Comp. & Maths. with Appls. Vol. 8, No. 2, pp. 165–166, 1982. Pergamon Press Ltd. Printed in Great Britain.

BOOK REVIEWS

Programmable Calculators, How To Use Them. By ROGER J. SIPPL and CHARLES J. SIPPL. Matrix Publishers, Champaign Illinois 1978. approx 148 figures, 526 pp., ISBN 0-916460-08-8, hard cover. Price $15.95

A lot has happened with programmable calculators since 1978. Nevertheless, this book can be of value, since most of the equipment it describes is still for sale. The authors describe programmable calculators in general, and many specific models in detail. They also discuss the hazy line between programmable calculators and computers, and explain when programmable calculators can be used instead of computers.

The book has an introduction, ten chapters, an appendix, and an index. Chapter 1, The Calculator: Its Progress, contains a discussion of the calculator market—past, present, and future. It discusses the use of calculators in schools, and describes several educationally oriented calculator products.

Chapter 2, Today's Calculator, contains a discussion of some available (but impressive) products: wrist watch calculators, pen calculators, talking calculators, etc. The chapter concludes with a section on "where we're going".

In Chapter 3 we see more novel products: a calculator that works with fractions, and more. There is a section on calculators with pre-programmed functions (i.e. statistical and financial operations) in which many specific machines are discussed. Much of this chapter seems to be taken directly from the manufacturer's literature. The chapter ends with an extensive glossary.

Chapter 4, Programmable Calculators: Hand Held Units, covers some basic features of programmable calculators. A sample problem is solved, and then several hand-held units are discussed in detail. Included are the HP-25, the SR-56, and the SR-52. The RPN and AOS entry systems are compared.

In Chapter 5, Several more hand-held programmable calculators are discussed: The TI-MBA, HP-92, and HP-29C. Coverage of the TI-58 and TI-59 is fairly extensive, although there are a few technical errors. Several programming techniques are introduced including flowcharting, the use of registers, indirect addressing, flags, and conditional branching. There is a very brief description of several personal computer systems, and then a section on the range of software available for hand held programmable calculators.

Chapter 6 contains a further discussion of the material in the previous chapter. The HP-67 and HP-97 are presented, and so is the concept of user-definable keys.

Chapters 7 and 8 discuss desktop programmable calculator systems. However, many of the specific products discussed are rapidly being displaced by microcomputer systems.

Chapter 9 talks about business applications. Several "case histories" are included to show the usefulness of programmable calculators. Chapter 10 does the same for engineering applications, and covers the use of programmable calculators in data acquisition systems and other bus-oriented systems.

Since the publication of the book, the market has seen the introduction of the HP-41C, and several pocket size machines programmable in a standard high level language (BASIC). These machines have alphanumeric displays and can be interfaced to external memories. This is only the beginning.

Programmable Calculators is a long book. It contains a lot of information and is aimed at a broad audience. Unfortunately, perhaps because of this, the book is often repetitive, and sometimes incomplete. It is definitely pitched a little low for the practicing scientist or engineer, but for the student or novice it is fine.

Washington University
St. Louis, Mo.

CREON LEVIT

A Programming Logic, by Robert L. Constable and Michael J. O'Donnell. Winthrop Publishers, Cambridge, Mass. 1978. 398 pp. $15.96.

The problem addressed by *A Programming Logic* is the achievement of a means for fully understanding complex computations. This understanding is to be founded in the development of a logic for reasoning about computer programs which is intuitive and understandable to the human user, yet formally and completely defined so as to be suitable for automatic verification. It is the capability of verifying a program that is considered key to systematic reasoning about programs.

The first seven chapters of the book present a formal logic of programs based on the first order predicate calculus. The syntax and semantics of a new formal system, PL/CV, are presented, along with proof rules and examples of proofs. PL/CV is a rigorous logical system developed by the authors to serve as a foundation for understanding programs. It was designed for instruction in verification-oriented programming, and the application of formal logic to the study of a program's structure and behavior. It consists of commands, expressed in the PL/CV command language (a subset of PL/CS, a PL/I dialect which enforces good programming practices), and assertions over intermediate states in a computation. Axioms and rules for a theory of programming over integers and strings are developed, as well as for elements of the command language. This formalization allowed the development of a deductive Proof Checker, which is employed for the verification of the correctness of the commands with respect to assertions encoded with them. A program written in the command language, with assertions embedded within it, is termed an "asserted program". The Proof Checker is not a theorem prover; rather, it uses the assertions supplied by the programmer, along with its embedded knowledge of the logic and the semantics of the command language, to verify the correctness of the program and its assertions.

The last hundred pages of the book consist of an appendix, serving as a reference manual for the PL/CV system implemented at Cornell University. This section deals less with the development of the formal logic than with the definition of a particular implementation and its use. The end of each chapter through the book includes exercises suitable for classroom use.

The topics addressed by this text are advanced, and require a familiarity with formal logic and, to an extent, the issues of formal semantics and program verification. It is well organized and presented, with many examples. The claim presented by the authors that the system is suitable for use by a naive undergraduate audience is, however, somewhat debatable. The degree of mathematical sophistication required for the understanding of the logic and the semantic definition of the command language places this text more in the realm of a graduate level course in the theory of computation or programming language theory. The concept, however, is a worthwhile one. It is encouraging to see a concern for the teaching of good programming methodologies and verification of programs early in the computer science education.

JOHN THOMAS LOVE

Washington University
St. Louis, MO 63130
U.S.A.