Undergraduate Topics in Computer Science

‘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

Springer

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

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 an 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