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 include fully worked solutions.

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

Joe Pitt-Francis · Jonathan Whiteley

Guide to Scientific Computing in C++

Second Edition

Springer

Joe Pitt-Francis
University of Oxford
Oxford
UK

Jonathan Whiteley
University of Oxford
Oxford
UK

ISSN 1863-7310 ISSN 2197-1781 (electronic)
Undergraduate Topics in Computer Science
ISBN 978-3-319-73131-5 ISBN 978-3-319-73132-2 (eBook)
https://doi.org/10.1007/978-3-319-73132-2

Library of Congress Control Number: 2017962059

Printed on acid-free paper

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

Preface to the Second Edition

The principle changes in this updated edition are additional material on software testing and on some of the new features introduced in the C++11 standard. When introducing this additional material, we have followed the same philosophy as when writing the first edition of this book. That is, we focus on a concise discussion of the key features that are most useful to the novice and intermediate programmer in the field of scientific computing. We have found this an effective approach when teaching this course to graduate students—once the basics have been mastered, students then have the confidence to find out about less well-used features themselves when they are needed.

This second edition would not be as complete—or as enjoyable to update—without discussions with colleagues and other readers of the first edition, including those previously unknown to us who were kind enough to provide constructive feedback. We would like to express our gratitude to all who contributed in this way or offered their encouragement, and to the staff at Springer for inviting us to update the first edition.

Finally, we would both again like to thank our families for their love and support.

Oxford, UK
October 2017

Joe Pitt-Francis
Jonathan Whiteley

Preface to the First Edition

Many books have been written on the C++ programming language, varying across a spectrum from the very practical to the very theoretical. This book certainly lies at the practical end of this spectrum and has a particular focus for the practical treatment of this language: scientific computing.

Traditionally, Fortran and MATLAB®[1] have been the languages of choice for scientific computing applications. The recent development of complex mathematical models—in fields as diverse as biology, finance and materials science, to name but a few—has driven a need for software packages that allow computational simulations based on these models. The complexity of the underlying models, together with the need to exchange code between co-workers, has motivated programmers to develop object-oriented code (often written in C++) for these simulation packages. The computational demands of these simulations may require software to be written for parallel computing facilities, typically using the Message Passing Interface (MPI). The need to train programmers in the skills to program applications such as these led to the development of a graduate-level course *C++ for Scientific Computing*, taught by the authors of this book, at the University of Oxford.

This book provides a guide to C++ programming in scientific computing. In contrast to many other books on C++, features of the language are demonstrated mainly using examples drawn from scientific computing. Object orientation is first mentioned in Chap. 1 where we briefly describe what this phrase—and other related terms such as inheritance—means, before postponing any further discussion of object orientation or related topics until Chap. 6. In the intervening chapters until object orientation reappears, we present what is best described as "procedural programming in C++", covering variables, flow of control, input and output, pointers (including dynamic allocation of memory), functions and reference variables. Armed with this grounding in C++, we then introduce classes in Chaps. 6 and 7. In these two chapters, where the main features of object orientation are showcased, we

[1]MATLAB is a registered trademark of The MathWorks, Inc.

initially, for the sake of clarity, abandon our principle of using examples drawn from scientific computing. Once the topics have been presented however, we resume our strategy of demonstrating concepts through scientific computing examples. More advanced C++ features such as templates and exceptions are introduced in Chaps. 8 and 9. Having introduced the features of C++ required for scientific computing, the remainder of the book focuses on the application of these features. In Chap. 10, we begin to develop a collection of classes for linear algebra calculations: these classes are then developed further in the exercises at the end of this chapter. Chapter 11 presents an introduction to parallel computing using MPI. Finally, in Chap. 12, we discuss how an object-oriented library for solving second-order differential equations may be constructed. The importance of a clear programming style to minimise the introduction of errors into code is stressed throughout the book.

This book is aimed at programmers of all levels of expertise who wish to write scientific computing programs in C++. Experience with a computer to the level where files can be stored and edited is expected. A basic knowledge of mathematics, such as operations between vectors and matrices, and the Newton–Raphson method for finding the roots of nonlinear equations would be an advantage.

The material presented here has been enhanced significantly by discussions about C++ with colleagues, too numerous to list here, in the Department of Computer Science at the University of Oxford. A special mention must, however, be made of the Chaste[2] programming team: particular gratitude should be expressed to Jonathan Cooper for readily sharing with us his impressively wide and deep knowledge of the C++ language. Other members of the team who have significantly helped clarify our thoughts on the C++ language are Miguel Bernabeu, James Osborne, Pras Pathmanathan and James Southern. We should also thank students from both the M.Sc. in Mathematical Modelling and Scientific Computing and the Doctoral Training Centres at the University of Oxford for unwittingly aiding our understanding of the language through asking pertinent questions.

Finally, it is always important to remember—especially when debugging a particularly tiresome code—that there is far more to life than C++ programming for scientific computing. We would both like to thank our families for their love and support, especially during the writing of this book.

Oxford
October 2011

Joe Pitt-Francis
Jonathan Whiteley

[2]The Cancer, Heart And Soft Tissue Environment (Chaste) is an object-oriented package, written in C++, for simulations in the field of biology. More details on this package may be found at https://www.cs.ox.ac.uk/chaste/.

Contents