ELEMENTS OF MATHEMATICS

JAMES W. ARMSTRONG

SECOND EDITION

James W. Armstrong

University of Illinois, Urbana

Second Edition

Macmillan Publishing Co., Inc.
New York
Collier Macmillan Publishers
London

Macmillan Publishing Co., Inc.
866 Third Avenue, New York, New York 10022

Collier Macmillan Canada, Ltd.

Library of Congress Cataloging in Publication Data

Armstrong, James W
Elements of mathematics.

Includes index.
1. Mathematics—1961-   I. Title.
QA39.A73   1976   510   74-24367
ISBN 0-02-303910-8

Printing: 1 2 3 4 5 6 7 8   Year: 6 7 8 9 0 1 2

Preface

In the coming chapters we shall discuss some aspects of eight or nine of the most important and well-known branches of undergraduate mathematics. Every effort has been made to bypass the usual computational aspects of the topics covered so that many chapters involve no computation at all. In those few instances where a minimum of computation is absolutely necessary, we have included sufficient preliminary work in techniques. The student is not required or asked to bring more into his classroom than a willingness to look at some old and some new ideas with an unprejudiced mind.

The chapters are organized so that the text may be used in a variety of courses, although the primary purpose of the book is to present topics for what is often called a "liberal arts mathematics" course. Much effort has gone into designing chapters and sections that are sufficiently independent so that an instructor can easily skip those sections and chapters dealing with material not pertinent to his course. These chapters and sections are designated in the text by a dagger. Most chapter interconnections (aside from the obvious ones) appear only in the exercises, and the teacher's manual contains a finely detailed identification of these interconnections. The manual also contains carefully formulated estimates of the number of 50-minute sessions that might be needed to cover each section. This material is designed to assist the instructor in molding the contents of the text into the form best suited for his class.

The teacher will find a number of starred exercises in the text. The star does *not* indicate an exercise of more than average difficulty. It indicates that the exercise deals with an idea introduced in a previous exercise or with an idea that has not been explicitly discussed in the body of the text. Starred exercises of the latter type may be used as topics for additional class discussion.

I wish to acknowledge the assistance of the late Professor Carl B. Allendoerfer and of Professor Ronald E. Walpole, who reviewed the statistics

chapter. Shirley Wilson and Debra Romack assisted most ably in the preparation of the answers. Mostly, however, I must acknowledge the assistance of my wife, Dianna, who has worked alongside me throughout this revision and whose insight has had a measurable bearing upon the improvements that appear here. In a very real sense she is a co-author.

J. W. A.

Contents

2

Mathematical Systems *43*

3

The System of Whole Numbers *71*

4

The Theory of Sets *99*

0

Flow Charting

The use of computers has produced a method of representing mathematical and other procedures pictorially by means of diagrams called **flow charts.** In this preliminary chapter we shall discuss flow charting. We shall use these charts in later chapters to augment and illustrate the discussions in the body of the text.

0.1 Communication Between Men and Machines

The fact that communication is possible between men and machines ought not to surprise us, for we all communicate with machines every day. We can, for example, talk with a radio. We can't say to the radio, "Turn yourself on and tune in on wavelength 99.5" and expect the radio to do it, because the radio can't understand our English language words. What we have to do is to translate these English language words into "words" that are comprehensible to the radio. We have to translate from the language of the user to the language of the machine.

The author has a radio so constructed by its makers that the instruction "Turn yourself on" is rendered into radio language by flipping a certain switch to the left. Flipping that switch is the translation into radio language of the English language instruction "Turn yourself on." The radio language equivalent of the

English language instruction "Turn to wavelength 99.5" is given to the radio by pushing a button marked "B." Thus, if one person stands in front of this radio shouting "Turn yourself on," and another person walks up to the radio and flips the switch, then we would say that the only difference between these two people lies in the fact that the first person doesn't understand how to give instructions to that radio—he does not understand radio talk—but that the second person does.

Men can talk with automobiles as well. We communicate an instruction to an automobile when we turn the ignition key. The instruction that we are communicating in this way is the instruction that is rendered in English as "Turn yourself on." The instruction that we give to an automobile when we put our foot on the brake is rendered in English by the word "Stop."

Both the radio and the automobile were designed by their builders to respond to certain selected kinds of instructions. Generally, in such machines as radios and cars, for each instruction that the machine is designed to recognize, there is a special device built into the machine that will communicate the human operator's desire to have that instruction carried out.

Computers are also constructed according to this general principle, but the difference is profound because computers are designed to follow a great many more instructions than radios and automobiles. Because its capabilities go far beyond those of the radio, a computer must be able to receive a great many more instructions; and this makes it much more difficult to communicate with a computer. It would not be feasible to install a separate switch for each and every instruction that was to be followed because computers must be able to respond to thousands of instructions.

A new way is needed, therefore, to communicate instructions to a computer. The method that has been developed is to construct the computer with its own special numerical language built in. This numerical language consists only of numbers and is so complicated that only the computer itself can possibly understand it. So the computer language is impossible for humans to use. On the other hand, English is impossible for the computer to use. What we need is someone to translate the English language instructions of the user into the numerical language of the computer.

The person who translates the English language instructions into a language computers can understand is called a **computer programmer.** Programmers take English language instructions and put them into special programmer language (FORTRAN, COBOL, or BASIC, among others), which is much nearer to the computer's own numerical language but still understandable to people with the requisite training. Moreover, their language is sufficiently close to the numerical language of the computer so that the computer itself is able to translate the programmer's language into the numerical computer language. Hence translating English instructions into computer language is a two-step process. In the first step the programmer translates the English instructions into the programmer's special language. The second step is performed by the computer itself when it translates the programmer's special language into its own numerical language.

The programmer's job is in fact a very difficult one. If the set of instructions is very long (and most often they are), the job of the programmer could take weeks of very hard labor. The first step is for the programmer to become familiar with the instructions the user has given him. Then he constructs a kind of first approximation to the programmer's language translation that he is seeking. He lays out his English language instructions in a precise way by means of what we call a **flow chart.** Flow charts enable the programmer to organize even a very complicated set of instructions into a logical display that shows fairly clearly what the computer will have to do in order to carry out the instructions. The programmer then works from the flow chart to construct the final product of his art, the programmer's language instructions. We shall discuss flow charting in the next section and look at some examples of mathematical instructions and nonmathematical instructions organized by means of flow charts.

Exercises 0.1

1. Make a list of some of the instructions that a human may communicate to each of the following types of machines:

   (a) A table lamp. (b) A pencil. (Careful!)
   (c) A typewriter. (d) Your left hand.
   (e) A telephone. (f) A desk stapler.
   (g) An elevator. (h) An automobile.

2. Can you think of some machines that are designed to be able to read words, numbers, or symbols in some way, even some very limited way? Make a list and describe briefly what they are designed to read and what they are not designed to read.
3. Object recognition is one of the new applications of computers. For example, computers can be linked to television cameras and taught to tell the difference between a circle and a square or between a square and a rectangle. However, this is not too spectacular a feat. We are all familiar with machines that have been "taught" to discriminate between certain very special types of objects. Make a list of such machines and the objects that they are designed to recognize. (*Suggestion:* A dollar bill changer.)

0.2 Constructing Flow Charts

In this section we learn to construct flow charts for mathematical and nonmathematical operations.

Let us consider the job of an assembly line worker whose job it is between

8 A.M. and noon and between 1 P.M. and 5 P.M. to insert one baseball after another into its own individual box, close up the boxes, and put them on a conveyor belt, whereupon they are carried away. (We don't care where.) The supervisor of such a worker might very well train new workers with the following simple instructions: "Look, you sit here in this chair. To your left will always be a crate of baseballs and to your right will always be a crate of boxes. You are to pick up a baseball and a box, put that baseball into that little box, close up the top of the box, and then put the box onto this conveyor belt that is passing along right here in front of you. You do this starting at 8 A.M. and stopping at 5 P.M. You get from noon to 1 P.M. for lunch. OK, get to work."

For most people these instructions would be perfectly satisfactory, for the job is not complicated. But let us consider these instructions carefully. Perhaps of most interest is the part of these instructions that involves the actual job of placing the balls into the boxes and closing up the tops and then placing these boxes on the conveyor belt. We could refer to placing a ball into a box and closing the top of the box as "packing a box" and most everyone would understand. We can describe this portion of the job pictorially as shown in Figure 0.1.

This flow chart describes the actual individual piece of work that the worker does all day long many times over. Next, let us add to this flow chart the information that the worker is to repeat this piece of work over and over again. We do this by introducing a **loop** into the flow chart, as shown in Figure 0.2.

The only thing wrong with this flow chart is that now we have the worker working forever. We have not inserted instructions that will tell the worker to stop under certain conditions. Let us correct this flow chart for the morning shift. See Figure 0.3.

What we have added to Figure 0.3 is a **decision step.** We now have a flow chart that instructs the worker to stop by noon but to keep packing until then. Our flow chart does not yet tell the worker when to begin work so that we need

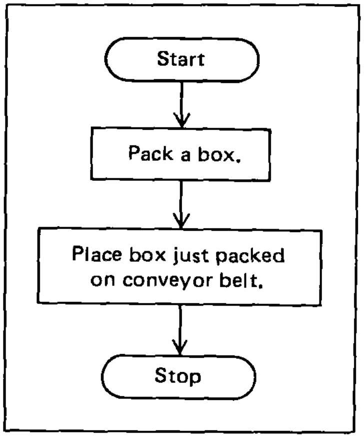

Figure 0.1

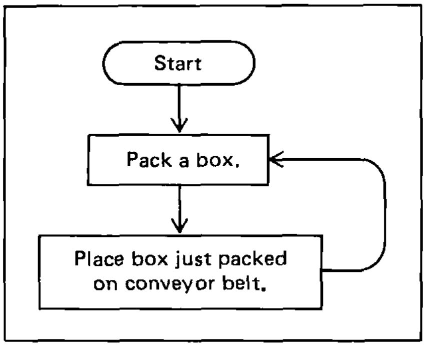

Figure 0.2

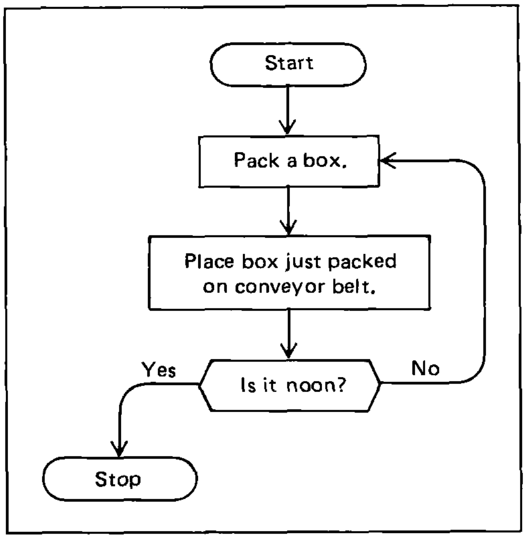

Figure 0.3

to add more instructions at the top of the chart. When should the worker start work? At 8 A.M., according to the supervisor. We must add another decision step, as shown in Figure 0.4, which will tell the worker whether or not it is time to begin work.

Notice that in order to tell the worker when to start in the morning, we have inserted a loop. If you have ever been around workers waiting for 8 A.M. before getting to work, you will have seen people doing just exactly what is described here. The loop that tells the worker when to stop for lunch might be called the "clock-watching loop."

Now, this chart is the morning chart. There is also an afternoon chart that is exactly the same except that 8 A.M. is replaced by 1 P.M. and noon is replaced

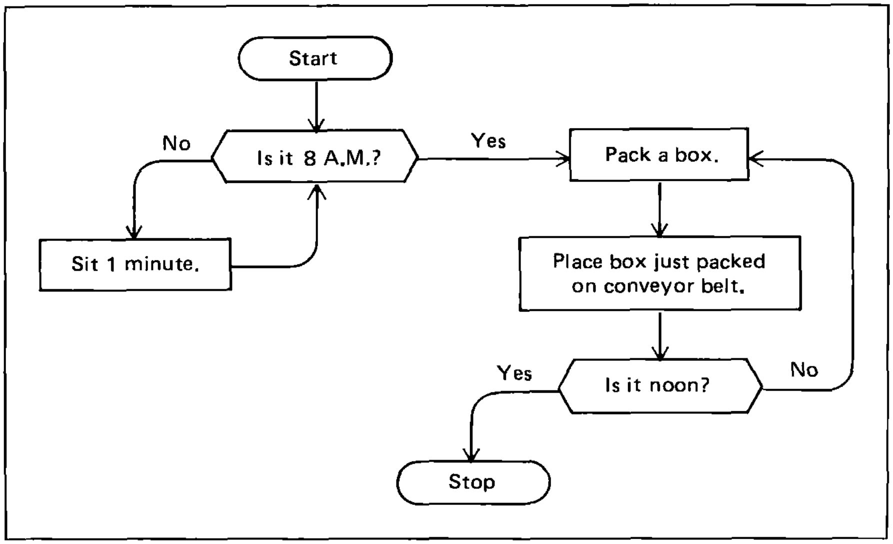

Figure 0.4

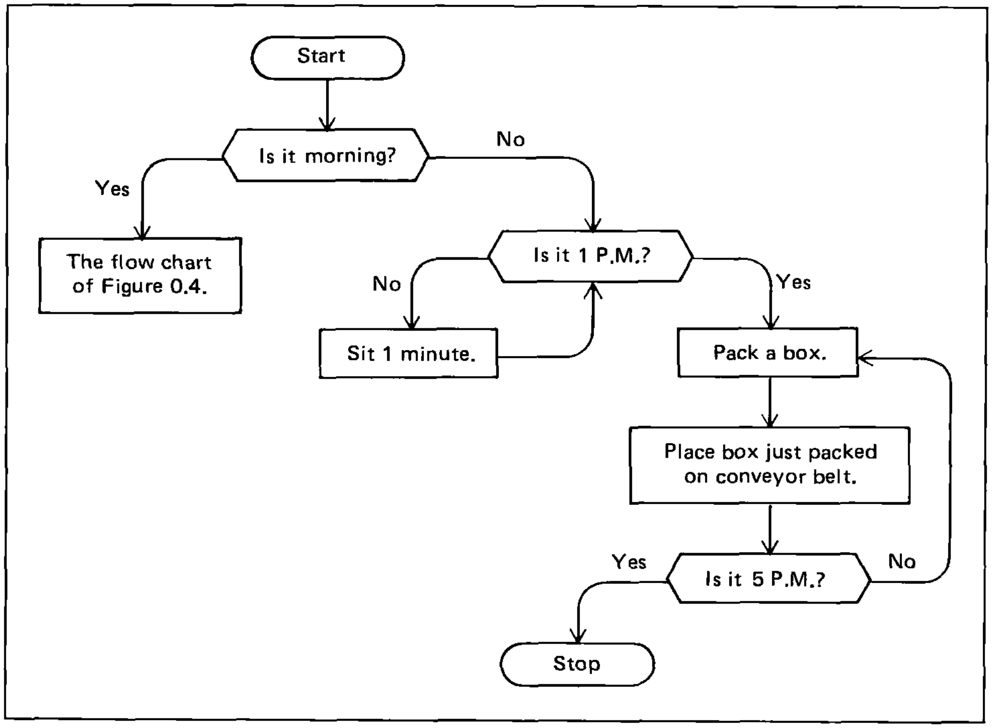

Figure 0.5

by 5 P.M. So, the total day's instructions consist of two *subcharts,* which are shown in Figure 0.5.

Looking at Figure 0.5, one could correctly argue that the chart is really incomplete. For example, the chart does not tell the worker precisely how to pack the box. Which hand should he use to pick up the baseball? Most boxes have three flaps to be closed; which should be closed first and which second? How exactly does one place a packed box on the conveyor belt? Do you heave it (after all these are baseballs) or put it down very gently? What happens if the worker who is supposed to supply you with crates of boxes gets behind and at some stage you don't have any more boxes? One could obviously go on forever (almost) with such questions and such legitimate criticisms of the flow chart we have constructed. In fact, if we were programming a computer to do this job, then we would have to answer all these questions, and more besides. But enough of this kind of talk. We are interested in programming human beings to do things and to understand things. We are not now in the business of instructing computers. We can say to a human being "Put the ball into the box and close up the box," and the human being will have a clear and sufficiently precise understanding of what needs to be done to do the job. The computer wouldn't, but that is not our problem—it is the computer programmer's problem.

The remainder of this chapter consists of a number of examples of the flow charting of familiar mathematical operations.

Example 1. A very simple mathematical operation is that of finding the average of two whole numbers. The operation is performed by summing the two numbers and then dividing this sum by 2. The resulting quotient is the average of the two numbers. See Figure 0.6. The instructions for finding the average are easily flow charted because there are no loops involved and there are no decisions to be made.

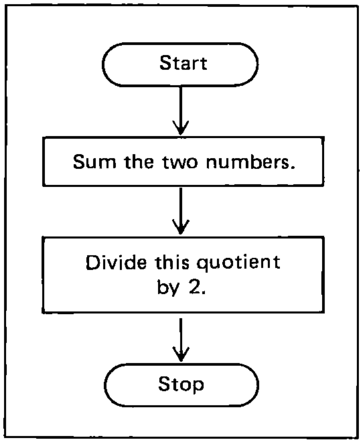

Figure 0.6

Example 2. The determining of whether or not a number x is even. See Figure 0.7.

Example 3. The single, most difficult computational operation that is taught in elementary school is the long division process. This process is introduced as early as the second or third grade, but students in the ninth and tenth grades often have difficulty in working even fairly routine problems. The process that you use to find quotients and remainders in long division problems is the one that is hardest to describe. This is the process that starts out, for example,

$$17\overline{)370}$$

and begins by your estimating the number of 17's contained in 37. To write out a flow chart for this algorithm is rather difficult; but after you have more experience with flow charting, you may be interested in trying your hand at it.

The easier process for finding long division quotients and remainders (the one that is easier to explain, although seldom the easiest to use in practice) involves repeated subtraction. That is, to long divide 370 by 17 you may simply begin subtracting 17 from 370 and keep subtracting 17's until you finally obtain a difference that is less than 17. The total number of times you will have subtracted 17 is the quotient to the long division, and the difference that you ended up with when you stopped is the long division remainder.

Figure 0.8 contains a flow chart that displays this process in a reasonably careful way. Note that because the process will, in general, require repeated subtractions, it will have to contain a loop. Because the repeated subtractions do not repeat forever, there must be a decision at the end of the loop that will tell us whether to go into another loop or whether to stop looping and terminate the chart. The

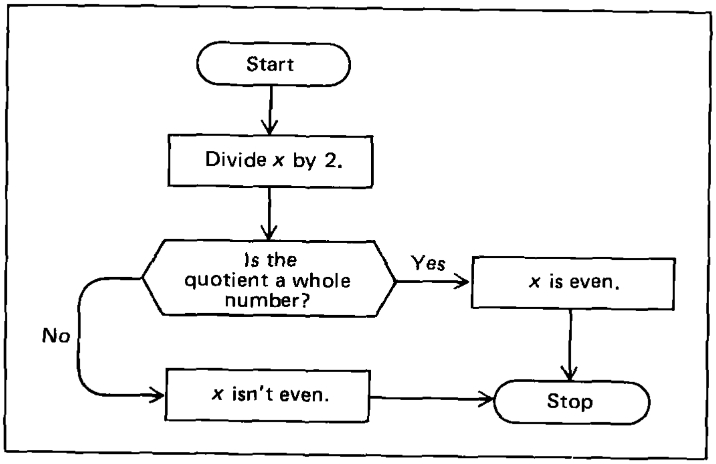

Figure 0.7