.9.1 Sum of Numbers (Prompt)	97
	4.9.2 Sum, Count, Average (Prompt)	98
	4.9.3 Greatest Common Divisor	99
	4.10 Send Output to a File	100
	4.11 Payroll	101
	4.12 break/continue in while	104
	Exercises 4	106
Chapter 5	Functions	109
	5.1 Introduction	109
	5.2 Function Basics	110

5.2.1 How an Argument Is Passed to a Function	113
5.3 Function - Examples	114
5.3.1 How to Swap Two Variables	114
5.3.2 Yesterday, Today and Tomorrow	115
5.3.3 GCD, Greatest Common Divisor	116
5.3.4 Using GCD to Find LCM	118
5.3.5 Factorial and Big Integers	118
5.3.6 Combinations	121
5.3.7 Calculate Pay	123
5.3.8 Sum of Exact Divisors	123
5.3.9 Perfect, Abundant or Deficient	124
5.3.10 Letter Position in Alphabet	125
5.4 Introduction to Recursion	126
5.4.1 GCD, Greatest Common Divisor	128
5.4.2 Fibonacci Numbers	128
5.4.3 Decimal to Binary	129
5.4.4 Towers of Hanoi	130
5.4.5 The Power Function	132
5.4.6 Find Path Through Maze	133
Exercises 5	137

Chapter 6

Characters & Strings	140
6.1 Character Sets	140
6.2 Character Constants and Values	141
6.3 The Type Char	142
6.4 Some Char Functions	143
6.4.1 Uppercase To/From Lowercase	144
6.5 Read and Print Characters	146
6.6 Count Space Characters	148
6.7 Compare Characters	149
6.8 Echo Input, Number Lines	150
6.9 Convert Digit Characters to Integer	151
6.10 String Basics	154
6.11 Compare Strings	156
6.12 Index Into a String	157
6.13 Example - Sum of Distances	159
6.14 Concatenation	161
6.15 Example - Get Words From Random Data	162
6.16 Example - Palindrome	164
6.17 A Flexible getString Function	166
6.18 Example - Geography Quiz Program	167
6.19 Other String Functions	169
6.19.1 findfirst	170
6.19.2 findlast	171
6.19.3 findnext	171
6.19.4 findprev	173
6.19.5 occursin	174
6.20 Array of Characters	174
6.21 For the Curious Reader	176
Exercises 6	180

Chapter 7	Arrays	181
	7.1 Introduction	181
	7.2 Simple vs Array Variable	182
	7.3 Array Declaration	182
	7.4 Store Values in an Array	186
	7.5 Average and Differences from Average	190
	7.6 Letter Frequency	191
	7.7 Array as Argument to a Function	194
	7.8 Name of Day Revisited	194
	7.9 Find Largest, Smallest in Array	195
	7.9.1 min, max, minimum, maximum	197
	7.10 A Voting Problem	198
	7.10.1 How to Handle Any Number of Candidates	202
	7.10.2 How to Sort the Results	202
	Exercises 7	205
Chapter 8	Searching, Sorting and Merging	207
	8.1 Sequential Search	207
	8.2 Selection Sort	209
	8.2.1 Analysis of Selection Sort	211
	8.3 Insertion Sort	212
	8.3.1 Analysis of Insertion Sort	216
	8.3.2 Sort Unlimited Data	216
	8.4 Sort Parallel Arrays	217
	8.5 Binary Search	219
	8.6 Word Frequency Count	221
	8.7 Merge Sorted Lists	224
	Exercises 8	228
Chapter 9	Structures	230
	9.1 The Need for Structures	230
	9.2 How to Write a struct Declaration	231
	9.2.1 Pass struct as Argument to a Function	232
	9.3 Array of Structures	234
	9.3.1 Sort struct Array	236
	9.4 Nested Structures	238
	9.5 Fractions	239
	9.5.1 Manipulate Fractions	240
	9.5.2 Rational Numbers	241
	9.6 Voting Problem Revisited	243
	9.6.1 On using isless in sort	247
	Exercises 9	250

Chapter 10	Dictionaries & Sets	252
	10.1 Dictionaries	252
	10.1.1 Letter-Frequency	254
	10.1.2 Dict Functions - haskey, in, delete!	256
	10.2 Sets	257
	10.2.1 Set Operations	258
	10.2.2 Find All Unique Words	263
	10.3 Thesaurus	264
	10.4 Scrabble	267
	Exercises 10	277
Appendix A	Install Julia/Atom/Juno	279
Index		286