



**A First Course in Computational Physics and Object-Oriented Programming with C++**, David Yevick, Cambridge University Press, 2004, pp: 403, ISBN 0521827787 (hc); Price: US\$70.00.

This is mainly a book on programming in C++ with references to mathematics commonly used in physics. I consider it fairly suitable for someone who is familiar with the C programming language (the ancestor of C++), or C++ itself, and the mathematics used. That person should also be willing and able to correct mistakes.

Included with the book is a compact disc that contains C++ compilers; the *Crimson Editor*, a text editor designed especially for programming; separate *dislin* graphic package for compilers; and electronic versions of source codes for Chapters 12 through 23 and Appendix D. The compilers and editor are designed for *Windows* operating systems, but the book indicates that *Dev-C++* is also available for *Linux*. I installed the *Dev-C++* compiler, the *Borland C++* compiler, and the *Crimson Editor* on a computer running *Windows 98 second edition*. I tried all the supplied electronic source codes and typed in many others.

Installation of the software was fairly complicated. Furthermore, while *Dev-C++* runs as a graphically oriented *Windows* program, *Borland C++* is a "command-line" program, running from the *MS-DOS* prompt. I am comfortable with the latter type of program, but many users of *Windows* are not familiar with *MS-DOS* and would have trouble using the supplied *Borland* compiler if they could use it at all. *Dev-C++* had its problems: Quitting it required my typing <Ctrl> <Alt> <Delete> and choosing **End task** several times. That procedure is usually reserved for malfunctioning programs. I tried many times to produce a watch variable in *Dev-C++*, but succeeded only once.

The book is divided into four sections: C++ programming basics, numerical analysis, advanced object-oriented programming, and scientific programming examples. The last section is much shorter than the others. Appendices describe *Matlab*, the *Borland* compiler, the *Linux/Windows* command-line C++ compiler and profiler, calling **FORTRAN** programs from C++, and the C++ coding standard. Each chapter ends in assignments for which no solutions are given.

After telling how to install supplied software, Yevick provides the famous **Hello world** program commonly used to introduce the C programming language and its extension, C++. I compiled the program with *Dev-C++*. I also compiled it with the command-line version of *Borland Turbo C++* version 3.0 and adapted it to *Microsoft Quick C* for *MS-DOS*, both of which I have had for years. I was able to reduce the sizes of the *Turbo C++* and *Quick C* versions with *Diet*, a program which compresses executable files but leaves them able to run. The sizes in characters (or bytes) of the executable files produced by *Dev-C++*, *Turbo C++*, and *Quick C* were respectively 424514, 8646, and 5068. Other programs compiled with *Dev-C++* were also characteristically big, and roughly 2 to 3 times as big as those produced by the supplied *Borland C++*. It is obvious that using *Dev-C++* can contribute to "bloatware"--programs which use more resources, but perform no better, than programs made by other development systems.

The book covers a reasonably broad range of topics, including brief discussions of computer and software structure, object-oriented programming, arrays and matrices, differentiation and integration, finding roots of equations, differential equations, linear algebra, reference variables, pointers, dynamic memory allocation, memory management, static variables, pointers to static variables, Monte Carlo methods, parabolic partial differential equations, and combining **FORTRAN** and **C** codes with **C++**. There is discussion of techniques with example programs, and each chapter ends with assignments which are often oriented to physics problems.

The *Crimson Editor* proved helpful especially with the *Borland C++* compiler.

However, the text often discusses the procedures used in example programs but omits their purposes, causing me to spend much time trying to learn what was intended.

I often found that I had to modify supplied programs to make them run. Sometimes that happened because of the ways I had installed software, but there were also mistakes in the programs. For example, to make the program of Chapter 22, Section 2, suit *Dev-C++*, I had to include `float.h` and change `FLT_MIN` to `FLT_MAX`. The program of Chapter 13 is supposed to integrate  $x*x$  (the square of a variable) between user-supplied limits, but does so only if the lower limit is zero. I corrected that problem. There is confusing use in programs of the same symbol for the numeral 1 and the lower-case 'L'. The program for Monte Carlo integration, in Section 1 of Chapter 22, gave usable results in my trials only when the integration limits were negatives of each other, like -5 and 5. In such cases, the results were always three times the correct results. In other cases, I could often not correlate the results with the correct ones. I tried unsuccessfully the example in Appendix D of calling a *FORTRAN* program from **C++**.

Occasionally, there were differences in the requirements or results of *Dev-C++* and *Borland C++*. The exercise on pages 164 and 165 produced zero-length output files when compiled by *Dev-C++* but made output files with content when compiled by *Borland C++*. I compiled the program of pages 328 and 329 with *Borland C++* and ran it, but could not do so with *Dev-C++*.

A few items in the main text need correction. Page 15 says using the keys **<Ctrl> <Print Screen>** captures the entire screen display to a part of memory, the clipboard. In fact, **<Print Screen>** alone does that. The characters `\` and `'` are sometimes confused. The term 'page fault' seems to refer to what is usually called 'virtual memory'--use of a disk for temporary storage.

The source code **section18-tree.cpp** is provided on the compact disc for both Chapters 16 and 18, but it corresponds to Section 19 of Chapter 18.

Section 6 spends much time on the "industry-standard" *Rational Rose* package, which is not supplied with the book. I am not familiar with *Rational Rose*. Note that computer people

commonly treat the term "industry- standard" as a joke because of the great number and variability of standards.

Section 6 of Chapter 7 discusses converting C-language code to C++. Since C++ is a superset of C, conversion seems unnecessary, though occasionally C code may need a few changes to suit a C++ compiler.

This book would benefit from a revision to correct mistakes, elimination of references to *Rational Rose*, addition of discussion of purposes of programs, and inclusion of electronic versions of all example programs and of solutions to assignments

Nevertheless, there is still much of value in Yevick's book for someone already familiar with C or C++ programming. I prefer to stick with development systems which are less demanding than *Dev-C++*, but perhaps some people need to use it. Experienced programmers may be able to adapt the discussion of Yevick's book to other compilers.

One last note: I am never sure how to say 'C++'. I usually hear it as "C plus plus" but in the C and C++ languages, '++' is an increment operator, and 'C++' is a complete sentence meaning 'After C is used, increase its value by 1'. Perhaps 'C++' should be pronounced "C plus 1" to refer to its being an extension of C.

David P. Maroun  
Chilliwack, British Columbia, Canada