BAUMANN, FELICIANO,
BAUER and SAMELSON

Introduction to Algol

PRENTICE-HALL
SERIES IN
AUTOMATIC
COMPUTATION

INTRODUCTION TO ALGOL

R. BAUMANN | *Mathematisches Institut der Technischen Hochschule München, Germany*

M. FELICIANO | *Mathematics Division Oak Ridge National Laboratory*

F. L. BAUER
K. SAMELSON | *Mathematisches Institut der Technischen Hochschule München, Germany*

PRENTICE-HALL, INC.

ENGLEWOOD CLIFFS, N.J.

Library of Congress Catalog Card No. 64-10740
Printed in the United States of America
47782 C

Revised and extended version of the ALGOL Manual of the ALCOR Group
Original Title: ALGOL Manual der ALCOR Gruppe
Elektronische Rechenanlagen *3* (1961) No. 5, 206–212
3 (1961) No. 6, 259–265
4 (1962) No. 2, 71–86
Oldenbourg, München, 1962

INTRODUCTION TO ALGOL

Prentice-Hall
Series in Automatic Computation

George Forsythe, editor

BAUMANN, FELICIANO, BAUER, SAMELSON, *Introduction to Algol*
DESMONDE, *Computers and Their Uses*
TRAUB, *Iterative Methods for the Solution of Equations*
VARGA, *Matrix Iterative Analysis*

PRENTICE-HALL INTERNATIONAL, INC., *London*
PRENTICE-HALL OF AUSTRALIA, PTY., LTD., *Sydney*
PRENTICE-HALL OF CANADA, LTD., *Toronto*
PRENTICE-HALL OF INDIA PRIVATE LIMITED, *New Delhi*
PRENTICE-HALL OF JAPAN, INC., *Tokyo*
PRENTICE-HALL DE MEXICO, S.A., *Mexico City*

PREFACE

ALGOL has found interest throughout the computing community. The defining report[1] was not intended to be a primer, however, since its aim of strict syntactic and semantic description did not allow this. This book is intended to give a needed introduction to ALGOL, which should enable the nonspecialist, for whose benefit ALGOL was primarily conceived, to write clear and readable ALGOL programs from which a reasonable translator will produce efficient machine codes.

Emphasis is on the normal use of the language rather than on artificial examples exploiting tricky possibilities; ALGOL has these as does any other language. The treatment of constructions which appear to have limited practical importance has been minimized. The use of certain features which would either conceal the form of a program or lead to machine code of doubtful efficiency has been ignored. It may be said that this voluntary restriction still leaves an extremely powerful language, acceptable to anyone who wishes to use ALGOL as a tool and not just for ALGOL's sake. A potential user will not find it difficult to learn this "normal use" of ALGOL. For didactic reasons we have not always kept to the syntactical formalities of the ALGOL report. Punctuation marks required in the text have been suppressed wherever they could be wrongly interpreted as ALGOL symbols. It would be impossible to cover the scope of applications and difficulties by means of selected examples; however, a small set of exercises has been included in order to point out common programming mistakes and important features of the language. An attempt was made to have Parts I and II correspond to the forthcoming IFIP Subset ALGOL 60.

[1] J. Backus, et al. Report on the algorithmic language ALGOL 60, Num. Math. **2**, 106–36 (1960), Comm. Assoc. Comp. Mach. **3**, 299–314 (1960).

J. Backus, et al. Revised Report on the Algorithmic Language ALGOL 60, Num. Math. **4**, 420–53 (1963), Comm. Assoc. Comp. Mach. **6**, 1–17 (1963).

The ALCOR group,[2] a cooperative association primarily interested in the construction of ALGOL compilers and on common hardware representations, found it desirable to provide the users with a common manual thus promoting program exchangeability. The manual arose from courses and lectures and grew during practical experience using and compiling ALGOL. This book results from revisions and extensions of that effort. In preparing it the authors are greatly indebted to H. H. Bottenbruch, W. Gautschi, A. A. Grau, A. S. Householder, M. Paul, H. Rutishauser, H. R. Schwarz, J. Stoer, K. H. Wiehle, and Chr. Witzgall.

The defining report of full ALGOL 60 is included as an appendix. The corrections and amendments recommended in April 1962 in Rome, Italy are incorporated in this revised version, which is an officially approved IFIP (International Federation for Information Processing) document.

MUNICH
OAK RIDGE

---

[2] At present, members of the ALCOR group are:
Institut für Angewandte Mathematik der Eidenössischen Technischen Hochschule, Zürich
Rechenzentrum der Technischen Hochschule München
Institut für Angewandte Mathematik der Universität Mainz
Institut für Praktische Mathematik der Technischen Hochschule Darmstadt
Siemens & Halske AG, München
Institut für Angewandte Mathematik der Universität Bonn
IBM-Forschungsgruppe Wien
Oak Ridge National Laboratory, Oak Ridge, Tennessee
Telefunken GmbH, Backnang
Zuse KG., Bad Hersfeld
Dr. Neher Laboratory of the Netherlands Postal and Telecommunications Services, Leidschendam
Standard Electric Lorenz AG, Informatikwerk, Stuttgart
IBM–Deutschland, Sindelfingen
University of Illinois, Digital Computer Laboratory, Urbana, Ill.
Eurocomp G.m.b.H., Minden
Remington Rand G.m.b.H., Frankfurt/Main
Control Data G.m.b.H., Frankfurt/Main
Purdue University, Department of Computer Sciences, Lafayette, Indiana
University of Western Ontario, Department of Computer Sciences, London, Canada
University of Maryland, Computer Science Center, College Park, Maryland
Rechenzentrum der Christian Albrecht Universität Kiel
Kommission für Elektronisches Rechnen der Bayerischen Akademie der Wissenschaft, München
Mathematisches Institut der Technischen Hochschule München
University of Michigan, Computing Center, Ann Arbor, Michigan

INTRODUCTION TO ALGOL

CONTENTS

INTRODUCTION

1. MODE OF OPERATION AND CAPABILITY OF DIGITAL COMPUTERS

The digital computer has astonishing capabilities as a tool for the experimental scientist. It is immediately evident that the use of such a tool requires thorough preparation. Accordingly, we shall first get briefly acquainted with the scope of this preparatory work, after which we shall investigate the methods of programming, that is, of setting up instructions for the machine.

The computer is no oracle. Without appropriate information it cannot answer the question how good will the weather be on the next weekend. With suitable preparation it is able to provide quantitative information which enables meteorologists to predict the weather on the basis of meteorological theories. The meteorologist bears the responsibility for it, and the prediction is as good as the theory which served as basis for the computation.

Put another way: the machine can do nothing that the user in principle could not also do; it can only do it faster. It gives no answers to vague questions and can work only on a precise set of instructions. Much work, however, is usually required to go from the original problem to be solved to this set of instructions. This work begins with the mathematization of the problem, that is, with the postulation of a mathematical model for the phenomenon under consideration.

Normally, this mathematical model consists of a number of conditions imposed on the variables which are used to describe essential parts of the phenomenon. In the case of the weather we have atmospheric pressure, temperature, humidity, wind velocity, etc. In general these conditions are not amenable to direct computational treatment. Methods must be found which permit calculation of values for the variables satisfying the conditions imposed on them. The development or selection of the method is the most important part of handling the problem. This must be done with great care since improper handling can yield completely misleading results.

Generally only approximations are computed because the exact solutions of the problem require numerically impractical passage to limits. Together with the development of the procedure one must then give an account of the goodness of the approximation. This usually clears the question whether the approximation corresponds to an acceptable modification of the originally conceived model.

An example of this is the use of difference methods to calculate the potential and current distribution in a conducting metal plate. This corresponds to the substitution of the plate by a grid of conducting wire. Sometimes the physicist can easily determine whether this model is useful, whereas it is very hard to establish strict mathematical limits to the error.

The preparation of the computational work starts once a method of solution has been selected. In simple cases this work is completely trivial. If, for instance, a formula given in a textbook or manual is to be evaluated, then it is best to undertake the evaluation directly. When an assistant is to do the work, he is given the book and the numbers needed for the computation. If the problem is more complicated, one must write down what should be done. The more there is to do, the more detailed the instructions, since less insight into the problem can be expected from the assistant.

The computing machine is merely an assistant which has no insight into what it is doing. Therefore, one must give it a set of instructions which clearly describes the process to the smallest detail. The machine executes these instructions word for word. In general it is not possible to predict what happens when one gives the machine an instruction that is not part of its limited repertoire. It may be that it stops working and points out that something is not in order. However, it may just as well continue working in a nonsensical manner. Accordingly, preparing the working procedure (the program) for a computing machine requires careful attention; in particular, a precise disposition and handling of all possible special cases.

2. PROGRAMMING

It is clear that we must write down the set of instructions that we want to give to the machine. However, we cannot use everyday language and expect the machine to receive instructions in the form of a letter or dictation. Instead we must adapt the language used to the capabilities of the machine.

Although the computing machine can be used in more general problems, in what follows we shall emphasize computation—more precisely, arithmetic computation. Arithmetic formulas which contain numbers, names denoting yet unknown quantities, and functions (such as *sin*, *cos*, and *ln*) have long been used to describe computational rules. Such formulas form a core embedded within a sequence of organizational statements which describe the flow of the computational process. Thus, for example, the execution of parts

of the computation can be made to depend on certain conditions, or one can prescribe the number of times a part is to be repeated. Indeed, the power of the *automatic* computer comes from its ability to make precise decisions at definite places during the course of the computation in accordance with preassigned criteria.

Finally, the machine must receive specifications as to type and dimension of the initial data entering into the computation (input data) and the numerical values given as the result (output data). A complete set of instructions and rules written in such a manner that it uniquely defines the course of a computation from beginning to end, we will call a *program.*

The preparation of the program entails more than due consideration to the arithmetic and organizational capabilities of the machine. The simple-minded intelligence of the computer requires that the language used be formed according to stringent rules. The present manual explains the standardized formal language ALGOL (*algo*rithmic *l*anguage) which arose out of an international effort. ALGOL programs are largely independent of the properties of individual machines and are conveniently readable to a wide circle of interested people. To an ever increasing extent algorithms and programs are being written and published in ALGOL.

3. CODING

In addition to what has already been said the program must be written in such a way that the machine can receive it. The input data as well as the program must be reduced to the specific code for input medium of the particular machine. To do this one uses off-line equipment—for example, punching mechanisms, printers, and hand punches. The machine receives the code through input equipment such as paper tape and card readers.

4. INTRODUCTORY EXAMPLES

We consider two simple examples which show certain fundamental aspects of the use of automatic computers.

EXAMPLE 1

Let us consider the system of linear equations in two unknowns

$$\begin{aligned} 2x + 3y &= 5 \\ 3x + 5y &= 4. \end{aligned}$$

Using Cramer's rule we obtain the following solution:

$$(1) \qquad x = \frac{5 \cdot 5 - 4 \cdot 3}{5 \cdot 2 - 3 \cdot 3} \qquad y = \frac{2 \cdot 4 - 3 \cdot 5}{5 \cdot 2 - 3 \cdot 3}$$

We shall, therefore, give instructions to the machine to compute the expressions on the right side of (1) and to assign these numerical values to the variables x and y. In order to simplify the code we use the slanted slash for the division sign. For the purpose of clarity we introduce the symbol "$\times$" for the multiplication sign; it must be used for each multiplication. Using these conventions we now write

$$x = (5 \times 5 - 4 \times 3)/(5 \times 2 - 3 \times 3);$$
$$y = (2 \times 4 - 3 \times 5)/(5 \times 2 - 3 \times 3);$$

This program yields the right solution, but at the expense of superfluous computation. We should not forget that the machine follows each instruction to the letter and for this reason computes the expression

$$5 \times 2 - 3 \times 3$$

twice.

Therefore, we amend the program to read

$$d = 5 \times 2 - 3 \times 3;$$
$$x = (5 \times 5 - 4 \times 3)/d;$$
$$y = (2 \times 4 - 3 \times 5)/d;$$

This is a special program for obtaining the solution of a definite system of equations. We get a program valid for the general case only if we admit arbitrary coefficients,

$$a_1x + b_1y = c_1$$
$$a_2x + b_2y = c_2$$

The corresponding program will read

$$d = a1 \times b2 - a2 \times b1;$$
$$x = (c1 \times b2 - c2 \times b1)/d;$$
$$y = (a1 \times c2 - a2 \times c1)/d;$$

However, this program does not take all contingencies into consideration. We must provide for the case when d vanishes for the given values of $a1$, $a2$, $b1$, and $b2$.

$$d = a1 \times b2 - a2 \times b1;$$
$$\text{if } d \neq 0$$
$$\text{then } \left\{\begin{matrix} x = (c1 \times b2 - c2 \times b1)/d \\ y = (a1 \times c2 - a2 \times c1)/d \end{matrix}\right\}$$
otherwise continue the computation
in some special manner;

Here we encounter an essential part of the program that goes beyond mere computation, namely, a condition (if $d \neq 0$) on which the further course of

computation depends, and specifications as to what must be done (then ... otherwise ...) on the basis of this condition. Notice also that the equality sign in the condition has a meaning different from that in other parts of the program.

EXAMPLE 2

In order to find the square root of a positive number a

$$x = \sqrt{a} \qquad a > 0$$

we can use Newton's method. This reduces to the following iteration

$$x_{n+1} = \frac{1}{2} \cdot \left(x_n + \frac{a}{x_n} \right) \qquad n = 1, 2, \ldots$$

Any positive number can be used as the initial value x_1.

The set of instructions is the same for each step of the iteration. It is manifest that it need only be written once. The result of each step serves as initial value for the next step. We can write the following program.

```
      y = x1;
2:    x = (y + a/y)/2;
      y = x;
      continue the computation at 2;
```

This program does not come to an end although it uses the iteration formula correctly. We must tie into the "continue" statement some condition, perhaps one that would interrupt the computation when two successive values agree to 10 significant figures. Here we introduce the notation $abs(x)$ which has the meaning of $|x|$.

```
      x = x1;
2:    y = x;
      x = (y + a/y)/2;
      if abs(x - y) > (.5 × 10^-10) × abs(x)
      then continue computation at 2
      otherwise end the computation;
```

We have already become acquainted with the conditional statement; here it is connected with the continue statement which brings about the repetition of a piece of program. Only in this way is it possible to describe completely the flow of the computation without knowing the values of a and $x1$ on both of which depend the number of required iterations.