

Die Grundlehren der mathematischen Wissenschaften

in Einzeldarstellungen
mit besonderer Berücksichtigung
der Anwendungsgebiete

Band 137

Herausgegeben von

J. L. Doob · E. Heinz · F. Hirzebruch · E. Hopf · H. Hopf
W. Maak · S. Mac Lane · W. Magnus · D. Mumford
M. M. Postnikov · F. K. Schmidt · D. S. Scott · K. Stein

Geschäftsführende Herausgeber

B. Eckmann und B. L. van der Waerden

Handbook for Automatic Computation

Edited by

F. L. Bauer · A. S. Householder · F. W. J. Olver
H. Rutishauser · K. Samelson · E. Stiefel

Volume I · Part b

A. A. Grau · U. Hill · H. Langmaack

Translation of ALGOL 60

Chief editor

K. Samelson

Springer-Verlag New York Inc. 1967

Prof. A. A. Grau

Mathematics and Engineering Sciences Departments, Northwestern University, Evanston, Ill.

Dipl.-Math. U. Hill

Mathematisches Institut der Technischen Hochschule München

Priv.-Doz. Dr. H. Langmaack

Computer Sciences Department, Purdue University, Lafayette, Indiana/USA
und Mathematisches Institut der Technischen Hochschule München

Geschäftsführende Herausgeber :

Prof. Dr. B. Eckmann

Eidgenössische Technische Hochschule Zürich

Prof. Dr. B. L. van der Waerden

Mathematisches Institut der Universität Zürich

ISBN 978-3-642-86939-6 ISBN 978-3-642-86937-2 (eBook)

DOI 10.1007/978-3-642-86937-2

Alle Rechte, insbesondere das der Übersetzung in fremde Sprachen, vorbehalten

Ohne ausdrückliche Genehmigung des Verlages ist es auch nicht gestattet, dieses Buch oder Teile daraus auf photomechanischem Wege (Photokopie, Mikrokopie) oder auf andere Art zu vervielfältigen

© by Springer-Verlag Berlin · Heidelberg 1967

Softcover reprint of the hardcover 1st edition 1967

Library of Congress Catalog Card Number 67-14568

Titel-Nr. 5120

Preface

Problem oriented programming languages as they have developed over the last ten years essentially serve two purposes which somewhat crudely can be described by the terms man-man communication and man-machine communication, respectively. As a carrier of information between humans, the problem oriented programming language is designed to express the essence of an algorithm in a way which is unambiguous and concise as well as independent of (and therefore meaningful without any reference to) the changing details of computing machinery. As a carrier of information from man to computer, the language permits the human programmer to express his computational needs in a compact way adapted to the general characteristics of computers, but freed from the burdening details of specific computer facilities. This presupposes the existence of algorithms, or programs, which permit the computer itself to transform efficiently programs written in the problem oriented language into machine programs. Thus the entire computing community profits from the work of the individual programmer.

The primary purpose of the Handbook is to present a set of algorithms of broad utility from the domain of numerical mathematics written in the problem oriented language ALGOL 60. Therefore, volumes Ia and Ib are in a sense supplementary as they serve to introduce this language. Volume Ia gives a description of the language proper and of its use for writing correct programs. Thus, volume Ia primarily covers the aspect of man-man communication by means of ALGOL 60. By contrast, the present volume Ib is devoted exclusively to the aspect of man-machine communications. It presents a complete model of a translator program for ALGOL 60, an outline of the general principles applied, and a description of the operation of the program.

The underlying translator philosophy is an outgrowth of experience gained within the ALCOR group (an international group of computing centers and computer manufacturers collaborating in the development and application of ALGOL translators) which, over the years, has constructed ALGOL translators for so diversified a set of computers as ERMETH-ETH Zürich, ORACLE-Oak Ridge National Laboratory, PERM-TH Munich, Zuse Z 22/23, Siemens 2002, Telefunken TR4, Control Data 1604/3400/3600, IBM 7040/7090.

A large number of ALGOL translators has been developed in the past in different places with different starting points and different basic philosophies. No attempt has been made to give an overall survey, as this would have exceeded the scope of this volume. Considering the status of the art of translation of programming languages the book addresses the uninitiated who want to obtain access to methods of compiler writing; the expert may yet find some interesting material.

Volume Ia restricts the discussion essentially to the IFIP SUBSET ALGOL 60. Volume Ib, on the other hand, is far less restricted because it addresses a different set of readers, and because the subset excludes some features of ALGOL 60 which have proven to be of particular interest to programmers.

The model translator program itself is written in a language which is a slight extension of ALGOL 60, and therefore should be intelligible to anybody acquainted with this language. The object code produced by the translator is an abstract machine code modelled as close as seemed possible to the instruction codes for, in a certain sense, "average" present-day computers.

The authors wish in particular to express their thanks to Professor K. SAMELSON for his interest in the work and his constant readiness to discuss questions which arose during the writing of the manuscript. Thanks are also due to many colleagues in Oak Ridge and Munich who gave us valuable suggestions. Last but not least we are very much indebted to the secretarial staffs of our institutions; without their capable help the preparation of the manuscript would have been impossible.

Evanston, Munich, Lafayette, May 23, 1967

A. A. GRAU
U. HILL
H. LANGMAACK

Contents

1. Introduction	1
2. Principles of ALGOL translation	4
2.1. Basic linguistic definitions	4
2.2. The Backus normal form	5
2.3. The analyzing process	6
2.4. The method of the "Klammergebirge"	8
2.5. Recursive sequential methods and push down lists	9
2.5.1. The decoding matrix	10
2.5.2. Precedence rules and orders	11
2.6. Example for the use of two push down lists and of precedence rules	13
2.7. The concept of recursive translation	15
2.8. Organization of the translator	16
3. Languages involved in the translation process	17
3.1. Source language	17
3.2. Target language	19
3.3. Meta-language for describing the translator	24
4. Correspondence between elements of source and target language	26
4.1. Declarations in general	27
4.2. Declaration of variables and arrays and data storage allocation in the main program	27
4.2.1. Simple variables	29
4.2.2. Arrays and their information vectors	29
4.2.2.1. Static arrays	32
4.2.2.2. Dynamic arrays	32
4.3. Handling of types	34
4.4. Assignment statements	43
4.4.1. Simple arithmetic variables	43
4.4.2. Name variables, function designators, and type procedure identifiers	44
4.4.3. Subscripted variables	48
4.5. Boolean expressions	51
4.5.1. Truth values	51
4.5.2. Relations	52
4.5.3. Boolean operators	52
4.6. Conditional statements and expressions	53
4.6.1. The if clause	53
4.6.2. Conditional statements	54
4.6.3. Conditional expressions	55
4.7. For statements	56
4.8. Go to statement and switch declaration	59
4.9. Procedures and dynamic storage	62
4.9.1. Variables in procedures	63
4.9.2. Dynamic storage	64
4.10. Procedure calls and declarations	66
4.10.1. Procedure calls	66

4.10.1.1. Actual parameters in procedure calls	67
4.10.1.2. Procedure calls after transformations of actual parameters	72
4.10.1.3. Name calls	73
4.10.2. Procedure declarations	74
4.10.2.1. Value listed formal variables	74
4.10.2.2. Formal arrays	74
4.10.2.3. Function procedures	75
4.10.2.4. Normal procedure exit	76
5. Recursive address calculation	76
5.1. Introduction	76
5.2. Assumptions necessary for the use of recursive address calculation	79
5.3. Use of recursive address calculation for one loop	81
5.3.1. Method I: Difference method	83
5.3.2. Method II: Decomposition method	85
5.4. Nested loops	87
5.4.1. Method I: Difference method	90
5.4.2. Method II: Decomposition method	90
5.4.3. An example	93
5.4.3.1. Difference method	93
5.4.3.2. Decomposition method	95
5.5. Loops with more than one list element	96
5.6. Further optimization possibilities	98
5.6.1. Identification of subscripted variables	98
5.6.2. Generated variables	98
5.6.3. Use of index registers	98
5.6.4. Program organization	100
5.6.5. The example of 5.4.3	101
5.6.5.1. Difference method	101
5.6.5.2. Decomposition method	101
6. Run time organization	102
6.1. The instruction storage allocation	103
6.2. The instruction procedure call	104
6.3. The instruction formal procedure call	111
6.4. The instruction normal procedure exit	114
6.5. The instruction jump to	115
6.6. The instruction formal procedure exit	116
6.7. The instructions name address and name call	119
6.8. The instruction name procedure exit	121
7. Model translator. Description	126
7.1. Introduction	126
7.2. Pass 1. The preparatory pass	127
7.2.1. Input	128
7.2.1.1. Delimiters	128
7.2.1.2. Entities	130
7.2.2. Output	130
7.2.2.1. The modified input program χ_1	130
7.2.2.2. The identifier list	131
7.2.2.3. The for list	135

7.3. Pass 2. The implementation of recursive address calculation	136
7.3.1. Output	136
7.3.1.1. Program	136
7.3.1.2. The identifier list	136
7.3.2. Lists used in pass 2	137
7.3.3. Program survey	141
7.4. Pass 3. Decomposition and production of target program	141
7.4.1. Output	141
7.4.2. Program survey	141
7.5. Editorial functions	141
7.5.1. Syntax checking	145
7.6. Run time system. The target language program interpreter	147
8. ALGOL 60 model translator. Formal part	148
Pass 1: <i>preparatory pass</i>	148
Pass 2: <i>recursive address calculation pass</i>	198
Pass 3: <i>decomposition and generation pass</i>	239
Check routine: <i>check procedure calls and substitutions of formal parameters by actuals</i>	348
Check routine: <i>check agreeability of actual parameter and specification</i>	356
Run time system: <i>target language program interpreter</i>	358
<i>START TRANSLATION</i>	376
Bibliography	378
Index	387
Appendix: Correspondence matrix for actual and formal parameters	