Scientific Programming 22 (2014) 53–55
DOI 10.3233/SPR-130364
IOS Press

Book Review

Salvatore Filippone
E-mail: salvatore.filippone@uniroma2.it

Abstract. The book by Arjen Markus is a veritable tour de force among the possibilities opened up by the latest incarnation of Fortran, the longest-lived programming language on the planet and still one of the favourites by scientific programmers. It is not an introduction to the syntax and semantics of the latest language standard: it is more of a gourmet cookbook showing off a wide range of examples of what the new features allow a daring programmer to do.

Arjen Markus, *Modern Fortran in Practice*, Cambridge University Press, 2012.

1. Introduction

Your reviewer likes this book a lot. It is one of the lucky events when I find myself resonating positively with that person on the other side of the printed pages, the author.

To begin with, I think the title is a very good one. Like the latest incarnation of the reference book by Metcalf, Reid and Cohen [2], it prominently displays on the cover a very important statement: that "Modern" and "Fortran" can and actually do coexist in a single sentence without it being an oxymoron. This is especially important given a widespread negative attitude of dismissing Fortran programming, and the related techniques, by people who more often than not end up reinventing the same ideas under different names, and hitting the same pitfalls.

The book is packed full of relevant and sometimes intriguing examples, ranging from "traditional" root finding and numerical integration schemes to GUI and network interfacing, with databases thrown in between. The reader will embark on a tour de force touching on multiple facets of the language, covering areas that are at first sight quite distant from each other; this is all the more important in the current age when programmers increasingly need to interface different components and tools, each of them carrying its own bag of tricks.

The important difference with other excellent books is the point of view: a systematic introduction to the Fortran language (such as [2]) will walk you through the language features giving examples. This book is solving what in mathematical terms might be called an "inverse problem": you (the programmer) are facing this the situation, these language features may help you in figuring out a solution.

If this is enough to whet your appetite, keep reading!

2. Contents

The book starts off with a brief review in Chapter 1 of the history of the Fortran language from the Fortran 77 standard to the latest Fortran 2008 incarnation, with a summary of what changed in each revision. This is a very necessary step, given the long history of the language and the sometimes surprising number of new features available today; few if any people know them all.

Chapter 2 discusses topics related to the array-programming feature set, specifically the use of functions with array-valued results and the properties of elemental functions and subroutines.

In Chapter 3, we are introduced to how a judicious use of procedure pointers can lead to an extremely concise and high-level style of programming, very close to a blackboard description of the underlying mathematics.

Chapter 4 gets back to the more mundane topic of memory management, and how the various types of dynamic arrays available in the modern Fortran standards can help. The chapter is full of sensible advice, and more importantly the advice is backed by performance data. After all, a Fortran programmer is likely to be interested in extracting the best possible performance from his/her computing platform.

Chapter 5 deals with a problem that has haunted developers of general purpose libraries for a long time: how to provide a procedure that is of general use even though it sits between a caller and a callee that are both

defined by the user and contain dependencies on very specific data configurations. The reader is presented with a dozen possible alternatives, from old ones already known from the Fortran 77 days to the ones made possible by the latest standards, and after reading through will have gained the perspective needed to plan a course of action appropriate to the situation at hand.

Chapters 6 and 7 deal with a very important issue for present-day professionals: how to interact with external tools developed in other languages and with very different constraints. Specifically, Chapter 6 presents the C-interoperability features that enable a standard solution to the vexing problem of figuring out the needed conventions for interlanguage calls. Chapter 7 provides advice on how to output data in graphical format and/or how to interact with structured data on the Internet, taking as a reference example the XML format.

Chapters 8 and 9 take us on a detour into software development techniques, namely unit testing and code reviews. While these two chapters are by no means an exhaustive introduction to software engineering, let us not forget that many Fortran programmers are application developers in their respective fields and may not have had specific training into modern techniques for managing the software development process; therefore, they will find this introduction quite useful.

Chapter 10 goes back to a "traditional" topic, one that probably is the bread and butter of any scientific programmer: devising robust versions of a number of simple numerical algorithms including interpolation and root finding.

Chapter 11 takes us into the territories of object-oriented programming with a crash-course introduction to such topics as type-bound procedures, abstract data types, generic programming and design patterns. Chapter 12 finishes with a look at parallel programming alternatives, including the coarray model that is part of the Fortran 2008 standard; the various possible approaches are compared with reference to a specific example.

3. Comments

I believe any serious Fortran developer should take a close look at this book. I personally found the section on object-oriented programming to be very well-written and to provide a very good overview of most important points about how to use OOP in practice. This overview includes the introduction of the concept of Design Patterns, which is familiar to computer scientists but much less so to scientific programmers; the interested reader may delve further into this particular subject with the book by Rouson, Xia and Xu [3].

Markus goes far enough to propose his idea about how to emulate template programming: while I may have slightly different opinions on how to tackle this problem, his examples are nonetheless intriguing and well thought-out.

The chapter on parallel programming provides a good demonstration of how the Fortran environment supports many different solutions, including OpenMP, MPI and coarrays, to the issue of programming parallel machines. We can see here an application of an important fact: Fortran is the only language that has an international standardization body that views high-performance, scientific programmers as its primary target audience. This cannot, in my opinion, be overemphasized: modern computing machines are very complicated devices; any hardware/software designer must make multiple trade-offs; and these may well have motivations that are not immediately conducive to high-performance solutions to the problems of interest to the scientific programming community. The Fortran standards committee has done a wonderful job of navigating over the years through sets of conflicting requests to find a good design point allowing both expressiveness of the resulting language as well as leaving compiler implementors the leeway necessary to do a good job.

Does the compiler solve all problems? Certainly not, some knowledge of where performance bottlenecks arise is necessary for all programmers. Consequently the chapters on memory management, on the robust implementation of algorithms, and the Appendix B "Caveats", will be extremely useful to scientists developing their application and to programmers coming from other languages to learn the Fortran mindset along with the Fortran syntax.

At this point I need to warn the reader that a working knowledge of the language is needed to wade through the chapters to bring home the valuable lessons they contain.

Therefore, if you, the reader, are not already familiar with Fortran 2003/2008, you are cordially invited to complement this book with one of the many introductory and/or reference books on the language, of which the aforementioned text [2] is but one example.

Furthermore, I highly recommend any reader of this book to download the code samples in the compan-

ion web site and to complement reading each chapter with perusing (and testing) the actual codes themselves, delving into the many details that make up the complete solutions.

Most chapters can be read independently of each other; the book can be used as a quick reference on many topics.

If I have given so far the impression that this is a perfect book, this is (alas!) not true. Among the things that I would have liked to see spelled out differently are:

- The URL for the software is not correct; it is easy to find the right one by searching for the author and book title, but this ought to be fixed in future printings. (Yes, I am wishing to see this book doing well enough to warrant further printings.)
- In Section 3.4 we are introduced to the wonders of integer operations; however the magic behind the type(integer_relation) is not revealed, and the reader has to look carefully through the software in the companion web site to figure out what is going on (as a side issue, the only relation implemented is equality and is coded with the integer value "1" whereas I would have found it in better taste to define a parameter with a symbolic name, to make it clear that the approach is easily extensible).
- In Chapter 4, about memory management, the author very rightly underlines the differences in performance between allocatable, automatic and pointer arrays; in doing so, however, he takes for granted that the reader knows the difference between allocating on the stack or on the heap. This may or may not be the case. A self-contained explanation would probably have been feasible with a few lines. Moreover, at the end of Section 4.4 the author mentions a strategy to reallocate a string to ever-increasing length in an iteration loop; I found surprising that there is no mention of the quadratic cost that such an approach entails.
- A couple of code fragments, e.g., the type point_2d of Section 11.1, contain syntax errors in the printed version; the corresponding code in the companion web site is correct.
- In many other cases (e.g., Section 4.4) the code fragments make use of auxiliary/support routines without defining them explicitly, thus leaving the reader to wonder whether they are part of the language or not. A careful look at the software from the companion web site will solve all such doubts, and I heartily recommend this option to all readers.

It is I think fair to say that if the author had spelled out in full details all the items contained in the book, the book would have become significantly larger; nevertheless, a reader coming from a different language background may be slightly disoriented. If you are not familiar with the Fortran 90 syntax, either because you are a Fortran 77 programmer or because you have a totally different background, get yourself acquainted with it first, and then you can appreciate this book.

4. Summary

In a nutshell: if you want to brush up your Fortran, or if you want to see for yourself whether Fortran is really a language suited only for dinosaurs, then this book will be very useful to you. By all means, check it out.

References

[1] J.C. Adams, W.S. Brainerd, R.A. Hendrickson, R.E. Maine, J.T. Martin and B.T. Smith, *The Fortran 2003 Handbook*, Springer, 2008.

[2] M. Metcalf, J. Reid and M. Cohen, *Modern Fortran Explained*, Oxford Univ. Press, 2011.

[3] D.W.I. Rouson, J. Xia and X. Xu, *Scientific Software Design: The Object-Oriented Way*, Cambridge Univ. Press, 2011.