JOURNAL OF AEROSPACE COMPUTING, INFORMATION, AND COMMUNICATION
Vol. 2, May 2005

Book Review

C++ and Object-Oriented Numeric Computing for Scientists and Engineers

Daoqi Yang, Springer-Verlag, New York, July 2000, ISBN: 0-387-98990-0, Hardcover, 464 pp., $47.75

"C++ and Object-Oriented Numeric Computing for Scientists and Engineers" is one of the few C++ books targeted toward the scientific community that uses numeric computing as the primary research tool. The book is divided into three parts: the first part introduces the reader to general programming concepts, the second part focuses on the unique aspects of C++ as an object oriented programming language, and the third part concludes the book with a description of the C++ standard libraries on containers and algorithms. The book consists of eleven chapters written in 440 pages, and the readers can download sample programs used in the text from a dedicated web-page. The book provides a concise, and at some places succinct, description of the ISO/ANSI C++ programming language. This is a good book for readers who have working knowledge of C++ and would like to learn to use it for scientific computing.

Unlike other programming books, this book does not commence with a "hello world" program, but introduces another, somewhat involved sample program to explain basic input/output (I/O), variable types, iterators, and basic C++ program structure. The first chapter provides a very useful, and often disregarded, discussion on numerical limits of the basic numeric data types (int, float, etc.) provided by C++. The importance of recognizing the "finiteness" of the accuracy of computer-generated data cannot be emphasized enough.

Throughout the book the author presents several programming tips for numeric computing such as offsetting pointers to easily access band matrices. Section 4.4 deserves a special mention as it briefly introduces some essential tools such as Makefile, timing programs, and linking C/C++ programs with Fortran codes. The description is very concise but is enough to allow the reader to start using these features with his codes.

The second part of the book begins with a succinct description of classes. The description is too brief for a beginner to absorb all the material in one reading. Perhaps all the important things are covered in this chapter (Chapter 5), but they are not repeated or visually emphasized to make a lasting impression on the reader new to the concept of classes. The three essential features of object oriented programming (OOP): data encapsulation, inheritance, and polymorphism are only briefly mentioned in this chapter, but described more fully in later chapters.

The author has tried to conclude every chapter with a section on practical numeric computing examples such as interpolation, root finding methods etc., which I found to be most appealing. It is often not the concepts of OOP that users find difficult to comprehend but the implementation of OOP style for computing problems. The examples and exercises in this book are very helpful in this regard. The exercise problems are mostly related to the application of OOP techniques to numeric computing. They are very "doable" and help in understanding the concepts. The book also provides helpful hints to relatively difficult exercise problems.

The discussion on efficient techniques for numerical integration using expression templates and template meta-programs in chapter 7 is a very revealing description of how the advanced features of OOP can be used without losing the efficiency of a sequential program. The book also mentions a few C++ scientific computing libraries such as PETE and POOMA, which were being developed at Los Alamos National Laboratory at the time of publication of the book. A free version of POOMA, FreePOOMA is hosted at `http://savannah.nongnu.org`. Chapter 8 describes the concepts of inheritance and polymorphism, virtual functions, and abstract classes. It also discusses efficiency issues with run-time polymorphism and suggests the use of static (compile-time) polymorphism for higher efficiency.

Received 14 February 2005; revision received 24 March 2005; accepted for publication 25 March 2005.

The third and final part of the book deals with the standard containers and algorithms in the Standard Template Library (STL) of C++. The explanation provided here is not at par with the quality of the rest of the book. It does not provide examples of scientific usage for features such as linked lists, sets, etc. Examples such as Huffman coding or binary search trees could greatly add to the content. Interested readers should consider Stroustrup[1] for in-depth understanding of these features of C++. In the final chapter, the different ideas presented throughout the book are combined and C++ codes for many linear system solvers are provided.

Overall, the book is very informative and useful to people interested in using OOP for scientific computing. It does little to motivate a sequential programmer to choose an OOP style instead. This can be addressed by adding a chapter or a section listing the benefits of using the OOP paradigm (supported by languages such as C++) over the sequential programming style (Fortran, C, etc.). The book is fairly easy to read and understand provided the reader already has a basic understanding of the C++ language. Readers with sequential programming experience may also find the last two parts of the book somewhat difficult to comprehend because of brevity.

Readers aiming at learning C++ should instead consider Shtern[2], Schildt[3] or Davis[4]. Readers looking for an introductory C++ book with some engineering flavor may consider Etter and Ingber[5] or Bronson[6] as optional texts.

References

[1]Stroustrup, B., *The C++ Programming Language*, Special Edition, Pearson Education, 2000.
[2]Shtern, V., *Core C++: A Software Engineering Approach*, Prentice Hall, 2000.
[3]Schildt, H., *C++ The Complete Reference*, Third Edition, Osborne/McGraw-Hill, 1998.
[4]Davis, S. R., *C++ for Dummies*, Fifth Edition, John Wiley & Sons Inc., 2004.
[5]Etter D. and Ingber, J., *Engineering Problem Solving with C++*, Prentice Hall, 2003.
[6]Bronson, G., *C++ for Engineers and Scientists*, Second Edition, PWS Pub Co, 1999.

Anupam Sharma*
Fluid Mechanics Laboratory
General Electric Global Research Center
One Research Circle, ES-500
Niskayuna, NY-12309, USA

* Aerospace Engineer, AIAA Student Member, sharma@research.ge.com