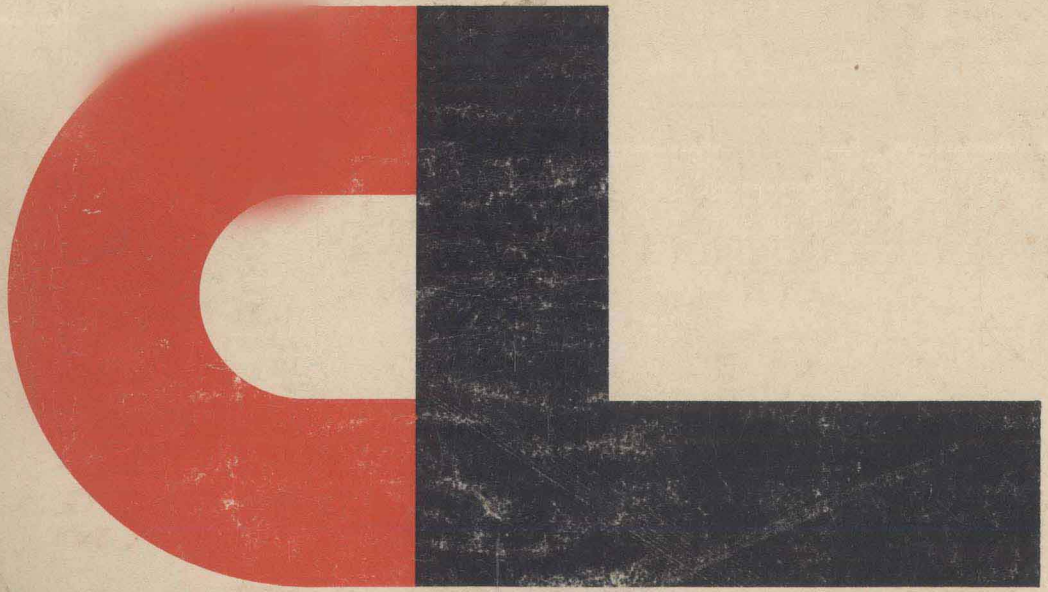


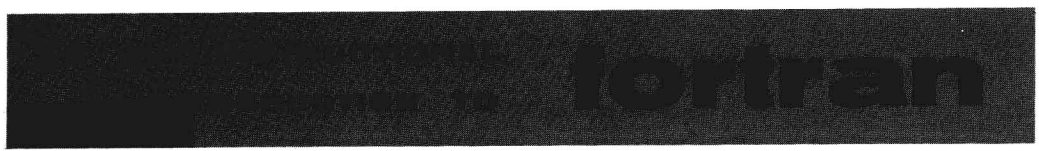
computer language

AN
AUTOINSTRUCTIONAL
INTRODUCTION
TO

fortran

McGRAW-HILL BOOK COMPANY

computer language



HARRY L. COLMAN
Armour Research Foundation,
Illinois Institute of Technology

formerly with
Western Data Processing Center

CLARENCE SMALLWOOD, *Educational Consultant*
Western Data Processing Center
Graduate School of Business Administration
University of California, Los Angeles

Foreword by
GEORGE W. BROWN, *Director*
Western Data Processing Center

McGRAW-HILL BOOK COMPANY, INC.
New York *San Francisco* *Toronto* *London* **1962**

Library of Congress Catalog Card Number: 62-17640

67890BB721069

computer language

Foreword

This autoinstructional text was originally designed to solve a difficult training problem at the Western Data Processing Center. The center serves academic research projects at the University of California, Los Angeles, and over seventy other institutions of higher learning in the thirteen Western states. Most users wish to use the Fortran Programming System even though they come from many different academic disciplines and their technical backgrounds vary markedly. The WDPC runs a Fortran Open Shop. Training problems at WDPC are similar to those met in most such open shops but exist here in exaggerated form. It would, for example, be extremely impractical to send instructors on monthly tours of seventy institutions. It is undesirable to have one person deliver the same lecture month after month, both because it is not sufficiently challenging for the instructor and because the audiences are so heterogeneous both in their interests and in their technical backgrounds.

Although the training problem at WDPC may be extreme, many of the relevant circumstances have counterparts at other academic and industrial installations. This text constitutes a self-contained training system for introducing students of almost any background or professional interest to the details of Fortran coding and the art of computer programming.

The system provides detailed information for students who have little or no familiarity with algorithmic notation but also permits students with this familiarity to skim that information efficiently. The examples and exercises are based on commonplace experience and do not require knowledge of a specialized application. Furthermore, the system contains an indexed reference manual that is cross-referenced to the program for periodic, voluntary review. Upon completion of the program, students will have written their first Fortran program. This program can be keypunched and compiled and executed. A short period of advanced training on the special features of any particular system should produce Fortran programmers ready to write useful computer programs for that system.

GEORGE W. BROWN
Director
Western Data Processing Center
University of California, Los Angeles

Preface

Fortran is the name of a simple algebra-like language for programming computers. It is a contraction of "FORmula TRANslator" and is potentially valuable for solving any problem that can be expressed as a series of numerical relations or algebraic formulas.

Such problems are common in most professional and academic specialties. People working in these specialties are rapidly coming to depend on computers to alleviate the cost, the drudgery, and the often insurmountable hurdles of manual and mechanical calculations. The computer, however, has become not only a symbol of relief from tedium, but an indispensable tool for the day-to-day operation of large-scale data processing systems, for the conduct of research projects, and for the solutions of complex, practical problems. As this trend continues, it becomes essential for more and more people to acquaint themselves with computer programming.

Fortran is a relatively "old" computing language. The first version of it, for the IBM 704, was distributed in 1957. Since then it has been expanded and enriched and is currently available for programming many manufacturers' computers. The Fortran presented in this text is taken from IBM FORTRAN II. To avoid an unnecessary source of potential confusion, however, some of the nonessential, optional flexibility has been omitted. What remains is fairly universal and basic. Once this basic information has been learned and applied, the additional features of a particular Fortran system can be learned with relative ease as the need arises.

This book is an introduction, not only to Fortran, but also to the art of computer programming. It will be helpful to those who are planning to use the computer for the first time and to those who wish merely to acquaint themselves with the rudiments in Fortran programming. It should be useful to managers and administrators who need a basic acquaintance with the concepts and terminology of computer programming and to high school and college students in mathematics, statistics, engineering, and business administration.

The teaching technique used is called *autoinstructional.* It is based on B. F. Skinner's reinforcement theory of learning—the theory of programmed learning and teaching machines. Traditional applications of this theory require students to respond overtly to *every* new unit of information by writing down a word, numbers, or phrase, the correctness of which is *immediately* verified in the text. Experiments, particularly by Arnold Roe at UCLA, indicate, however, that under some circumstances, if the overt response and verification are omitted, the result is a program of instruction that is equally as effective and significantly more efficient. This mode of autoinstruction is usually called the *no-response* mode. It has been used extensively and, we believe, successfully in the present text.

The book also differs from traditional autoinstructional texts in the layout of the pages. Each page was individually designed by Elizabeth Paine, on the assumption that its unique appearance will aid those who remember information visually.

The no-response sections are supplemented by exercises and program-writing tasks. The exercises were added, not as instructional, but as a means of building confidence. The tasks are larger segments of behavior, and are analysed and corrected upon their completion, as part of the autoinstructional technique.

The tasks, the exercises, and the examples were all chosen for the generality of their content. This was also done so that students without particular background knowledge would not be aversively affected.

The program has never been experimental. It was designed as a practical replacement for eight hours of traditional instruction. Early drafts underwent informal trials, which resulted in extensive revisions and retrials. We hope the end product will add a modicum of efficiency to the job of learning Fortran.

We thank the following institutions and people: the Western Management Science Institute, Graduate School of Business Administration, University of California, Los Angeles, for its support, under a grant from the Ford Foundation, of the preparation and reproduction of the early drafts of the program; the director and staff of the Western Data Processing Center for their guidance and assistance: Mr. Arnold Roe of the UCLA engineering department's Teaching Systems Project for consultation; and the students who exposed themselves to the early drafts and provided feedback so essential for effective revision.

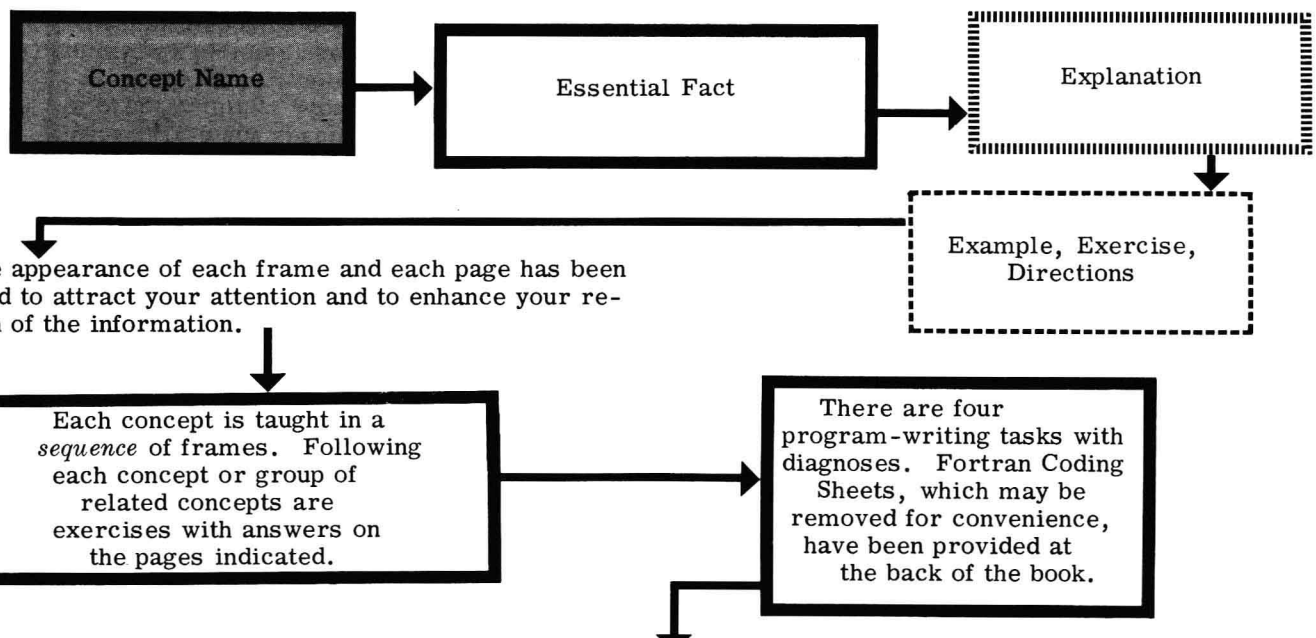
HARRY L. COLMAN

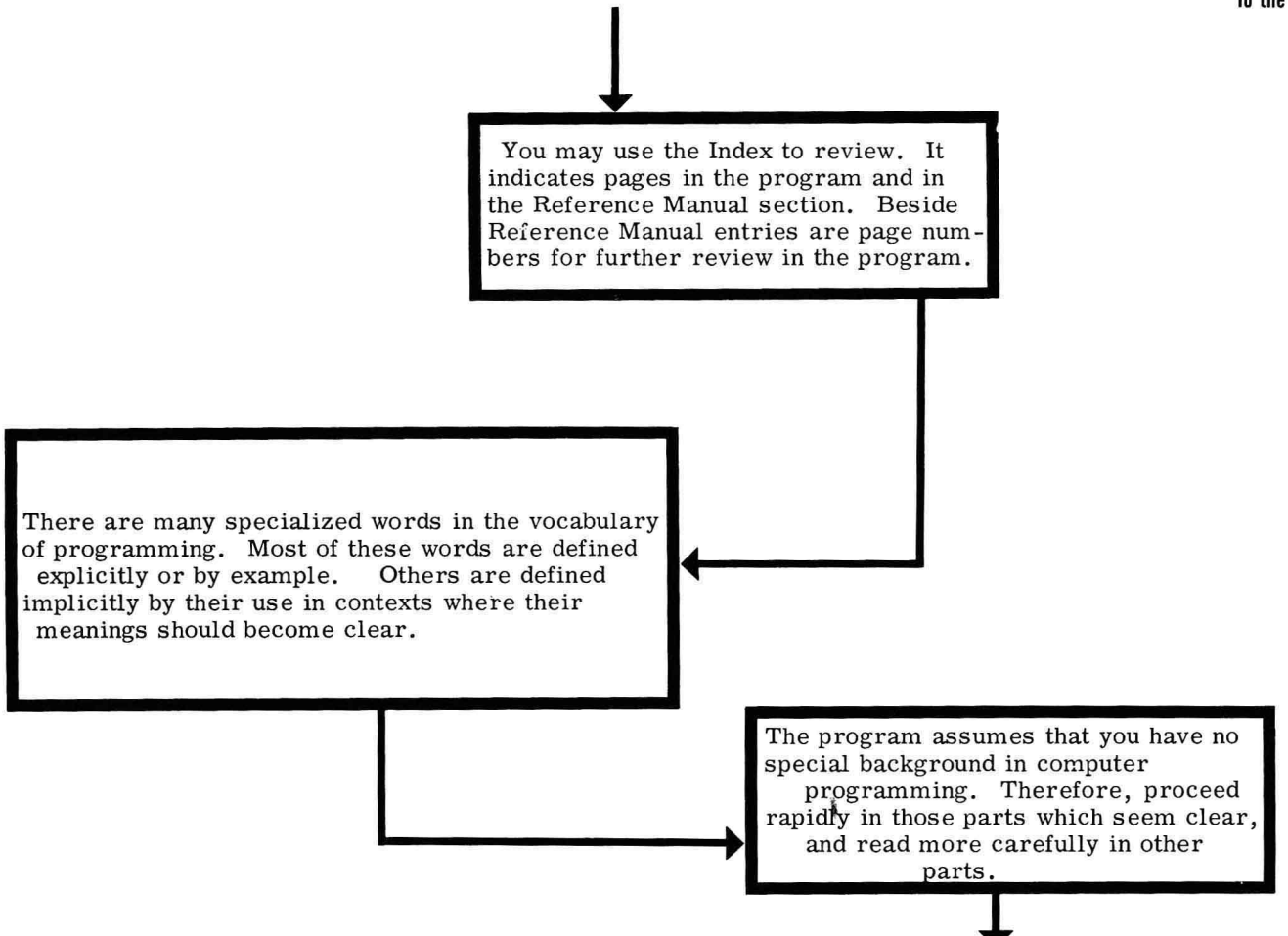
CLARENCE SMALLWOOD

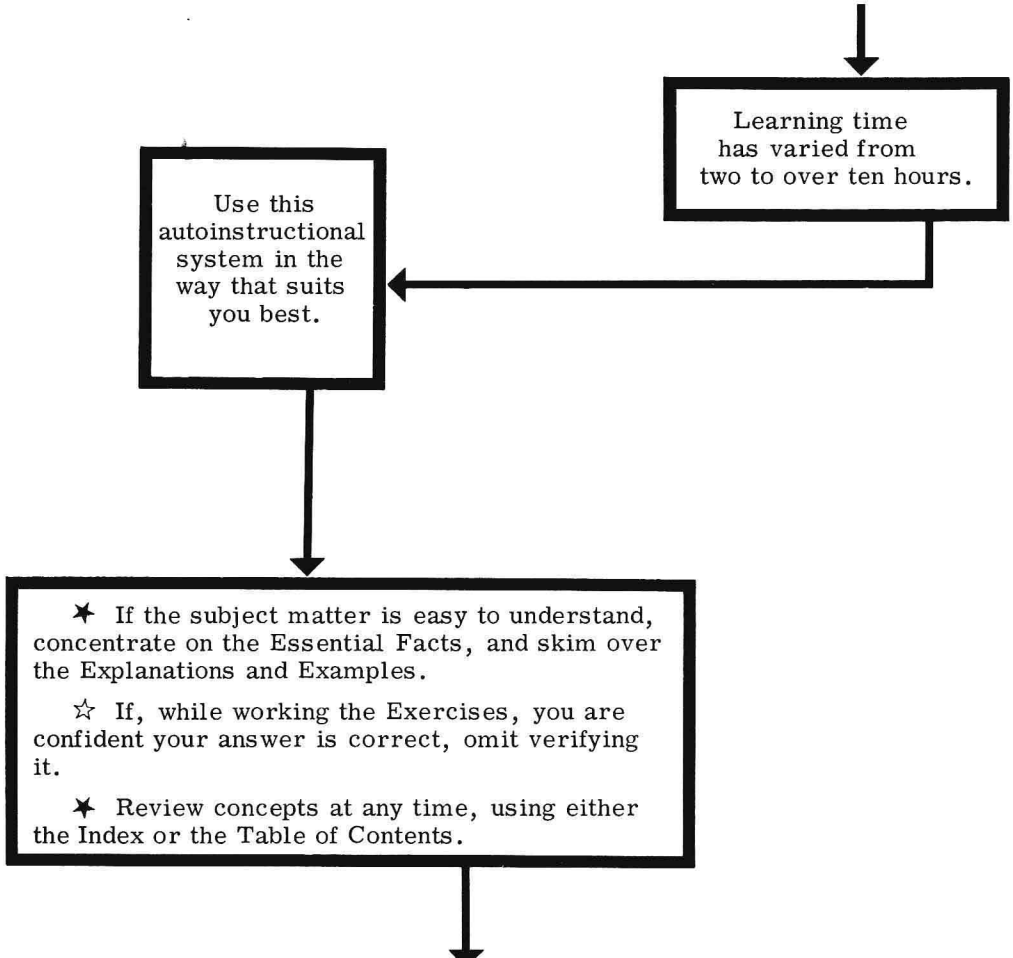
To the Reader

This text is an autoinstructional program. It has been written and designed to provide an efficient, self-contained system for learning the basic concepts of Fortran programming.

The subject matter is presented in short, concise increments printed inside a frame. The type of frame denotes the kind of information inside it.




You may use the Index to review. It indicates pages in the program and in the Reference Manual section. Beside Reference Manual entries are page numbers for further review in the program.
There are many specialized words in the vocabulary of programming. Most of these words are defined explicitly or by example. Others are defined implicitly by their use in contexts where their meanings should become clear.
The program assumes that you have no special background in computer programming. Therefore, proceed rapidly in those parts which seem clear, and read more carefully in other parts.


Learning time has varied from two to over ten hours.
Use this autoinstructional system in the way that suits you best.
✦ If the subject matter is easy to understand, concentrate on the Essential Facts, and skim over the Explanations and Examples.
☆ If, while working the Exercises, you are confident your answer is correct, omit verifying it.
✦ Review concepts at any time, using either the Index or the Table of Contents.

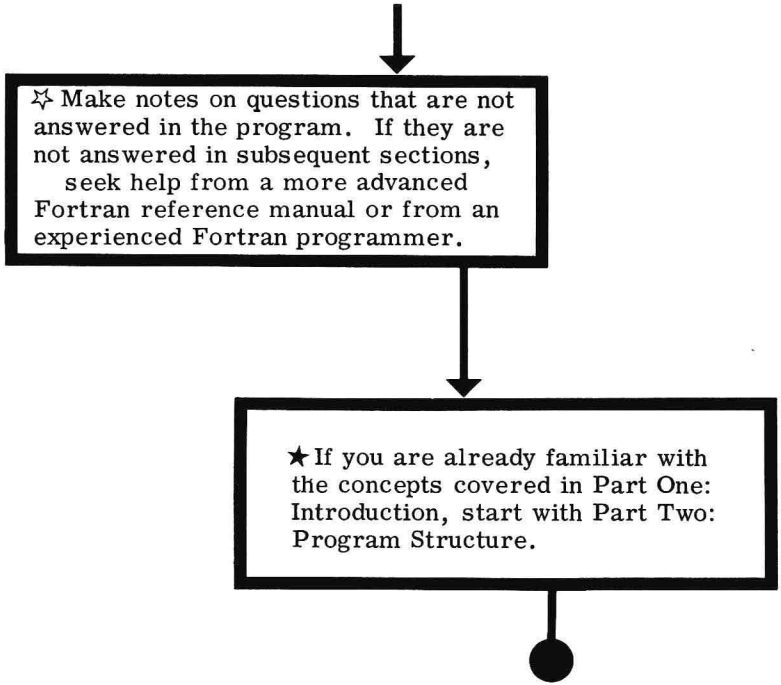


Table of Contents

Part One: INTRODUCTION

Computer Programs ● Computer Instructions Flow Diagrams ● Memory ● Object Programs and Source Programs ● Fortran Source Program Compilation ● Kinds of Fortran Statements Fortran Coding Sheets ● Punched Cards ● Exercises

Computer Programs

A program for a computer is like a set of instructions for a clerk who is running a desk calculator.

We can tell the calculator clerk *what* to do with some numbers without saying what the particular numbers are.

We can, for instance, instruct the clerk to add all the numbers on List A, write that total on a piece of blue paper, and label it Total of List A.

The clerk will know what to do without knowing the numbers on the list or the purpose for adding them.

In fact, the same instructions to the clerk will work for *any* list of numbers that we choose to call "List A."

On one day List A may contain one set of numbers, and on another day another set.

Computer Programs are like the set of instructions to the calculator clerk.

Notice that there are three fairly distinct operations involved:

Input - e. g., List A

Calculation - e. g., Accumulating the total

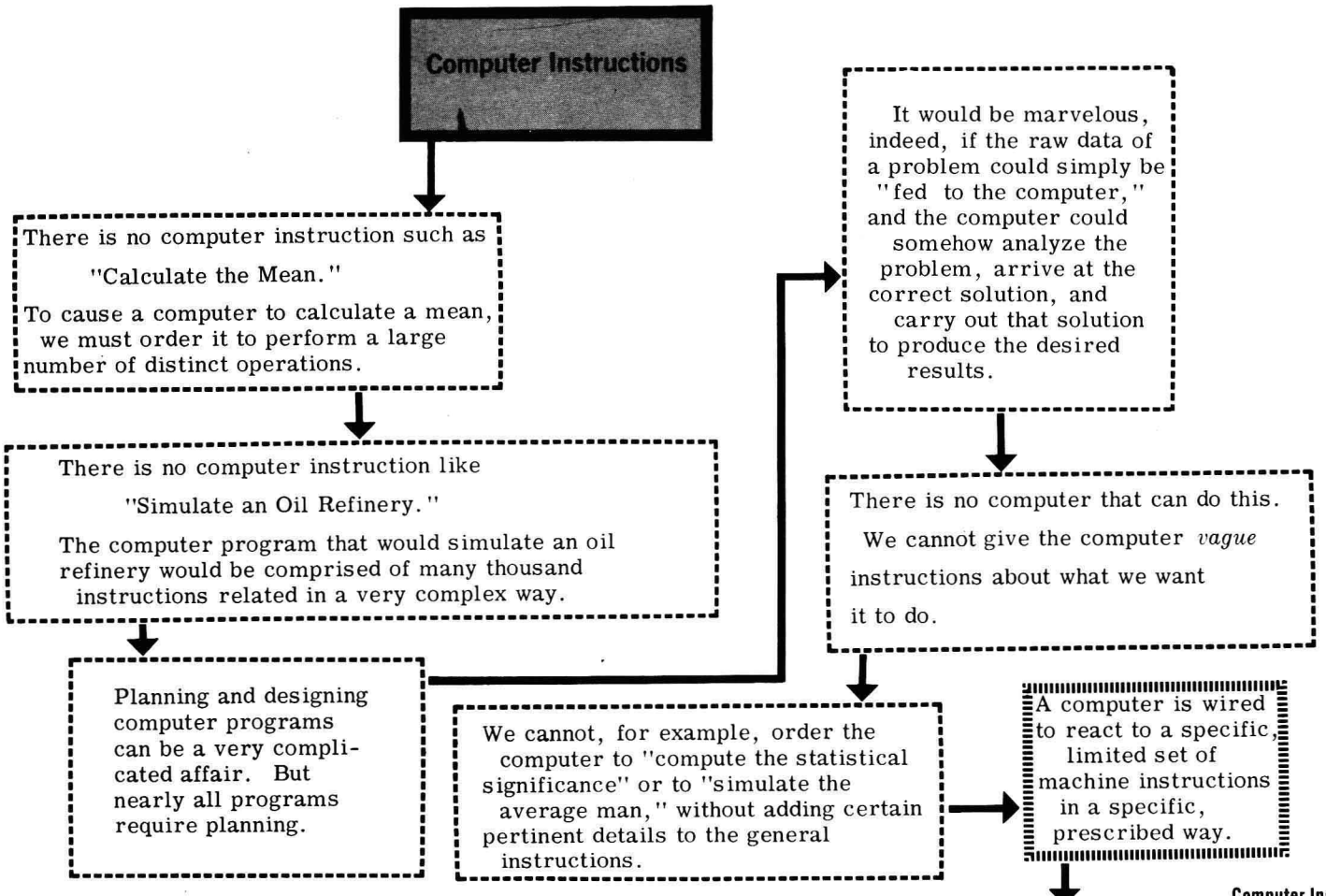
Output - e. g., Writing the total on a piece of blue paper.

All three phases are involved in most computer programs. The instructions that make up a computer program cause the computer to read the input data, perform some calculations, and print the results.

Naturally, the analogy between the calculator clerk and the computer has limits.

We can tell the clerk to add the numbers without saying how many there are, and without saying, "Now add the next one, now the next one, etc."

We must, however, instruct the computer first to add the first number, then to add the second number, then the third, etc.


Computer Instructions
There is no computer instruction such as
"Calculate the Mean."
To cause a computer to calculate a mean, we must order it to perform a large number of distinct operations.
There is no computer instruction like
"Simulate an Oil Refinery."
The computer program that would simulate an oil refinery would be comprised of many thousand instructions related in a very complex way.
Planning and designing computer programs can be a very complicated affair. But nearly all programs require planning.
It would be marvelous, indeed, if the raw data of a problem could simply be "fed to the computer," and the computer could somehow analyze the problem, arrive at the correct solution, and carry out that solution to produce the desired results.
There is no computer that can do this.
We cannot give the computer *vague* instructions about what we want it to do.
We cannot, for example, order the computer to "compute the statistical significance" or to "simulate the average man," without adding certain pertinent details to the general instructions.
A computer is wired to react to a specific, limited set of machine instructions in a specific, prescribed way.