Pascal Programming for Music Research
(Chicago: University of Chicago Press, 1990)

by Alexander R. Brinkman

Reviewed by James L. Snell

Occasionally a book appears which fills a major and long-standing gap in the literature, and which documents a methodology for an entire area of study comprehensively for the first time. This is such a book. The technical methodology of computer-aided research in music theory and musicology, from the pioneers (e.g., Rothgeb, LaRue, Jackson, Forte, and Kassler[1]) through the present, has been anything but standardized. Almost every researcher has had to invent, and in most cases reinvent, appropriate internal computer representations for even such basic items such as pitches, durations, and intervals, not to mention part

---

[1]John Rothgeb, *Harmonizing the Unfigured Bass: A Computational Study* (Ph.D. dissertation, Yale University, 1968); Jan LaRue, "Two Problems in Musical Analysis: The Computer Lends a Hand," in *Computers in Humanistic Research: Readings and Perspectives*, ed. E.A. Bowles (Englewood Cliffs, New Jersey: Prentice-Hall, 1967): 194–203; R.H. Jackson, "Harmonic Analysis with Computer: A Progress Report," *Institute for Computer Research in the Humanities Newsletter* I (1966): 3–4; Allen Forte, "The Domain and Relations of Set-Complex Theory," *Journal of Music Theory* 9/1 (1965): 173–180; Michael Kassler, "A Sketch of the Use of Formalized Languages for the Assertion of Music," *Perspectives of New Music* 1/2 (1963): 83–94.

synchrony or complete compositions. The reasons for this duplication of effort were partly (1) that various modifications of the basic techniques had to be developed for different kinds of analysis, necessitating some reinvention, but more importantly (2) that technical methodologies were usually not described in the literature in sufficient detail for re-use. Brinkman's book does an enormous service to the computer-aided music research community by presenting carefully refined techniques for a wide variety of applications, exposited clearly and in complete detail.

But the book's coverage of research techniques, while its most valuable contribution, is only one of its aspects. It also incorporates numerous features that make it an exemplary textbook, suitable for graduate students or advanced and strongly motivated undergraduates. These features include: a chapter on concepts of computers; four chapters introducing the Pascal programming language; a section on general programming methodology; nine chapters containing rigorous treatments of the more advanced features of Pascal, using examples relevant to music analysis; numerous references and exercises at the ends of chapters; two appendices useful for the beginning Pascal programmer; an extensive glossary; and two indexes (one to computer techniques, one general).

Yet a third valuable aspect of this mammoth book is its inclusion, in three appendices, of a sizable collection of ready-to-use software: an interpreter for a large subset of the DARMS score-encoding language; a library of general utility programs for music analysis; and a group of programs for score processing, i.e.,

manipulating the data structure generated by the DARMS interpreter. Researchers wishing to use these programs can transfer them to a computer using a text scanner (typing them would be excessively time consuming), or obtain them on a diskette by contacting the author at the Eastman School of Music.[2]

The book is organized as follows:

---

[2]Personal communication.

APPENDICES
A. The ASCII Character Set
B. Type Compatibility and Operator Precedence
C. A DARMS Interpreter
D. A Program Library
E. Score Programs from Chapter 20

Glossary
Index to Programs, Subprograms, and Algorithms
General Index

Research Techniques

As Brinkman notes in the Preface:

> . . . much of the important work in music theory, especially the analysis of twentieth-century music, has relied heavily on techniques for modeling music and testing theories with the aid of the computer. (p. xv)

Besides atonal research, studies in tonal music theory and in musicology have also increasingly benefitted from the computer as a tool, both for its "low-level" capacities to count, sort, format, compute statistics, etc., and for "higher-level" tasks such as parsing and generating compositions according to formal grammars. But despite the benefits afforded by using computers in music research, their use has been quite limited. In part this has been due to ordinary human fear or misunderstanding of new technology. But even among music researchers who would like to begin using computers, there have been at least three major impediments: (1) lack of programming knowledge by some; (2) lack of refined programming techniques and ready-made software usable by non-expert programmers; (3) lack of computer-encoded scores available to researchers "off-the-shelf." Brinkman's book certainly

addresses issue (1), in that it is uniquely suitable for self-instruction (as well as for classroom instruction—see below) by musicians, who otherwise would be forced to study from conventional computer science texts filled with examples having no relevance to music.

But the book's greatest contribution for researchers is in the area of item (2): programming techniques and finished programs. Brinkman begins his serious treatment of music-related techniques in Chapter 6, presenting six different methods for representing pitch, ranging from simple pitch class (pc) using integers 0–11, to continuous binomial representation (cbr) which encodes octave, pitch class, and diatonic spelling; he also includes a complete list of operations on these pitch representations (transposition, inversion, interval calculation, etc.). Similarly, for representing time intervals he presents reciprocal duration code (rdc), which he later generalizes to rational duration representation (rdr), along with the associated operations. In every case it is clear that Brinkman has thoroughly thought out his choices of representation to achieve optimal flexibility and ease of use.

Finally in this chapter, he discusses encoding of other musical features (bar lines, articulation, part indication, etc.) as he introduces two widely used score encoding languages: DARMS (Digital Alternate Representation of Musical Scores), and the much simpler language SCORE. It is important to note that the book is not primarily concerned with these external (i.e., character-based) score languages; rather, its focus is on building and processing the internal data structures needed for music analysis, which are independent of the external form in which the music was encoded.

In Part II of the book, interleaved with explanations of general-purpose data structures and algorithms (arrays, records, sets, lists, etc.), Brinkman develops a number of useful musical applications, building on the pitch and duration representations given earlier: processing twelve-tone matrices, pitch-class set operations, processing grouplet durations, computing permutations and combinations of pitch-classes, melodic contour counting, and thematic indexing. These applications are carefully ordered in increasing difficulty, and in each case, the musical application is appropriately matched with the data structure being discussed.

The four chapters of Part III present more advanced music-processing applications, while continuing to add to the general repertoire of data structures and algorithms (binary search trees, bitmaps, etc.). Chapter 17 covers algorithms for four definitions of normal order and prime form (those of Starr, Forte, Alphonce, and Rahn). Brinkman carefully analyzes the algorithms in terms of both musical utility and computational elegance and time and space efficiency, and presents complete programs in Pascal, painstakingly explained and commented. Chapter 18 deals with the problem of searching a matrix for specified pitch-class collections. As is typical throughout the book, Brinkman provides not merely a program that does the job, but one that is graphically effective (using reverse video to highlight matches), flexible (the user can specify absolute or 0-based display of the matrix), and user-friendly (it presents a convenient menu of commands for various operations). Brinkman has refined his search and display

algorithms for optimal efficiency, and carefully explains and motivates their designs.

Chapter 19 presents an instructional program that provides drill and practice in pitch spelling, while developing algorithms for generating random numbers and manipulating pitch collections. The program provides an exhaustive repertoire of exercises: forty-one categories of practice, organized in six groups (intervals, scales, scale steps, chord spelling, diatonic chords, and altered chords), 283 distinct questions, and over 87,000 question variations. Anyone concerned with the pedagogy of music theory will be impressed by the features of this program. As usual, Brinkman explains every part of the program thoroughly, for those who wish to modify or expand it.

Chapter 20, the final chapter, is Brinkman's *tour-de-force*: a versatile software system for processing musical scores. He first gives a detailed explanation of his DARMS interpreter, a program that translates DARMS-encoded scores into a data structure suitable for analysis. He then presents programs illustrating the use of this data structure for various purposes, including matching melodic contours, finding arbitrary pitch sets, and performing simple harmonic analysis. Here we see, for the first time in a widely-available and practical form, the realization of an idea that goes at least as far back as the invention of DARMS by Stefan Bauer-Mengelberg in the 1960s:[3] software that accepts musical

---

[3]Stefan Bauer-Mengelberg, "The Ford-Columbia Input Language," in *Musicology and the Computer*, ed. Barry S. Brook (New York: City University of New York Press, 1970): 48–52; see also Raymond F. Erickson, *DARMS: A Reference Manual* (New York: DARMS Project, Dept. of Music, Queens College, CUNY, 1976).

scores encoded in virtually total detail, and performs useful analysis of arbitrary complexity automatically.[4]

This culminating chapter can be taken as a cue to those music theorists who have been waiting for these foundational programs to become available in order to perform analyses of a type and scale that are simply unrealistic to do by hand. Make no mistake: this is not slick, packaged software for use by the computer-illiterate—its use requires programming, but at a level that should be within reach of many music theorists. But these programs are rigorously thought through and skilfully implemented; they are splendidly explained and documented, allowing relatively easy adaptation to any project; and they are truly portable in the form of Pascal source code (one of Brinkman's chief reasons for choosing Pascal) to virtually any computer. For these reasons, they are likely to become *de facto* standards over the coming years.

One major impediment still remains before very large-scale, computer-based music analysis projects become practical: lack of availability of many computer-encoded scores (item (3) above). Part of the problem is technical: encoding scores into DARMS manually is slow, tedious, and error-prone, and the technology necessary for automatic encoding using optical scanning has yet to be perfected, although there has recently been progress in this

---

[4]Designs of two related software systems, one to translate from all the allowed variants of DARMS code into a canonical form, the other to generate a completely general internal score representation, are described in: Bruce A. McLean, *The Representation of Musical Scores as Data for Applications in Musical Computing* (Ph.D. dissertation, SUNY-Binghamton, 1988).

area.[5] But as much of the problem is organizational: the number of researchers who could use computer-encoded scores has not yet reached "critical mass," and there is thus no clearinghouse for even those few scores that have been encoded. One hopes that Brinkman's book will stimulate computer-based score analysis enough that the research community will organize DARMS sources, standards and methods for quality control, etc., which will in turn stimulate further encoding, and eventually "explode" into a commonplace form of research.

Pedagogy

Brinkman's book is exemplary in its thoroughness and clarity, as much for computing topics as for musical ones. Indeed, this book contains as excellent a set of explanations for teaching basic computer science—machine concepts, language constructs (types, looping, etc.), data structures (from arrays through linked structures and hashing), algorithms (sorting, searching, etc.), and single-author programming methodology (top-down design, debugging, etc.)—as any beginning computer science textbook on the market. The explanation of linked data structures (pp. 548ff.) is particularly fine. Brinkman's treatment of the Pascal programming language itself is more thorough than that in many computer science texts, including such esoteric features as internal files and parameter passing of procedures and functions. Another refreshing aspect, absent from many computer science texts, is

---

[5]See David S. Prerau, "DO-RE-MI: A Program that Recognizes Music Notation," *Computers and the Humanities* 9 (1975): 25–29.

Brinkman's frequent mention of common programming pitfalls, i.e., what possibilities are *in*correct, and why (e.g., pp. 233ff.). Finally, no student could wish for more understandable programs: they are virtually all, from small examples through large applications, properly structured, optimally efficient, beautifully formatted, and clearly commented.

The careful ordering of topics results in a progression of concepts and techniques that build smoothly throughout the book. In general, exercises are precisely worded, and provide a good range of levels of difficulty, particularly at the "difficult" level: despite the myriad program listings provided, plenty of possible enhancements and variations remain for use as student projects. The references and selected readings at the end of each chapter are appropriately complete and helpfully annotated. (A minor complaint is that the references are not also collected into one place at the end of the book, or at least indexed together by author.)

One feature of the book that makes it particularly suitable for self-teaching is the glossary. Over twenty-four pages, Brinkman has produced a remarkably complete and well cross-referenced list of terminology with definitions and often extensive examples, in both music and computing, covering the terms used in the book as well as many others.

Production

This book is one of the rapidly increasing number of those provided to the publisher in camera-ready form by the author; but while some books published in this way suffer from noticeably

inferior production, Brinkman's book does not. The typography, musical score examples, and graphics (mostly diagrams of computer structures) are all of a quality indistinguishable from that of conventional book production. The typeface used for Pascal program code (other than keywords) is a little lighter than one might wish, but this is not a serious problem.

Minor errors

In the first edition of any technical book of this size (nearly 1000 pages), it is not surprising that some errors escaped the notice of the editors. It is crucial to note that I am *not* speaking of errors in the program listings: although I have not tested the programs, I have little doubt that they are correct, since they were typeset directly from files on the computer on which they were executed.[6] Rather, I am referring to occasional typographical and other inadvertent errors, points of ambiguity, and minor errors of fact, which are few and harmless enough not to cause serious problems for students, and which will doubtless be corrected in subsequent printings. The frequency of the errors I found was much greater in the glossary than elsewhere; this is presumably due to the fact that definitions carry an implicit claim of absolute precision, and are thus inherently more likely to be flawed than ordinary exposition.

To illustrate how minor the factual errors are, I will list a few examples from the main text:

---

[6]For customary legal reasons, Brinkman includes no warranty of correctness for these programs, but this omission should not be taken as indicating lack of confidence.

Pg. 4, para. 3: In fact, floating-point numbers are usually represented by either 3 bytes for significant digits and 1 byte for the exponent, or 6 and 2 bytes, respectively.

Pg. 27, para. 1: In fact, *all* are optional, including statements.

Pg. 44, box: Pointers are not structured types, so the box caption could be misleading.

Pg. 59, line 2: On most machines, the smallest integer that can be represented is not *-maxint*, but *-(maxint+1)*.

Pg. 63, bottom: It isn't the compiler that wastes time determining that (*expression = true*) is *true* or *false*. Rather, the compiler wastes time (and space) generating code to compare the Boolean expression with the Boolean constant; then at run time, more time is wasted when the comparison is actually executed.

Pg. 88, exercise 2: The value of *e* should be 2.718282 rather than 2.718284.

Pg. 168, note 2: An interpreter does not do "compiling and running. . . in one step"; it parses each computational primitive, looks it up in a table, substitutes values, and calls an execution routine. There are also program processors that do compiling and running in one step, but these are not interpreters.

Pg. 441, para. 1: It would be more accurate to say ". . . but no standard version of Pascal allows a function to return a non-scalar type."

Summary

Brinkman's book would be worthwhile simply as an exemplary computer science text, albeit with an unorthdox choice of applications. Of course it is not only that, but an excellent introduction to programming tailored to the interests of music students. But it is yet more: It represents a major advance in software tools for researchers in music theory and musicology. The musical encoding conventions, data structures, and algorithms he presents are, in most cases, not merely implementations of others' work, but original research results. The DARMS interpreter, in particular, is the result of several years of intensive program development, and the sections of this book covering it would by themselves constitute a substantial research volume. The DARMS interpreter and its associated score processing algorithms, besides providing long-needed standard tools for computer-assisted music research, set a high standard for program quality in this or any field of study.