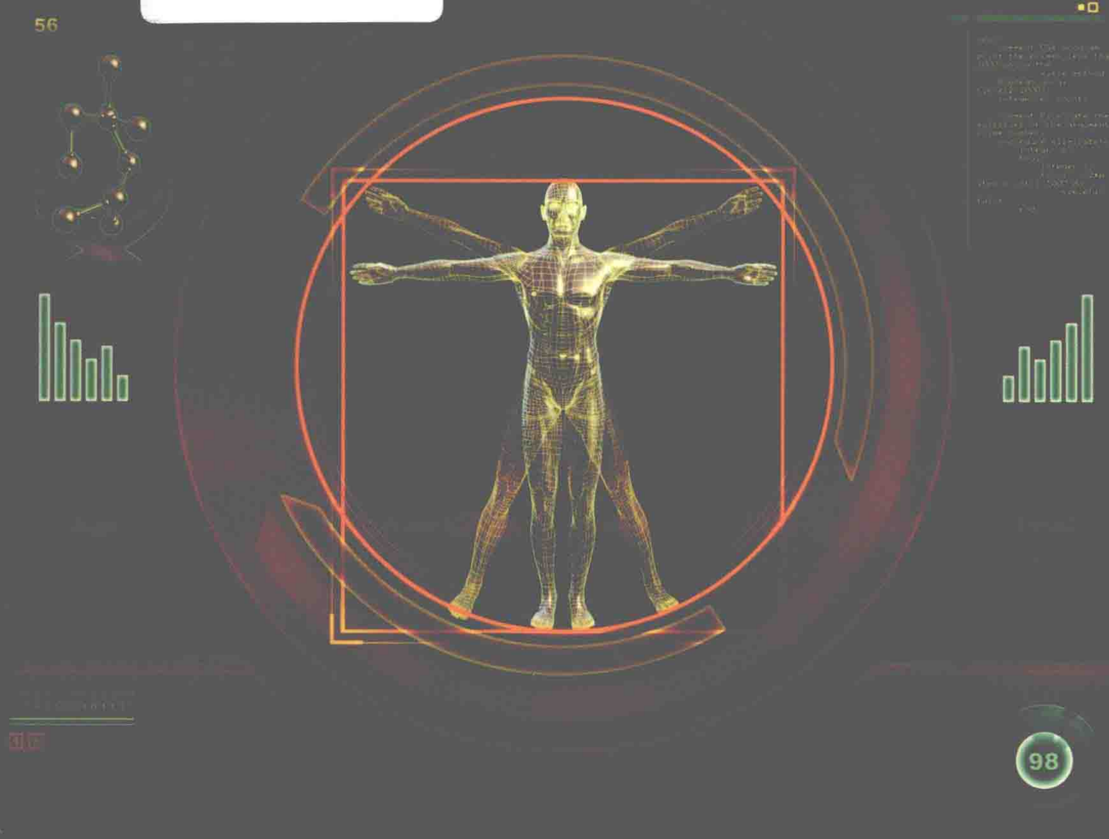
SCIENTIFIC PROGRAMMING

C-Language, Algorithms and Models in Science

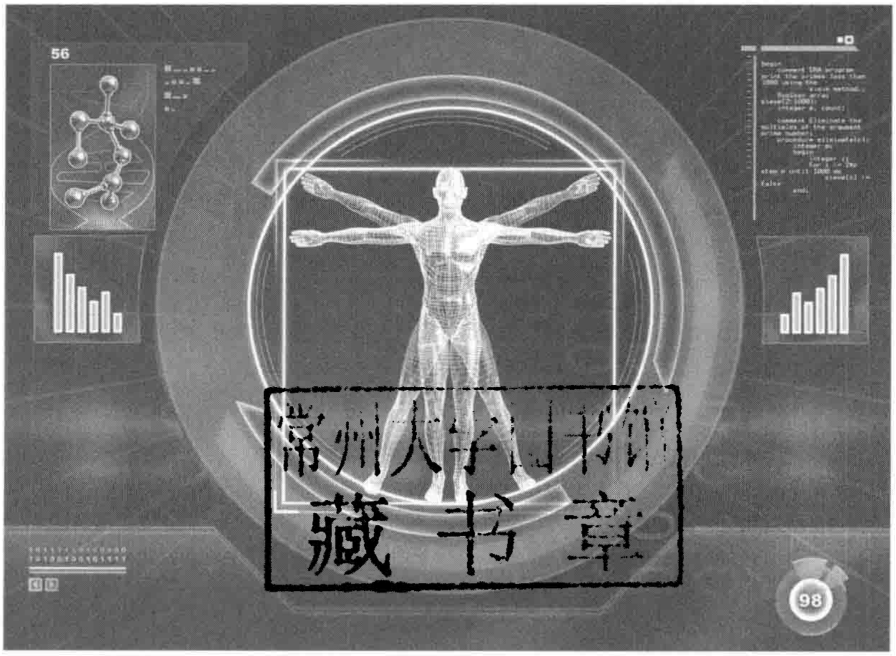


Luciano Maria Barone • Enzo Marinari
Giovanni Organtini • Federico Ricci-Tersenghi

World Scientific

SCIENTIFIC PROGRAMMING

C-Language, Algorithms and Models in Science



Luciano Maria Barone • Enzo Marinari
Giovanni Organtini • Federico Ricci-Tersenghi
"Sapienza" Università di Roma, Italy

World Scientific
NEW JERSEY • LONDON • SINGAPORE • BEIJING • SHANGHAI • HONG KONG • TAIPEI • CHENNAI

Published by

World Scientific Publishing Co. Pte. Ltd.

5 Toh Tuck Link, Singapore 596224

USA office: 27 Warren Street, Suite 401-402, Hackensack, NJ 07601

UK office: 57 Shelton Street, Covent Garden, London WC2H 9HE

Library of Congress Cataloging-in-Publication Data
Barone, Luciano M. (Luciano Maria), author.
Scientific programming : C-language, algorithms and models in science / by Luciano M. Barone (Sapienza Università di Roma, Italy), Enzo Marinari (Sapienza Università di Roma, Italy), Giovanni Organtini (Sapienza Università di Roma, Italy) & Federico Ricci-Tersenghi (Sapienza Università di Roma, Italy).
pages cm
Includes bibliographical references.
ISBN 978-9814513401 (hardcover : alk. paper)
1. Science--Data processing. 2. Science--Mathematical models. 3. C (Computer program language) 4. Computer programming. I. Marinari, Enzo, author. II. Organtini, Giovanni, author. III. Ricci-Tersenghi, F. (Federico), author. IV. Title.
Q183.9.B37 2014
502.85'513--dc23

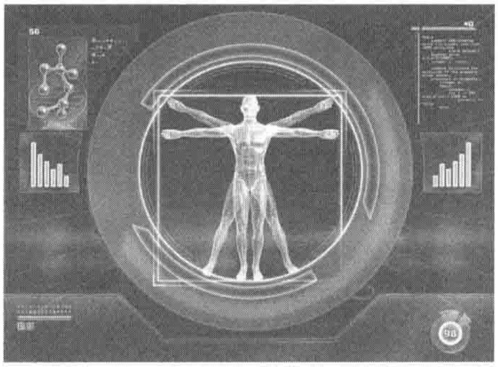
2013012751

British Library Cataloguing-in-Publication Data
A catalogue record for this book is available from the British Library.

Printed in Singapore by World Scientific Printers.

SCIENTIFIC PROGRAMMING

C-Language, Algorithms and Models in Science



For Rossella
so patient with me and much more

For Margherita,
a flower indeed,
and for Chiara,
fair by name and by nature

For Federica, Lorenzo and Giulia
"the summum bonum was in the gifts of nature,
in those of fortune, in having many friends,
and many and good children"

Miguel de Cervantes Saavedra
"Don Quixote" (1615)

For my children Gaia and Marco
that make my life full of joy and fun,
bright colors and laughing sounds
...every day more and more!

Preface

A book teaching how to program should help its readers to learn with minimal effort how to write properly working, efficient and interesting programs. Its potential readers may have very different interests, and the books that are currently available on the shelves generally do not take these needs into account. They mostly address a generic reader, whose interests the author is unaware of.

Books presenting a programming language in a specific context are much more effective and nicer to read, since the reader learns at the same time both the language and the basic concepts needed to formulate the problems one wants to solve. These books obviously address a more specialized audience. However, in many cases they are very precious and they are the right choice to teach professional and university courses.

The collection of programming manuals addressing a specialized audience is rather scarce and those dedicated to scientific programming even less numerous. Since computers play an increasingly important role in modern science this situation is very unfortunate. In many fields it is essential to both being able to express ideas into words or formulas and to translate them into a precise and rigorous programming language.

This book very nicely fills this gap in the current literature. The different parts of the C language are presented together with scientifically interesting algorithms, each one of which is then put into practice. Often the algorithms are not at all basic, and their theoretical motivation is explained in detail. Various implementations of a single algorithm are discussed in depth, keeping the different, sometimes opposing needs into account: precision, execution efficiency, code compatibility and readability. In this way the reader learns how to tackle a new problem while keeping these various aspects in mind, and how to select an optimal choice fitting his needs.

The topics have been carefully selected and the problems are very accurately discussed, as are the codes included in the book. Indeed, this book is the result of years of experience in teaching academic courses in scientific programming. Moreover, the authors have many years of experience addressing scientific problems with the use of the computer, introducing new techniques and designing algorithms which have become commonly used. It is hard to think of authors more qualified than the present ones to write this kind of book.

The original setup and the authors' extraordinary talent make this book an excellent product, a reference point for all those interested in scientific programming.

Rome, February 10, 2006

Giorgio Parisi

Giorgio Parisi is a full professor in Theoretical Physics at the *Sapienza* University of Rome. He is a member of the the French and of the American Academy of Sciences and of the Lincei Academy. For his work in theoretical physics he received the International Feltrinelli prize in 1986, the Boltzmann medal in 1992, the Dirac medal and prize in 1999, the Fermi medal in 2002, the Nonino prize and the Dannie Heineman prize in 2005. the Galileo Prize in 2006, the Microsoft European Science prize in 2007, the Lagrange in 2009 and the Max Planck medal in 2011. He wrote three books and over 500 scientific articles.

Foreword

The decision to write a textbook is always a difficult and demanding choice for scientists engaged full-time in research activities. It is a very time consuming task, and it requires to have a clear idea about the direction to take in order to produce an innovative book.

Fortunately, some years ago, we started teaching, together with some other colleagues, a course about scientific computing to physics students of the Mathematical, Physical and Natural Science Faculty of the *Sapienza* University in Rome. These courses were a novelty. They were introduced in the context of the recent Italian and European reform, the so-called *three plus two*, aiming to provide students with skills which allow them to operate professionally in a scientific and technological environment.

The subject of these courses is based on our computer programming experience in various fields of scientific research. Throughout our teaching we realized that many fundamental notions about computing for scientific applications are never discussed in the standard manuals and textbooks. We were not able to find an adequate textbook for these kind of courses. Indeed, commonly used textbooks belong to one of the two following categories:

(1) programming language manuals;
(2) numerical analysis textbooks.

The former often contain examples on how to use a programming language without considering the application context. They contain plenty of examples and exercises, which are more game-like or have a concern a managerial context; the few examples written in a precise mathematical form are too simple for a university student and are therefore not appropriate for a scientific (engineering, mathematics, physics, chemistry, biology) faculty. On the other hand, numerical analysis textbooks are appropriate for advanced students and usually require a good knowledge of programming and of mathematical and physical notions. Using two textbooks, one of each category, does not solve the problem either, because the important topics are not correlated as they do not follow the same track.

Finally, we became convinced of the fact that the current survey of textbooks does not include an introductory programming manual for scientific university courses.

Forgetting all about the existing textbooks, we started working on an innovative scheme which we considered adequate for the new university courses. Moreover, the possibility to write a new textbook seemed interesting. We also felt that the project we were working on had an international feel and this stimulated us.

We conceived and built a scheme not just to teach students a programming language, but rather to give them the ability to build models to solve scientific problems by programming. The teaching method we chose takes into account the student skills evolution in time. The examples given are always accessible to the student. The first examples simply refer to mathematical problems in infinitesimal calculus. In the second part fundamental problems of both physics and mathematics are tackled, such as the study of differential equations and their application to problems in dynamics. Finally, in the third part, we discuss rather complex optimization problems.

The textbook is structured in three parts, reflecting the three teaching phases and corresponding more or less to three university courses. The complete journey goes hand in hand with a complete engineering or scientific graduate course. The three courses we propose are strongly linked, and guide the student from the condition of illiterate in computer science to an advanced and competent level. At the same time, the three parts are rather independent and allow a teacher to select only certain parts for particular courses.

The title of this textbook is short, but expresses various strong ideas. The *Programming* expresses the fact that we teach techniques on how to build models and generic algorithms to solve a problem, while creating a good executable code. The *Scientific* has a double meaning: it partly refers to the fact that we treat scientific problems, and partly refers to our way of introducing programming methods. Methods, statements and models are not given as *recipes*, but for each one of these we carefully examine their "raison d'etre", the mechanisms which make them useful, the possible alternatives and their pro and cons.

What this book is not

As we already mentioned, this book is not a C manual: there are already many manuals available in print and on the Internet. In *Scientific Programming* we introduce the C language as a tool to learn how to program: a process which is fully analogue to the one in which you learn how to talk. Just like you need a language when you learn how to talk, you need one to learn how to program. However, it is beyond the scope of this textbook to analyze the C language in-depth in all its aspects. For this kind of analysis, we refer the reader to the bibliography. Still, this textbook contains enough information to introduce the reader to the language without having to refer to a manual.

This textbook is not a generic programming manual either: no managerial, graphics or multimedia applications are discussed in detail. The scientific applications mainly consist in calculating and organizing data. The results are often represented in a graphic way, but these were mostly obtained with specialized programs. These programs are easily available on the Internet, and we do not discuss their setup.

Nor is this book a numerical analysis manual. It includes many, sometimes advanced, computation algorithms, but they are always presented in a simple and intuitive way. We do not want to discuss all aspects of numerical analysis from a mathematical point of view. We refer the reader to the bibliography for the proof of theorems, the detailed discussion of what certain choices imply and the theoretical aspects. Nevertheless, a diligent student will acquire all the tools needed to write complete, accurate and efficient programs.

How to use this textbook

This textbook consists of three parts. Each part reflects the contents of the courses we teach in the corresponding year of an undergraduate program. Therefore, each part contains the material of a first level, 6 credits university course.

The first part does not require any specific prior knowledge about programming. The student is guided along a track including the knowledge of the basic tools (the computer and the programming language), which is essential to understand the techniques to design the solutions. In the first part we aim to familiarize the student with the techniques by learning how to use them in simple and interesting examples. The student learns the elementary and fundamental C constructs and how to apply these in the basic numerical techniques (derivatives, integrals, sorting).

In the second part, we focus on the design aspect. In each chapter we consider the solution to a different problem. This is the right time to show, with examples, the general techniques to solve problems, to thoroughly analyze the advanced aspects of the language and to consider elements often neglected, such as the program's robustness and efficiency. The examined problems are original and stimulating.

In the third part we assume the student fully masters the tools (in particular the programming language). We discuss complex computation problems, which are common in engineering or science, along with code optimization techniques. Also in this part the topics we treat offer a generic overview of the techniques in use, but we frequently examine new and lesser known ones. In our opinion, the third part is more dense of what can be reasonably treated in a standard course and the teacher is free to choose which topics to discuss.

We hope that the student using this textbook becomes at least half as enthusiastic as we were while writing it. In which case our book will turn out to be, beyond any doubt, very useful.

Acknowledgments

At the end of this project, we first of all want to thank our colleagues, our collaborators, our PhD students and all those who depend on us in one way or the other, since in order to write the book we have taken away crucial time from them.

A special thanks goes to the Director and the personnel of the Physics Department of our University, and to the Directors and the personnel of the National Institute of Nuclear Physics and the National Institute of Matter Physics (now unfortunately no longer existing) Sections of Rome 1: without their help our job would have been impossible.

Thanks to our colleagues Zhen'an Liu of Beijing and Wu Hao of Shanghai, for helping us write the citation in Chapter 3 and to Anna Malerba for the citation in Chapter 2.

We also got some very important help from all our colleagues teaching the programming courses in the Physics Graduate Program of the *Sapienza*

University. The students attending our courses stimulated us and pressed us with questions and discussions which were often very important for the development of this textbook. Though they are far too many to quote them all here, we are very grateful to all of them.

Some PhD students and young researchers helped us during our courses throughout the years, and they corrected many, small and big, errors: we sincerely thank Tommaso Castellani, Tommaso Chiarusi, Jovanka Lukic and Valery Van Kerrebroeck.

Carlo Piano has been helping us with organizing the Italian version of the book, and his help has been for us very important.

Lisa Ferranti has been somehow the real responsible of all this. As the editor of the Italian version of the book she has believed in the book from the first moment, she has helped us, she has pushed us, she has brought us to print the book. We could not thank her more warmly.

Introduction to the English version

It took a little time, after we realized that we considered appropriate having this book available in English, finding the strength to go ahead with the project. We were indeed convinced it was a very good idea, since there is here an original approach that we hope will be useful to many students approaching the world of scientific computing. Also we had signals from many non-Italian colleagues, using the book (thanks to their brilliant knowledge of our language) as a trace for their courses but being unable to really give it, because of the language, as a source to the students.

We are very indebted to Valery Van Kerrebroeck that translated the text from Italian, with passion and dedication: without her our effort would probably have been vain. Many young colleagues close to our research groups have helped us in a final revision of the translation; even if they are too many to be quoted individually, they should know we are really grateful to all of them.

Rome, June 7th, 2013.

Luciano M. Barone, Enzo Marinari, Giovanni Organtini, Federico Ricci-Tersenghi.

Technical note

This textbook was written using LaTeX[Mittelbach and Goosens (2004)] version 3.141592-2.1 on a Linux operating system[1].

We used the `emacs` and `nedit` editors to write the text and the programs. We compiled the latter with `gcc` version 3.3.3. Generally we used the `-pedantic` option which generates a message in case the code is not compatible with the ANSI standard. We adopted this standard throughout the complete textbook, except for a few cases which we mentioned explicitly.

The figures were generated with the LaTeX `pstricks` package or the programs `gnuplot` and `xfig`, in Linux. We used different packages following the common habit in scientific environment to choose the best tools for a given problem; this is quite different from other environments, where usually the choices are dictated by conventions, convenience or imposition. This also allows us to show how versatile *open source*[2] programs are.

Typographical Conventions

We have quoted the programming language constructs and the system commands with a different font than the one used for the text. Its horizontal dimension is constant (*monospaced*) and does not vary with the character, as in

[1]Linux is a trademark registered by Linus Törvalds.

[2]Open source software is computer software whose source code is freely available and provided under an open source license allowing it to be studied, copied, changed and redistributed.

```
while
ls -la
return 0;
for (i = 0; i < 10; i++) {
```

We also used a similar font to represent URL names (*Uniform Resource Locator*) of Websites:

```
http://www.pearsoned.com
http://www.scientificprogramming.org
```

Characters that possibly cause confusion in the middle of the text (especially punctuation marks) are represented with a gray background: `.`, `;`, `{`.

To show the syntax and the general definitions of commands, statements and functions we used a different monospaced character, as in

```
#define symbol [value]
```

In this scope words in italics, such as *symbol*, represent generic elements which have to be replaced by the actual ones. When they are included between square brackets, as in [*value*], they represent optional elements which may or may not be present. The example above describes several possible alternatives, as in

```
#define _DEBUG
#define MAX 100
```

In the first example *symbol* is replaced by `_DEBUG` and the optional value [*value*] is not present. In the second, *symbol* is replaced by the string `MAX` and an optional value `100` is present, described as *value* in the syntax example.

Program variable names are always chosen on the basis of what they represent.

Generally, programmers use a US keyboard. The reason is simply because programming languages often use the symbols present in this type of keyboard, which is considered to be international, and not in others. The C language is no exception to this rule and uses curly brackets and the tilde (~). In case one does not have a US keyboard, the required symbols can be created using certain combinations of the keys in order to avoid having to redefine the keyboard.

In the Microsoft Windows operating systems, the opening curly brackets can be obtained by holding the `Alt` key and pressing subsequently the keys 1, 2 and 3 of the numerical keypad. The closing curly bracket is obtained