3000

INTRODUCTION TO ELECTRONIC COMPUTERS

CHAPTER 3 - FLOWCHARTING

F. J. Gruenberger
and
D. D. McCracken

October 1961

DISCLAIMER NOTICE

THIS DOCUMENT IS BEST QUALITY PRACTICABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.

INTRODUCTION TO ELECTRONIC COMPUTERS

CHAPTER 3 - FLOWCHARTING

F. J. Gruenberger*
The RAND Corporation, Santa Monica, California

D. D. McCracken*
Consultant to The RAND Corporation, Santa Monica, California

This paper consists of a preliminary version of Chapter 3 of a textbook, Introduction to Electronic Computers.

This material is still subject to revision. It is issued at this time for three reasons:

(1) So that it may be criticized by many interested people,

(2) so that it may be further debugged in actual classroom use, and

(3) so that manufacturers who wish to prepare a paraphrase (involving a computer other than the IBM 1620) may begin work.

---

*Any views expressed in this paper are those of the author. They should not be interpreted as reflecting the views of The RAND Corporation or the official opinion or policy of any of its governmental or private research sponsors. Papers are reproduced by The RAND Corporation as a courtesy to members of its staff.

INTRODUCTION TO ELECTRONIC COMPUTERS

CHAPTER 3 - FLOWCHARTING

3.1 What a Flowchart Is

In order to explain how we can direct a computer in the solution of problems, we will first need a new type of language. People who deal with computers express their thinking about a problem pictorially, using a flowchart (or flow diagram, or block diagram).

3.2 The Parts of a Flowchart

A flowchart is made up of symbols, each of which has a specific meaning. The symbols are connected by straight lines. The whole chart is generally read from left to right and top to bottom, much as a printed page. Arrows are frequently attached to the lines to make clear the direction of flow. There are several sets of conventions currently in use on the meaning of symbols in flowcharts. For our purposes, only a few symbols will be needed.

1. A rectangle is used to indicate that some specific action is to be taken. There is only one entrance and one exit for a rectangle, which is called a function box. The

information written in a function box may be a statement in English, or an algebraic statement, or some kind of meaningful shorthand. The commonest example of the latter is the arrow symbol which is used to indicate the operation "replace." Examples of function boxes are given in Figs. 1, 2, and 3.

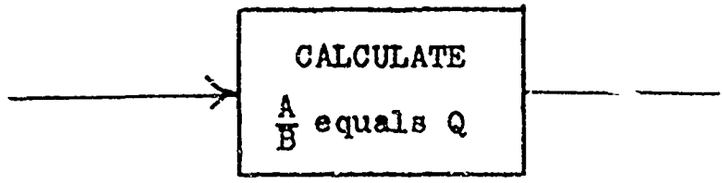


Figure 1--A function box indicating a calculation.

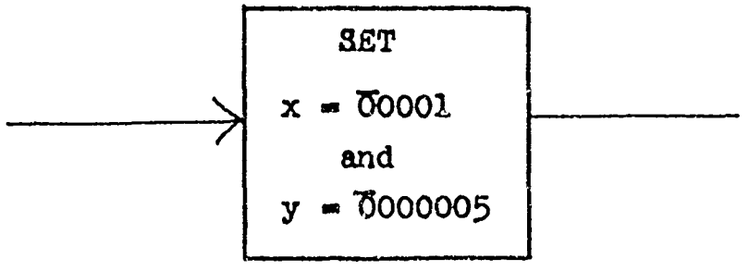


Figure 2--A function box indicating initialization.

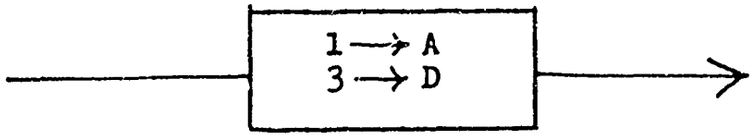


Figure 3--A function box to reset two values.

In Fig. 3, the number 1 is to replace the value of the field we have called A; the number 3 is to replace the value of the field we have called D.

It is important to realize that nothing we say about flowchart notation involves rigid rules. The chief use of a flowchart is as a tool to help organize <u>your</u> thinking on a problem.

If you prefer to write "field A is to be replaced by the number 001" instead of the compact notation of Fig. 3, you may do so. Merely keep in mind that others may have to read your flowcharts (your instructor, for example, when you get into trouble with a problem), and it is well not to invent new notation too freely.

2. A diamond is used to indicate that a decision of some sort is to be made. A diamond, which is called a decision box, usually has one entrance and two exits; it may have three or more exits. Figs. 4, 5, and 6 illustrate some typical decision boxes. In Fig. 5 there are three exits (from top to bottom, they are the logical paths to follow if field B is greater than 3600, is equal to 3600, or is less than 3600). Since such comparisons are quite common, a shorthand way of expressing them lies in the use of the colon. Fig. 6 expresses the same comparison as in Fig. 5. In addition to compactness, this gives us more flexibility. The two decision boxes of Fig. 7 are logically equivalent, for instance.

3. Finally, we use small circles on a flowchart to guide us around without having to draw so many lines and arrows. Their use is evident in Figure 8. Starting at reference circle 1, we proceed to the right. After one function box we arrive at a circled 2. This says "go to reference 2" and we look (still observing the normal convention that we read from left to right, top to bottom) for a similar circle, to be found at the left end of the next row of symbols. Similarly, at the right end of that row, we again find directions to proceed to reference 2.

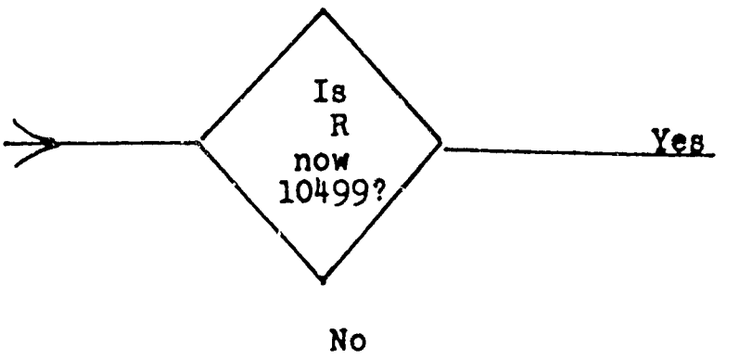


Figure 4--A decision box showing a logical test.

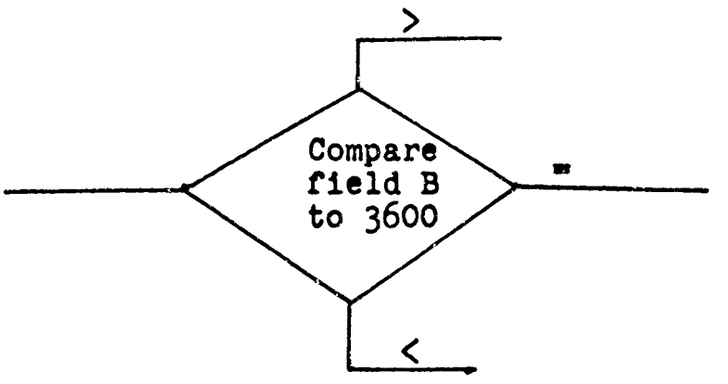


Figure 5--A decision box for a 3-way branch based on a comparison of two values.

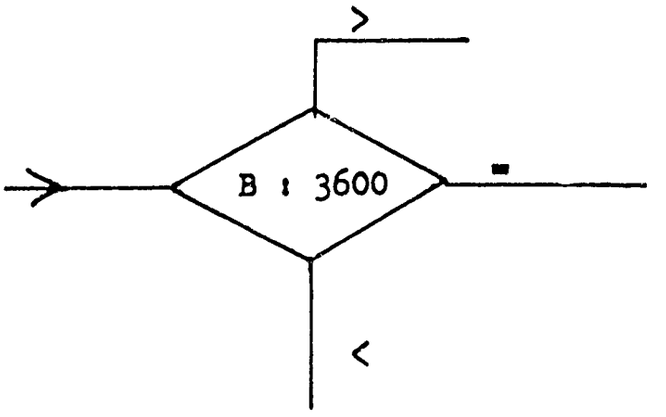


Figure 6--Another way of expressing the logic of Figure 5.

These various conventions allow us to follow the logical paths of a flowchart with ease.

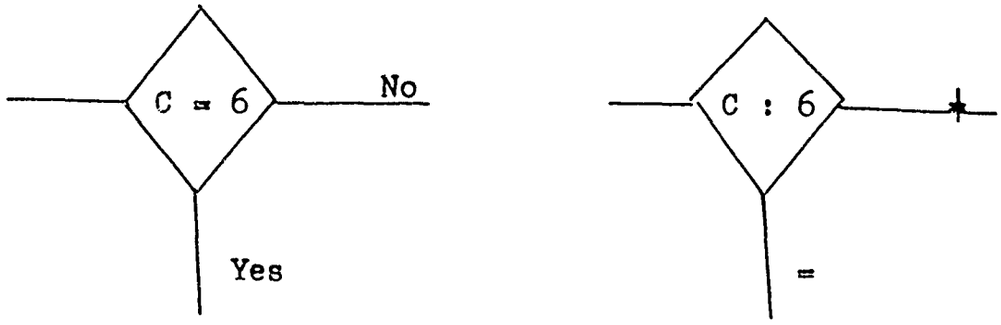


Figure 7--The use of the colon in flowcharting.

3.3 An Example of a Flowchart

With this much background on the symbolism to be used in this book, we may proceed to an example.

Let's consider a problem we considered briefly in Chapter 1, to see how a flowchart helps us lay out the logic of a computational procedure. The problem was to compute

$$Q = \frac{A + 2B}{C + 3D}$$

for all values of A, B, C, and D, each independently varying from 1 to 5 in steps of 1.

We begin by deciding on the general plan of attack. Here, it is clear that we need to form all combinations of the values of A, B, C, and D and compute the value of Q for each such combination. Since each of the four variables ranges independently over the values 1, 2, 3, 4, and 5, there are 5 x 5 x 5 x 5 = 625 cases.

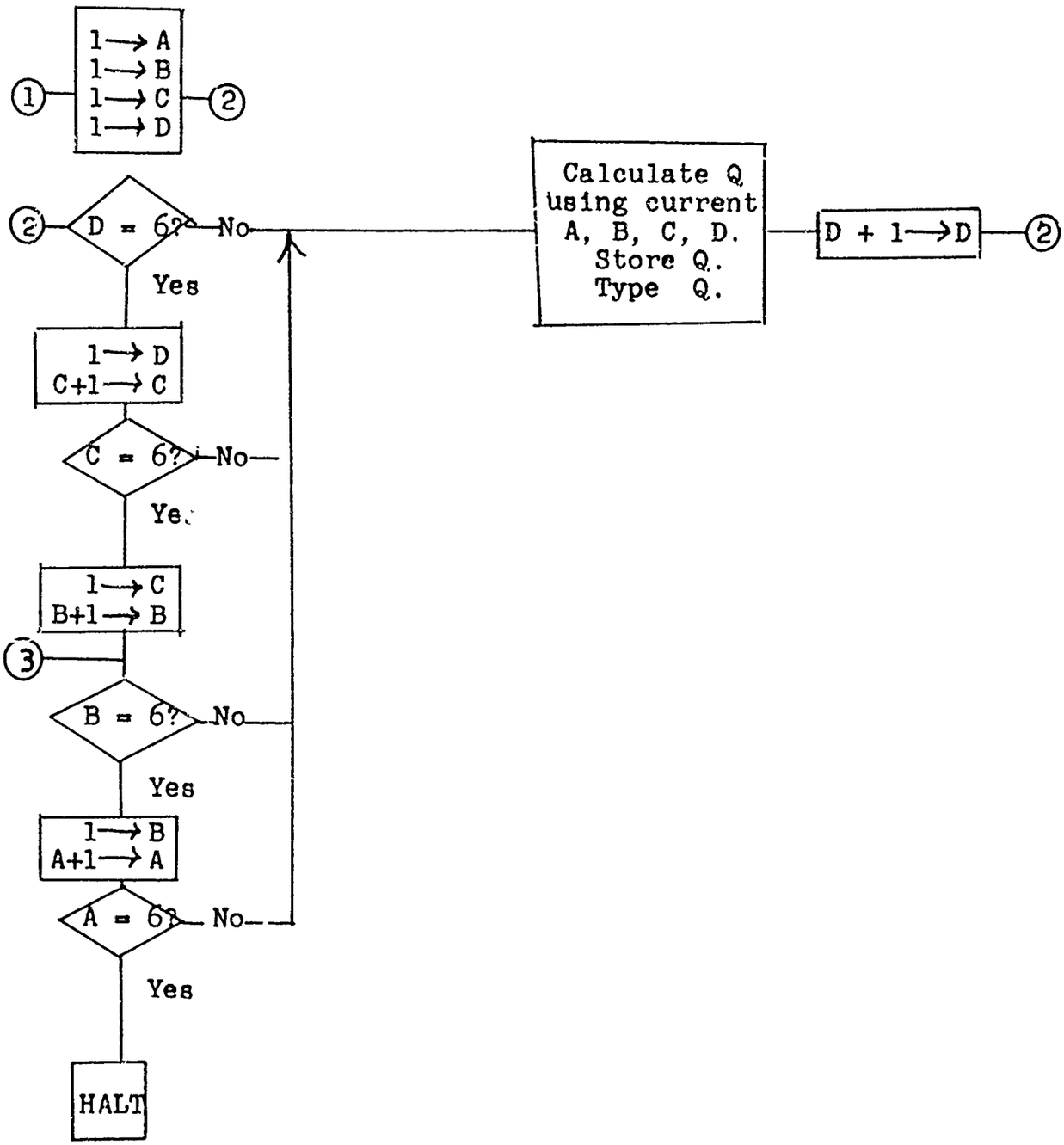


Figure 8--A flowchart example.

So far we have carried out the analysis of the problem (which is almost trivial in this case). In many problems it is a major part of the task. The next step, which should be kept distinct from the analysis, is the construction of the flowchart. On the flowchart we want a bird's eye view of the logic of the procedure to be used in solving the problem. We don't want any of the details that are special to our particular computer. For instance, when we later come to code the procedure, it will be important to know precisely where the values of the variables are stored--but while drawing the flowchart we have no interest in the matter. Thus, we see that it is important to keep the flowchart separate both from the analysis (which precedes it) and the coding (which follows it).

The flowchart for this problem is shown in Fig. 8. We begin (at reference 1) by initializing the cells (wherever they may turn out to be) containing A, B, C, and D to one. We next ask the question, "Does the D cell at this time contain a six?" Of course it doesn't! It will later, however, since we will return to reference 2 many times.

The most difficult part of this problem to code will undoubtedly be the calculate step in which Q itself is produced. But that's not the problem at the moment.

After calculating Q and typing it, we add 1 to D and return to reference 2. The value of D is still not equal to 6, so we go through the calculation steps again (this time with A, B, and C equal to 1, and D equal to 2). We go through this "loop" five times; after the fifth time, D is made equal to 6, and the decision

box following reference 2 will be answered "yes." D is then set back to 1; C is increased by 1, and we proceed to the calculation again. The value of D advances through the values 2, 3, 4, 5 once more. If you will trace through the flowchart carefully, you will see that Q is indeed calculated for all 625 combinations of values of A, B, C, and D.

What we have done is to define a precise procedure for obtaining all the 625 combinations. This sort of thing must be mapped out before getting into the details of the coding. The process of coding (which we do from the flowchart) has its own difficulties, and we don't want to have to worry about two complex and error-prone processes at the same time.

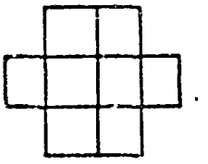
You may have noticed that the flowchart of Fig. 8 contained a reference (3) that never seemed to be used; that is, we don't seem to "go to reference 3" from anywhere. This is deliberate. We will write instructions with the flowchart as our guide. The instructions on the coding sheets should be tied to the flowchart as much as possible. Having extra reference symbols (lots of them) on the flowchart will aid in labelling the instructions.

3.4 The Purpose of the Flowchart

As we have seen, the main purpose of a flowchart is to isolate the logic of a problem from the later details of writing the instructions to implement that logic. There is another purpose: The flowchart allows us to break up a problem into parts as small as we please. In fact, experience indicates that it is wise to

flowchart a problem in several levels. The first level gives a broad view of the overall problem; succeeding levels expand all or part of the broader view.

Let's illustrate all this with a puzzle problem. Here is an array of eight squares:



The object is to place the digits from 1 to 8 in the eight squares in such a way that no two adjacent squares (horizontally, vertically, or diagonally) contain consecutive numbers. You might wish to try solving this puzzle by hand; most people find the solution within 15 minutes.

Suppose we were to try to solve this problem with a computer. We shall take a brute-force approach and capitalize on the speed of the machine; in short, simply try every possible arrangement and check each one to see if one fits. This is inefficient, of course; if the arrangement

```
   3  5
1  6  7  2
   8  4
```

is unacceptable, we would not normally consider

```
   8  4
1  6  7  2
   3  5
```

But since a computer can sift through all the possibilities rather fast, and since the brute-force approach might well be the easiest for us to code, we shall try it. A broad view of the problem would look like Fig. 9.

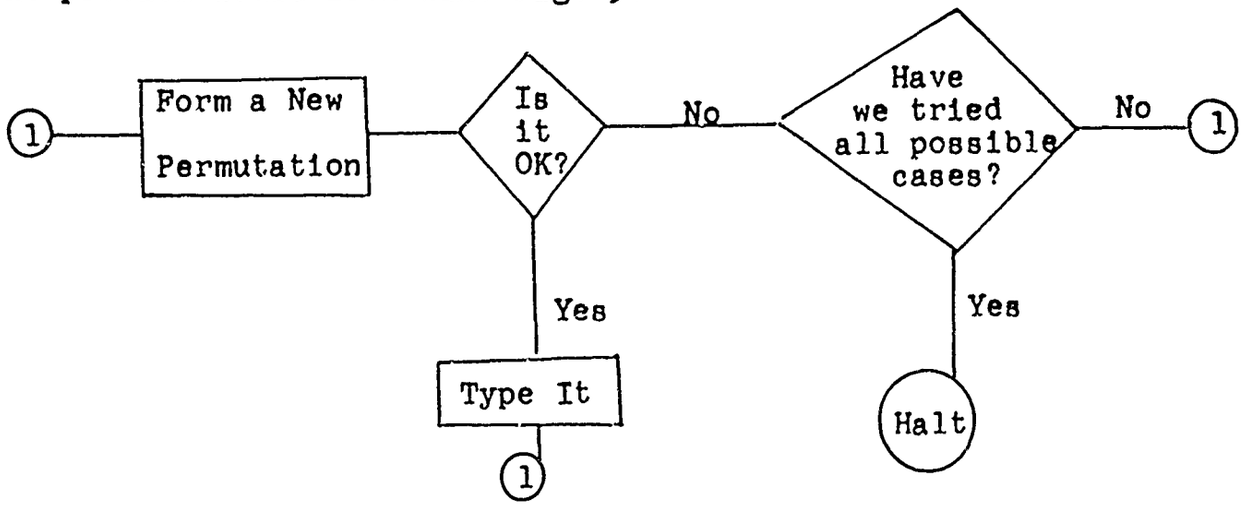


Figure 9--The eight squares problem.

Each of the rectangles and diamonds on this broad flowchart can now be expanded into a full flowchart of its own. Even such a simple task as "type it" may need considerable throught (since it will turn out to take some 30 instructions to perform the task and those instructions must be planned, too).

We are not quite ready to attack this problem in detail. At the moment, the important point is that we have already started to organize our attack on this problem. Each of the logical boxes on this flowchart may need considerable work (indeed, the box labelled "form a new permutation" will require a very large flowchart of its own). We shall refer to this problem again

in Chapter 18.

3.5 Another Example

Let's look at an example which we can pursue further. Consider example 4 on page 1 (of Chapter 1).

> A man cashes a check at a bank. The teller accidentally interchanges the amount for dollars and the amount for cents. After spending ¢14.19 the man finds that he still has twice as much left as the amount of the check. What was the amount of the check?

We will seek the answer by a brute-force approach--simply by running systematically through every possible value for the check to see which one (if any) fits the conditions of the problem. We let x equal the amount for dollars and y equal the amount for cents. The simplest approach would be to run through all the values 00.00, 00.01, 00.02, 00.03,..., 99.99.

A little reflection about the problem, though, will show us some shortcuts which will speed up the search enormously. First, y must be greater than x in order that the teller pay out more than the check called for. Hence, when y exceeds 99, x is to be increased by one (reference 8), and y can then be reset to the value of x (reference 3).

Similarly, x can never be greater than 49. (Suppose x were greater than 49. For example, the original check might be for ¢50.99. The man gets ¢99.50. After spending some of this--the amount of the parameter--he still has twice as much as the original amount. But half of ¢99.50 is an amount less than ¢50.)

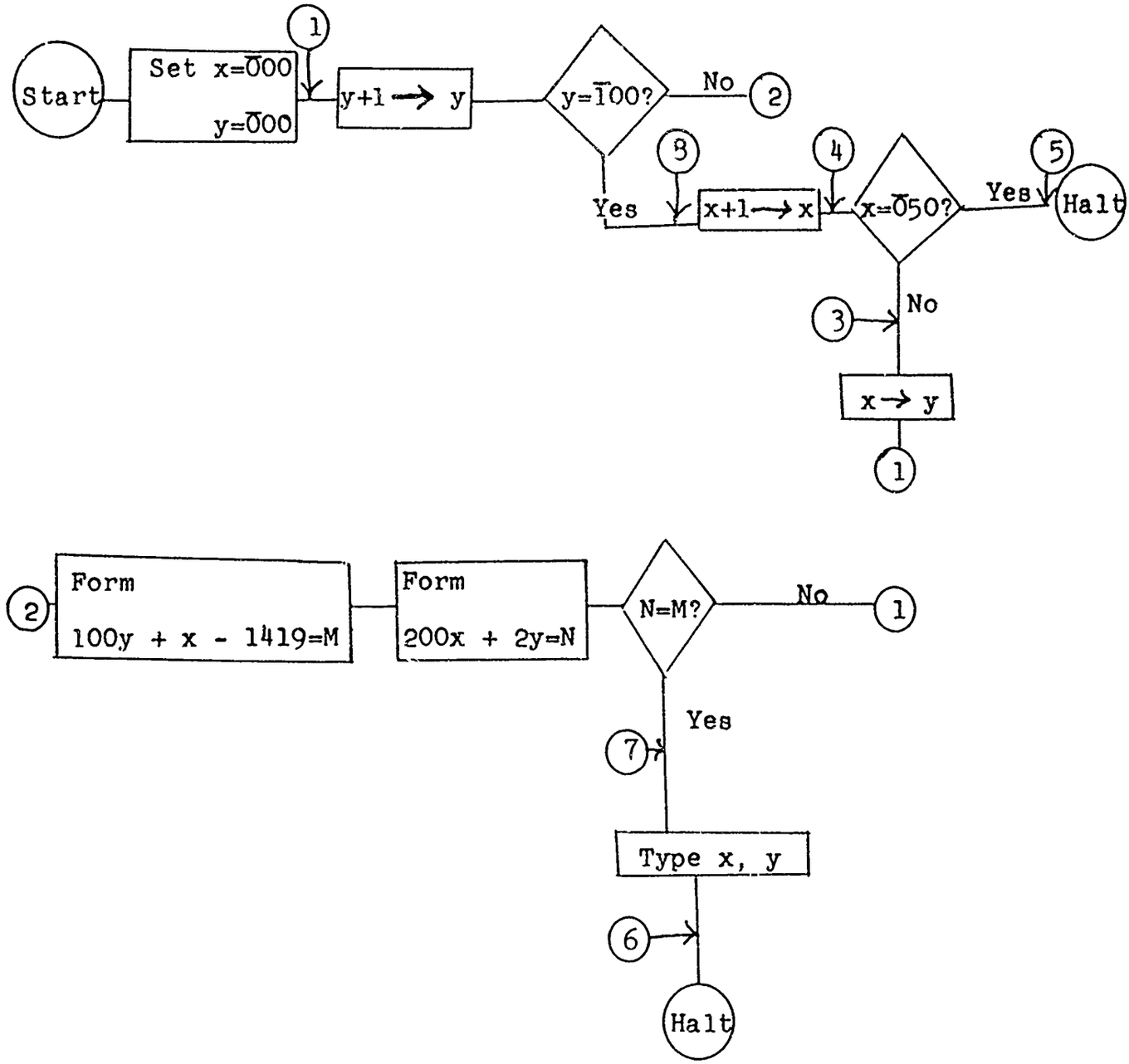


Figure 10--The bank check problem.

The test, then, for complete exhaustion of the possibilities, is at reference 4, where we inquire whether x is now 50.

You are probably not ready to code a problem as complex as this without a great deal of practice on simpler problems. It is shown here as an exercise in flowcharting; that is, for practice in following the logic of a problem.

3.6. Another Use for Flowcharts

We have indicated two main reasons for drawing flowcharts. First, a flowchart lays out pictorially the logic of the problem solution; ~~Second,~~ the flowchart ~~then helps us~~ breaks up the solution into small pieces, each of which is easy to work with.

There is a third ~~good reason~~ for making a flowchart for every problem. A flowchart is an excellent means of communication. To you as a beginner, it provides a means of communicating your thinking about a problem to your instructor, as well as to yourself. Seldom is a problem on a computer completely disposed of in one day. Whatever you do, you will probably come back to it repeatedly. You will find that good flowcharts will help tremendously in recalling or explaining what you thought you were doing yesterday.

The flowcharts you see in this book are all quite neat, having been drawn with lettering tools and templates. Most workable flowcharts, in practice, are much cruder looking. The goal here is to organize your thinking on paper.

3.7 Summary

We have shown the bare minumum number of symbols to enable

us to use flowcharts in subsequent chapters of this book. More notation will be introduced as it is needed. We have seen in this chapter three good reasons for using flowcharts. As we proceed, you will see a fourth reason: to allow us to present complex material to you in simple pictorial form.

EXERCISES

1. List the symbols used in flowcharting and their meanings.

2. Three numbers, called A, B, and C, are in storage. We are at a stage in a problem where, in order to go on, it is necessary that all three of these numbers be positive. Assuming that we can write computer instructions to do what we wish, draw a flowchart of the logic involved in testing the three numbers. If any of the three numbers is negative, we wish to halt; if all three are positive, we wish to proceed.

3. Three numbers (at addresses A, B, and C) in storage are known to be positive and all different. We wish to move the largest of the three to location D. Flowchart the logic of this situation. Assume, in this exercise and in the next two, that all the numbers involved are of the form $\overline{x}xxx$, and that the numbers involved are all different.

4. Three numbers (at addresses A, B, and C) are in storage. We wish to move them around so that the algebraically smallest winds up at address A; the largest at address C, and the middle-sized number at address B. Draw a flowchart.

5. Four numbers (at addresses A, B, C, and D) in storage

must each be 1000 or less in order that a particular problem can continue. If any of the numbers is less than or equal to 1000, it is to be left alone; if greater than 1000, it is to be replaced by the number 1000. Draw a flowchart.

6. How would the flowchart of exercise 5 change if the problem concerned twenty numbers instead of four? Would you simply extend your flowchart to include twenty decision boxes? See if you can find a way to use <u>one</u> decision box twenty times.

7. Given a number, N, in storage, we wish to calculate a number, P_1, as follows:

$$\frac{N - 1}{N} \cdot \frac{N - 2}{N} = P_1$$

and type P_1. Then we go on to calculate and type a series of values:

$$P_1 \cdot \frac{N - 3}{N} = P_2$$

$$P_2 \cdot \frac{N - 4}{N} = P_3$$

$$P_3 \cdot \frac{N - 5}{N} = P_4$$

and so on.

The first value to be typed is exceptional. We must arrange, in the housekeeping of the problem, to calculate and type P_1 as a separate situation. Having done that, all subsequent values can be calculated using a pattern of instruction which is repeated over and over. Draw a flowchart for this problem.