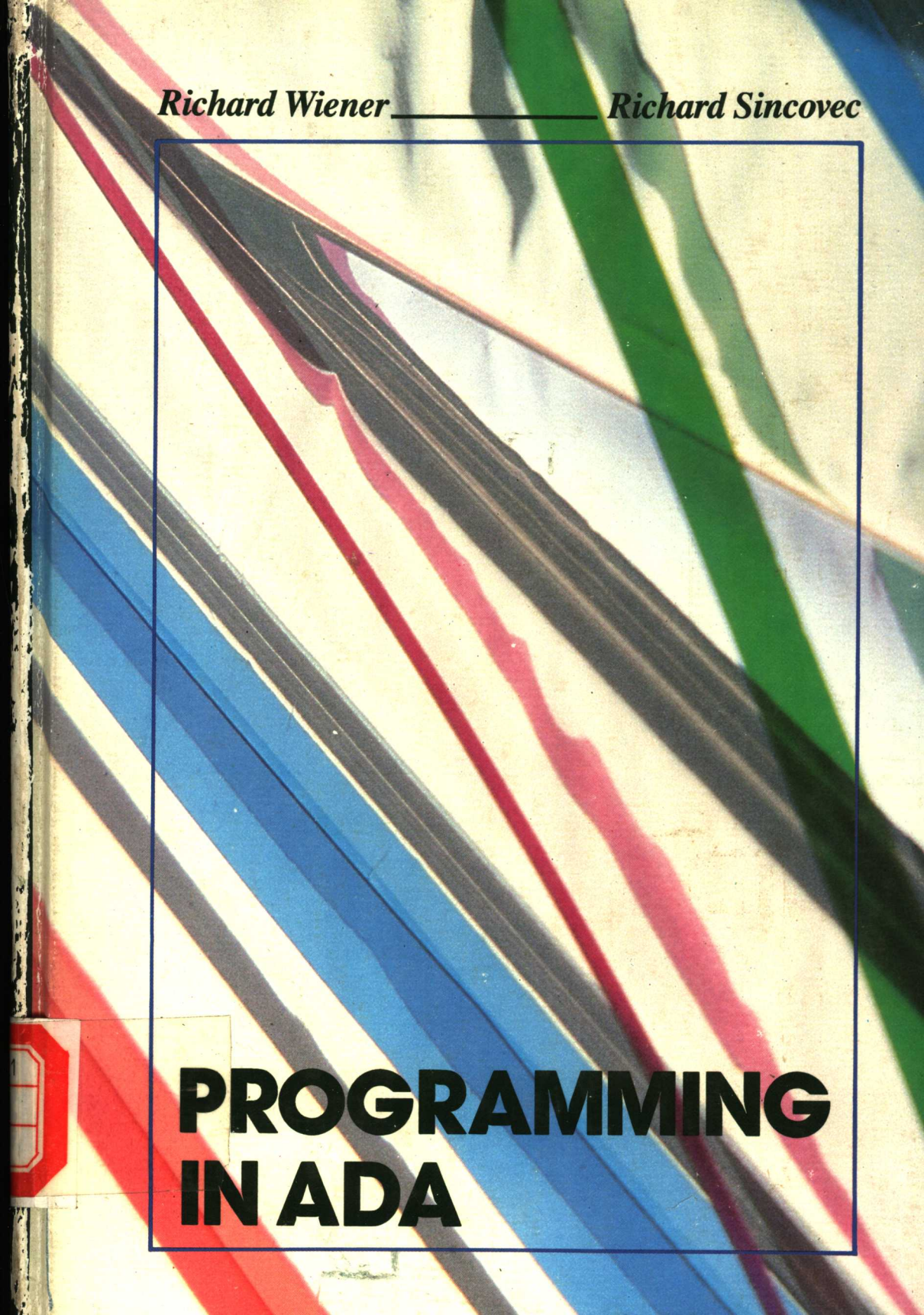
Richard Wiener _____ *Richard Sincovec*



PROGRAMMING IN ADA

Richard Wiener _____ *Richard Sincovec*

University of Colorado at Colorado Springs
Western Software Development

PROGRAMMING IN ADA

John Wiley & Sons
New York Chichester Brisbane Toronto Singapore

To our parents,
Irving, Mary and Frank, Kathryn.
And to our families,
Sheila, Erik, Marc
and Deanna, Mary, James.

Library of Congress Cataloging in Publication Data:

Wiener, Richard, 1941–
Programming in Ada.

Includes index.
1. Ada (Computer program language) I. Sincovec,
Richard. II. Title.
QA76.73.A35W53 1983 001.64'24 82-20046
ISBN 0-471-87089-7

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

PREFACE

The primary goal of this book is to introduce and illustrate the major features of a new programming language, Ada. The book is aimed at practicing computer science and data processing professionals and students of computer science. Ada's many features for supporting software development and maintenance make the language ideally suited for those involved in large scale software projects. Although Ada builds on many concepts of Pascal and PL/1, we do not assume that the reader has programmed in either of these languages. We assume that the reader has some prior programming experience in at least one high level language such as Fortran, Basic, or Pascal.

Relatively few Ada courses are being taught at colleges or universities because of the current unavailability of Ada compilers. With the imminent completion of Ada compilers, many computer science departments will begin to offer Ada courses, since this language can be used as a vehicle for introducing advanced programming concepts (e.g., data abstraction, data hiding, concurrent processing, complex scoping, programming environments) and for teaching software engineering. When the compilers become available, we believe that Ada should be first taught at the junior or senior level. Since many computer science departments have introduced the elements of structured programming and Pascal at the freshman level and have exercised these concepts in a sophomore level data structures course, a junior or senior level Ada course that incorporates principles of advanced programming and software engineering appears to us to be most appropriate. By the time students encounter Ada, they should be concerned about the methodology associated with the development of a large software project—Ada provides an ideal mechanism for teaching such methodology—and not just about writing a "correct" program. To be able to exploit the full power of Ada, students should have some skills in numerical analysis, data structures, and algorithm design.

Ada is a more complex language than many of the languages that are currently enjoying heavy use such as Fortran, Basic, and Pascal. The military language reference manual (Reference 2) and several early Ada textbooks (e.g., References 1, 3, 6) present the details of the Ada language in a formal way. The

relatively few complete Ada programs that are presented in these books are often short and sometimes trivial.

This book gives many nontrivial Ada programs to support the presentation of new Ada constructs and concepts. Application programs are presented in the areas of data structures, numerical analysis, and algorithm design. As we introduce new Ada features, some of the programs are updated to demonstrate the improvements in program design that the more sophisticated features support. We hope that the reader will acquire a faster and more meaningful appreciation of the scope and power of the Ada language by studying these Ada programs.

We employ an informal style of narrative, and we hope that the reader is not offended by our occasional attempts at humor. By concentrating on the major and significant language features and not getting bogged down on some of the fine and often complex detail(s) associated with a language feature (occasionally such details are omitted entirely), we hope to have enhanced the value of this book as a learning tool. We suggest that the reader use a current language reference manual as a supplement to this book, since we make no pretense of covering every fine detail of the language.

Chapters 1 through 8 present the basic control and data structures associated with Ada. The student who is experienced with Pascal or PL/1 should be able to go quickly through this material. The student who is unfamiliar with Pascal or PL/1 should study carefully the new concepts associated with the control and data structure features discussed in these chapters. Chapters 9 through 16 set forth powerful and advanced features of the language that set it apart from previous programming languages. Many of the concepts associated with advanced programming and software engineering are supported in the material of these later chapters, which present large Ada programs that illustrate the full power of Ada. We anticipate that the material of this book will support a one semester course in Ada programming.

We acknowledge the generous support of the Microsystems Institute, a subsidiary of the Western Digital Corporation, in helping us produce this book. In particular, we thank Alan Boal, president of the Microsystems Institute, for his confidence in us and his vision, without which this book would not be a reality. We thank William Carlson, president of the Advanced Systems Division of Western Digital, for his support. Both Western Digital and the Microsystems Institute have provided us with valuable resources and support. Many of the programs that appear in this book were tested using a Western Digital Supermicro (TM) computer and a MicroAda (TM) compiler.

Under the sponsorship of the Microsystems Institute we have been offering a series of Ada short courses for people in government and industry. Some of the useful suggestions that we have obtained from the students in these courses have been incorporated in this book. We also thank Kathleen D. Velick, of Western Digital, for her helpful suggestions and support during the production of this book. Robert Wilson, president of Hi-Country Data Systems, provided invaluable support during the production stage.

We are deeply grateful for the support of our families during the long and sometimes lonely hours that we spent writing.

Richard Wiener
University of Colorado at Colorado Springs
Colorado Springs, Colorado

Richard Sincovec
Western Software Development
P.O. Box 953
Woodland Park, Colorado

CONTENTS

LIST OF PROGRAMS

Chapter 1

TOP-DOWN VIEW OF ADA

1.1 INTRODUCTION

Ada is a programming language that was designed to satisfy a variety of programming requirements including the reduction in the overall cost of software systems. It is to be used in such numerical applications as large numerical and statistical simulation packages, and it is intended to be compatible with the mathematical and statistical software libraries of the future. The development of computer operating systems and compilers is also a goal that has been set for Ada. Finally, Ada is to be used in applications with real-time and concurrent execution requirements such as those found in the avionics system of aircraft or in the coordination of complex embedded computer systems. The language is considered to be a major advance in programming technology because it brings together the best features of earlier programming languages.

The structure of Ada is simple, yet its capabilities make it one of the most powerful programming languages. Ada contains features that should significantly lower the cost of software development and maintenance. These features include the option of separate compilation of program unit specifications and program unit bodies, software packages, generics, tasks to support embedded computer systems, overloading of operators, flexible scoping and visibility rules for data objects, subprograms, and strong typing of variables.

In this book we present the features of Ada in a systematic, easy to understand manner. That is, as we introduce each new feature of the language, we usually present a complete Ada program illustrating its use. Almost every program in this book was executed and checked on a Western Digital Microengine computer using their Micro-Ada compiler. Often the examples con-

sist of previous examples, reworked to demonstrate the use of a new language construct.

Anyone with programming experience should be able to grasp the essential details of Ada after one reading of this book. The approach we use to present Ada is carefully tailored to those readers with programming experience in almost any high level language. Our examples demonstrate the use of Ada on a variety of problems that arise in computer science, engineering, mathematics, and statistics. The programs should be easily readable, illustrating that Ada minimizes the slope of the learning curve for becoming familiar with software developed by others.

1.2 HISTORY AND THE PROBLEM ADDRESSED BY ADA

Ada was sponsored by the U.S. Department of Defense (DoD) in an attempt to reduce the rapidly increasing expense of military software systems. DoD identified language proliferation as a primary cause of the software problem. Custom languages and compilers were being developed for specific applications, but they often led to project failure because of inadequate languages and associated compiler problems. These factors prompted DoD, in 1975, to form the High Order Language Working Group (HOLWG). The HOLWG was charged with identifying and recommending solutions to DoD's language problem. Many existing languages were evaluated and found to be inadequate for the long term, and no language was found to satisfy the requirements for a common language. Additional studies indicated that a new language should be designed to meet DoD's requirements.

The language that evolved has become known as Ada. Ada is not an acronym like most computer language names. Rather it is the first name of Ada Lovelace, who worked with Charles Babbage on his difference machine. In today's terminology Ada Lovelace would probably be considered to be Babbage's programmer. That would make her the world's first female programmer. Some people say that Ada may be the last major high level language that will ever be developed, since automatic program generation techniques may be available in the not too distant future. Thus it seems fitting that the last major programming language should be named in honor of the first female programmer.

DoD is certainly not the only organization that has experienced the rapidly increasing cost of software systems. Many organizations have probably, on occasion, found their software development projects behind schedule, or the final development cost over budget, or the delivered program unreliable and/or not satisfying the original problem specifications. Another factor that has contributed significantly to increasing software cost is software maintenance. It is not unusual for the life cycle maintenance cost to exceed the original development cost. Anyone who has been involved in large software projects has probably experienced most of these dilemmas associated with computer software.

Other factors besides language proliferation have contributed to burgeon-

ing software costs. Another principal factor has been the inability to manage complexity. Structured programming, top-down program development methodologies, and program development and analysis tools have been used to deal with this problem. Studies have shown that this approach can increase productivity by as much as a factor of five when measured with respect to the debugged number of instructions per day that are produced. However, correct methodology does not force the programmer or the programming team to organize the complexity of the problem before program development is started. The use of high order programming languages is also advocated to reduce software costs. But it is well known that the use of a high order language alone does not necessarily increase productivity unless the language is properly used in conjunction with modern programming methodologies.

The high cost of software has been accentuated by declining hardware costs. It is not unusual for an organization to discover that software costs more than 75% of its total computing budget consisting of computer hardware, software development, and software maintenance. The trend seems to indicate that eventually hardware costs will comprise only 20% of the total computing budget, with software development and maintenance accounting for 80%.

The diversity of programming languages in common use requires that programmers become expert in several different computer languages. Thus the employing organization must hire programmers with different language specialties. Often programs developed in one computer language must be translated to other languages before they can be used on a newly acquired computer system or transported to the computer of another organization. Software developed in a computer language is often not portable from one computer to another or from one compiler to another on the same computer. Some software development tools have been developed to partially solve the portability issue. In any case, such problems have contributed significantly to software development and software maintenance costs.

Another fundamental problem is that as a programmer becomes fluent or expert in one particular programming language, he or she tends to develop software in that language regardless of whether the language is suitable for the problem at hand. The resulting program is often difficult to verify and usually not very readable, hence is also difficult to maintain. These factors have a significant impact on the life cycle cost of the resulting software.

Since some languages now in common use do not support modern program development methodologies, they also contribute to high software costs. That is, the use of such languages tend to result in late, error-prone software that is costly to maintain.

1.3 WHAT IS ADA?

Ada is a software engineering language. It is a high level, structured language that incorporates modern software engineering concepts within the features of the language. The language constructs encourage top-down program develop-