
This book provides an excellent method to learn about compilers. At an introductory, intermediate, and advanced level, it discusses topics central and related to the construction of compilers, many of which have previously appeared only in journal articles or publications not generally available. The work has only two faults. It was composed on an IBM 360 line printer using FORMAT and thus suffers typographically. It is not a cookbook for advanced compiler implementation. That is, when one attempts to build an advanced compiler with optimization, and when he attempts to find the answer to his pressing practical problems in the book, the answers may well not be there. In fairness, however, the book provides excellent indices and a very good historical and general bibliography that might aid the serious implementer.

After an introduction that describes the pieces of a compiler, the author discusses grammars and languages in a depth not usually found in a general-compiler book. He then proceeds to a discussion of finite automata and shows how the theory previously presented leads to a scanner (or lexical analyzer) as a copath of the parser. Next he discusses top down recognizers and the method of recursive descent. This is followed by a discussion of bottom up recognizers including precedence grammars, operator and higher order precedence, and bounded context methods. He introduces a production language for writing syntax recognizers and performing semantic actions. He follows this with an in-depth discussion of run time storage organizations, and the organization of, and data in, the Symbol Table. After dealing with various internal representations of the source program such as polish notation, triples, and quadraples, the author shows how semantic routines are used, first in general and then for Algol-like constructs. Finally, he deals with allocation of run-time variables, error recovery, code generation, and optimization. In addition, the author covers topics of peripheral interest not normally covered in such books, including interpreters, scanner constructors, the use of transition tables, the optimization of Boolean expressions, error recovery, the implementation of macros, and translator writing systems.

While this reviewer emphasizes that the book is not exhaustive, it is by no means sketchy and provides more than enough material for a half-year graduate course in compiler construction. The book is very well written and is suitable for self-study. It is also well motivated by the inclusion of a great deal of practical material. It is authoritative to the novice and intermediate compiler constructor and offers him a great deal of practical advice and help in getting into the material. Even commercial compiler writers could benefit from reading this text. Finally, the exercises are useful and numerous. However, one might prefer that in some cases they had been included in the material discussed instead of being given as exercises.

With all points considered, I believe this book, as the Knuth series, should find its way onto the bookshelves of all serious computer scientists.


It is unusual these days to find a product that does more than expected. Mr. Groner's book not only covers an appropriate subset of the PL/I language, but also imparts much wisdom from the field of scientific computing. It does much to bridge the wide gap between the functional understanding of a computer language and the skill necessary to effectively use it in solving problems. The reader is assumed to be knowledgeable in engineering or the sciences, but a beginner to computer programming.

A sample problem of heat transfer calculations is first considered as a vehicle for discussing the planning and design phase of program development. Questions such as: "Should we check the validity of data entered into the program?" and "Should we build into the program or read it into the program?" are raised, making this chapter very enlightening. Advice based upon much experience in the field is given in answer to the questions. A realistic program has now been planned and carefully designed.

The following chapters describe the elements of the language useful in technological applications. The explanations are good and the development is logical. An abundance of meaningful examples illustrate the use of the PL/I language facilities as well as demonstrate techniques and considerations for problem solving typically learned from experience. Many problems are given for the reader to try his understanding. Their solutions are in the Appendix.

The reader should be aware that PL/I is a very large language with many facilities that permit it to be used in business applications and numerous other fields in addition to technological applications. The author, by necessity, selected a small part of the language for use in his book. Subsetting the language is a difficult task because it is a tradeoff between including more facilities and keeping the subset small for teaching purposes.

The use of computer files for the temporary or permanent storage of data was not included in the subset, even though they are frequently used in technological applications. The omission may be justified on the basis that the book is for beginners. On the other hand, a beginner rarely has need for bit string manipulation, which was included. The reader should be cautioned against the use of the fixed decimal data type. It is rarely of any special value in technological applications. In addition, fixed decimal data in arithmetic expressions are very apt to lose the desired precision if the Multiply and Divide built-in functions or other special techniques are not used. Fortunately, the use of fixed decimal data was not emphasized.

The highlight of the book is the last chapter where the reader learns about debugging and validating a PL/I program. A very realistic sequence of events is concocted in the development of a statistical analysis program. The program is submitted to the computer for the first time. The output from the computer is shown, including diagnostic messages that are explained by the author. The program is resubmitted several times; the output from each run is shown. A variety of errors is discovered and explained as the validation procedure progresses. The techniques of program validation are well presented and logically developed.