

Forward

Why Another C++ Book?

An Introduction to C++ is a textbook with the purpose of introducing high school students to computer science using the C++ programming language. In addition, chapters 25, 35, 36 and 40 through 44 specifically can be used by students that find themselves in an environment that expects knowledge of linked lists, stacks, queues and binary trees rather than merely using standard library template classes. Another audience could be students that studied Java programming, but now find themselves at a University primarily using a C++ environment. It could be a text for older students wishing to learn the fundamentals of C++ but who possess a modest background in computer science, but it is not intended to be a text for all people in all circumstances and such students are referred to their local university bookstore.

This text includes the basic syntax of C++, strings, functions, classes, control structures and file I/O. It introduces data structures including arrays, structs, dynamic lists and trees, along with the basic algorithms used to manipulate them. It includes only one chapter on using a container class (Chapter 34 on Vectors), preferring to teach computer science rather than just programming. Some Microsoft Windows programming issues are also included, but are not seen as central to the text. (Students familiar with the concepts of object orientation and who wish to learn more about programming with C++ for Microsoft Windows are referred to Nitty Gritty Windows Programming with C++, by Henning Hansen, published by Addison-Wesley.)

A Text should Fit the Circumstance

There are many excellent programming textbooks that seek to introduce students to computer science using the C++ programming language. Why another one? In truth, this text is the direct product of my personal frustration. In my opinion, the best available texts that introduce C++ and computer science are intended for university level programming classes. Such texts tend to have a great many pages spread between few chapters. Often, concepts are presented in a variety of ways in the same chapter in order to suit the learning styles and background of a diverse student body. A college student may go to lecture two or three times a week for less than 3 hours on average. College programming assignments are generally done in a large lab or private setting, where the instructor is generally not available. In such circumstances, textbooks become prime sources of examples and additional explanations of arcane concepts and knotty algorithms. When confronted with an intractable problem, a college student may seek out the professor during office hours, but the professor is almost guaranteed to not be available late at night or during the wee hours of the morning when the student is actually trying to program. The text in this type of class is expected to help the student during this time.

On the other hand, high school students generally go to class every day, with the lab time being included in the class. More importantly, the teacher is almost always available during this time for fairly immediate help. While college students may spend lecture time taking notes, high school students are more likely to be given a short

explanation and an example that can be directly implemented. High school students also spend more of their time each day in class and have less time to devote to reading large texts.

For of these reasons, this text has been written in many short chapters. The chapters are short because multiple explanations for concepts are avoided. Instead, a concept is explained, syntax is given, then an example is provided that the students are expected to immediately implement. Of course, most chapters contain question exercises and almost all contain programming problems.

Data Structures, Strings and Flow Charts

In has been my experience that programming books for high school students tend to shy away from some string concepts, object orientation and data structures beyond arrays. Especially as regards data structures, this trend has been aggravated in recent years in part because of the choices made by a popular testing service. In this text, C-string concepts are introduced early in a minimal way, then revisited as necessary to make more advanced concepts available. About 1/4th of the way into the text, students learn to create programs in multiple files, create classes and use classes created by others. About 3/4^{ths} of the way into the text data structures beyond arrays are introduced, along with basic algorithms to access them.

This text does not contain a unit on flowcharting, nor does it use flowcharts as examples or explanations. I have nothing against flowcharting (and have used it and other techniques often as a programmer), except that it just does not seem to help novice programmers understand programming logic. (I refer you to the work of Dr. Ben Schneiderman, especially Software Psychology : Human Factors in Computer and Information Systems; Winthrop Publishers, Inc., Cambridge, MA; 1980.)

How the Book was Tested

I have used the material in this text to teach high school freshmen through seniors. The prerequisites for success have tended to be (1) interest on the part of the student and (2) a good background in symbolic reasoning, such as found in basic Algebra. A background in non-object oriented 3rd generation languages such as BASIC, Pascal or C helps in such things as decisions, loops and functions, but it has also been my experience that students with extensive backgrounds in such languages may struggle converting to object oriented programming and design concepts. For many computer science teachers, this is not a surprise. Those of us who remember FORTRAN and COBOL and the consequent struggle to convert to Pascal, C and PL/1, then object oriented languages such as Java and C++ can appreciate the difficulties in "switching gears".

Organization of the Text

This text is written to be covered in chapter order, with a few exceptions. If the class meets 45 to 55 minutes per day, chapters 1 through 24 make a good semester length course for motivated students with good math skills, the remainder of the text through chapter 46 taking up another year. If classes are longer or meet more often, this time frame can be reduced.

The organization of the text is as follows.

- 1) **Chapters 1 through 24** cover the basic features of object-oriented programming and the C++ programming language. Chapter 14 (operator overloading) may be delayed until needed. These include
 - a) basic C++ program organization
 - b) types
 - c) identifiers
 - d) constants
 - e) object declaration and use
 - f) assignment
 - g) input and output
 - h) functions and scope
 - i) parameter passing
 - j) programs in multiple files
 - k) classes
 - l) Boolean expressions
 - m) decision structures
 - n) repetition structures
 - o) file input and output

No data structures are introduced at this time in favor of students acquiring greater experience in programming with and writing functions and classes, which are the fundamental organizational elements of object oriented programming.

- 2) **Chapters 25 through 45** concentrate on data structures and the algorithms that use them. Although additional C++ syntax is introduced at this time, the main emphasis of these chapters is on the basic ideas of the modeling and manipulation of data so central to computer science. In this section, chapters 27 (Typedefs), 34 (Vectors), 36 (Sparse Matrixes) and 39 (Heuristics) can be considered optional material and may be skipped or covered later.

a) Core Topics Covered

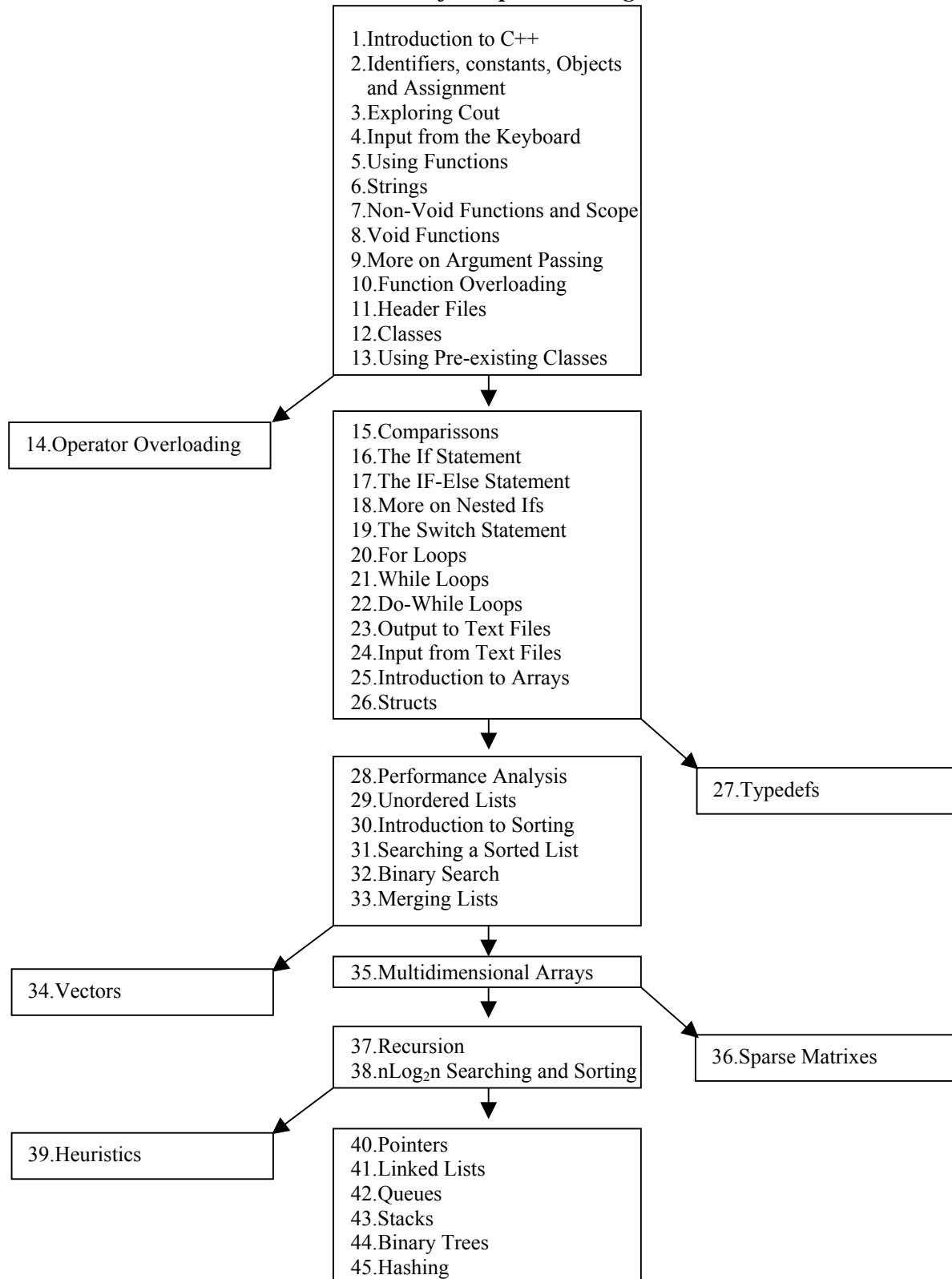
- i) arrays (simple and dynamic), single and multi-dimensional
- ii) structs
- iii) worst case performance analysis
- iv) unordered and ordered lists, including sorting, searching and merging
- v) recursion

- vi) pointers and dynamic objects
- vii) linked lists and operations on them
- viii) queues and queue operations
- ix) stacks and stack operations
- x) binary search trees and traversal and searching binary trees
- xi) hashing

b) Optional Topics Covered

- i) typedefs
 - ii) vector class
 - iii) sparse matrixes
 - iv) introduction to heuristics
- 3) **Chapter 46** contains two larger examples of programs intended to either serve as "walk through" instructional examples (i.e. case studies) or as ideas for larger projects for advanced students complete with example solutions.
- 4) **Chapters 47 and 48** contain very brief introductions to writing programs for Microsoft Windows using Microsoft Visual C++.
- 5) **The Appendix** contains various brief explanations on questions that can arise when working with groups of bright students. It is by no means comprehensive.

A note to teachers planning to use this text in preparation for the College Board's Advanced Placement Exam in Computer Science A or AB: There are case studies that need to be reviewed with students and that these case studies use container classes that are provided to perform operations on most data structures.

Order of Chapter Coverage

Weaknesses of the Text

The text is currently short exercises in many of the chapters and some of the chapters could use alternative programming assignments. The key to the existing exercises and the programming assignments only partially exists at this time and will be posted on my web site over the next few months.

In Closing

I welcome your comments, corrections, suggestions, exercises, programming assignment ideas and would appreciate hearing about your experiences using this text. Any corrections, exercises and programming assignments sent to me for use with this text will be posted on my web site with full attribution.

Frank D. Ducrest
fdd@louisiana.edu
<http://www.louisiana.edu/~fdd5501>
University of Louisiana at Lafayette
May 21, 2002