1

The Birth of the Computer

Ever since the invention of numbers, humanity has tried to make instruments to help in performing calculations. There were tablets for calculating earlier than 3000 B.C., and the well-known Chinese bead frame existed well before the birth of Christ; but tablets, bead frames, and abaci are all completely nonautomatic. The idea of mechanizing calculation is very old, although it was not a practical possibility until mechanical engineering (whose finest applications had been in the design and making of clocks) became sufficiently highly developed. In the early nineteenth century the British astronomer and mathematician Charles Babbage described what could have been a machine with the ability to perform any calculation whatever; but unfortunately the mechanical-engineering technology of the time provided neither the reliability nor the speed that were necessary for the realization of his dream. The construction of the first calculators had to await the arrival of electromechanical technology, and it was the development of electronics that led to the first computers.

1 Problem Solving and Its Mechanization

If it is to carry out a calculation, a machine must know the route it has to follow. But a machine has no intrinsic knowledge and knows nothing about the external world; therefore it has to be given instructions on how to proceed in the minutest detail. The details of the method of solution of a problem that can be performed by a machine are in the form of a statement of a sequence of operations that the machine has to carry out. We call this the *process*. This way of describing a process is in fact common practice, and we use it whenever we follow what we may call an *algorithmic procedure*.

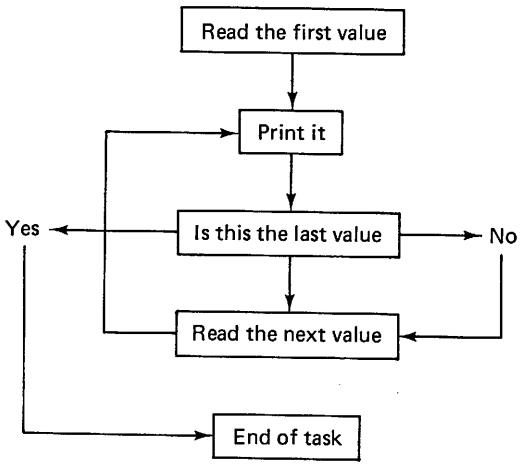

Figure 1.1
Reading a table of values.

Algorithmic Procedure

Among the problems continually presenting themselves for solution are a very large class for which the process of solution can be given in advance in terms of a finite number of exact statements of operations to be used in order to produce a specific result. Figure 1.1 illustrates this type of procedure by showing the steps that must be followed in order to read a table of values. We say that a procedure for solving a problem is *algorithmic* when it can be expressed as a sequence of statements of operations to be performed and when no knowledge or intelligence is required beyond what is strictly necessary in order to perform these operations. The statements must therefore be sufficiently clear and precise as to present no difficulties of interpretation. The diagrammatic representation of an algorithmic procedure is called a *flowchart.*

The statements in an algorithmic procedure are of two types. The first type is illustrated in Figure 1.1 by "Read the first value" or "Go from the part of the flowchart where the operation required is to print the value read to that where the operation is to check whether the last value has been read." These are what we might call imperative commands and we describe such statements as *unconditional.* The second type can be put in this general form: "*If* such-and-such a result is observed, *then* do this *else* do that." Such a statement in figure 1.1 is "*If* the value in the table that has been read is the last value, *then* the task is ended *else* read the next value." These are called *conditional* or *logical* statements.[1]

Thanks to the conditional statement there is the possibility, at any stage in a procedure, of choosing between different routes to follow from there on depending upon the conditions at that stage: for example, whether or not to read another value from the table. A procedure with no conditional statements permits of no variations at all and therefore can be used only to solve a single problem: for example, "Read a value from the table." The inclusion of conditional statements, on the contrary, allows a very much broader class of problems to be solved by making it possible to vary the procedure according to the requirements of the individual problems.

Algorithms

From now on we shall be concerned only with problems for which the process of solution, expressed as an algorithmic procedure, can be given to a machine and followed step by step by the machine until the result is obtained. To put it another way, we shall be studying only those types of algorithmic procedures corresponding to classes of problems that can be attacked with the help of a machine. We shall speak of *operations* to be performed rather than of statements, and of *algorithms* rather than of algorithmic procedures.

In general, an algorithm is constructed for the solution of a problem when it has been found, by reasoning or by experience or as a result of teaching, that a process for the solution can be described as a sequence of operations that can be performed without any need to attach meaning to them. We learn such step-by-step processes for solving certain problems from our earliest school days: for example, the addition or multiplication of numbers, finding square roots, and finding the volume of a sphere. These methods are all algorithms; they all lead from the use of a finite number of operations required to solve a particular problem (for example, the addition of 21 and 33) to their use in solving a broad class of problems (for example, the addition of *any* two numbers).

Algorithms have a long history. Those for addition, multiplication, and division were produced in prehistoric times. The Babylonians are a case in point; they devised algorithms for the solution of decidedly complex arithmetical problems posed by their studies of the movements of the stars and the planets. The word itself, however, comes from the name of the Persian mathematician Abu Ja'far Mohammed Ibn Musa Al Khowarismi, whose writings on arithmetic circa 825 A.D. influenced for centuries the development of mathematics.

Algorithmic Processing

As I have said, carrying out the sequence of operations represented by the statements forming an algorithm requires nothing more than what is necessary to understand these operations; therefore the process is essentially mechanical. Thus if a machine can be built for this special purpose, it will perform better than any human because it will follow the sequence precisely without any variation and will not tire. It is not surprising therefore that the idea of mechanizing algorithms is very old. (Rosenberg [1969], who gives attempts to mechanize algorithms before the year 1000 A.D., holds that the most typical were those made by Gerbert d'Aurillac, who became Pope Sylvester II in 999.)

What characteristics must a machine have in order to carry out such a sequence of operations, which we shall call *algorithmic processing*? At the outset, it must be able to deal with the two types of statement—unconditional and conditional respectively—defined earlier. For unconditional statements the machine must be able to perform the specific operations required by the algorithms; for example, it must have a device that can add two numbers. Such operations or transformations fall into two groups. In one are the operations of transferring information from one part of the machine to another. In the other are the operations having a single *operand*, called *unary operations* (such as the operation of changing the sign of a quantity, the quantity concerned being the operand), those having two operands, called *binary operations* (such as that of adding together two numbers, those numbers being the operands), and, further, operations having more than two operands, although these are much less common.[a] For conditional statements the machine must be able at least to perform simple tests, such as to decide whether one number is greater than another. Such operations, whether unary or binary, are called *logical operations.*

Operations of this type—transfer, unary, binary, logical—are called the *primitive operations* of the process. The elementary arithmetical and logical operations are of very great importance in any algorithmic process, as is easily seen from a consideration of a few examples. To answer the question "What is the sum of the first 10 numbers?," one needs only to make 10 additions—arithmetical operations—testing after each one to find whether that was the tenth. It is less obvious that a yes-or-no type of question can be answered by similar sequences of operations. Consider, for example, the question whether the word "greatness" occurs in a particular piece of writing of General de Gaulle's. To answer this, each word of the text must be given a code number, the same number for each occurrence of the same word and a different number for each occurrence of a different word; the text is thus

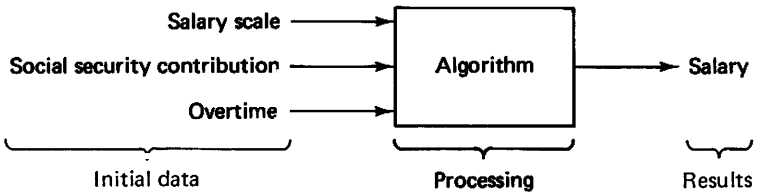

Figure 1.2
Algorithmic processing: calculating salaries.

represented by a sequence of numbers. The machine starts with the first number and subtracts from it the number representing the word "greatness"; if the result is zero, the two numbers are the same and "greatness" has been found as the first word of the text. If not, the machine moves on to the next number in the sequence and repeats the operation, and so on. If it arrives at the end of the text without recording a zero result, then the answer to the original question is no.

A machine that can carry out algorithmic processing must therefore have a number of basic operations. It must be able to perform the fundamental operations of arithmetic, and for this it must have what is called an *arithmetical unit*; similarly, it must have a *logical unit* for the logical operations. The two are often combined in one unit, called the *arithmetical-logical unit (ALU)*. Finally, the actual performance of the operations as a sequence in time must be monitored and directed, and for this there is a special unit, the control unit.

The effect of any algorithm is to transform the set of elements supplied to it at the start into the set forming the results. For the calculation of a salary, for example, the starting, or initial, set can include salary scale, hours of overtime worked, social security contribution, and so on; Figure 1.2 illustrates this.

The quantities involved in an algorithmic process can be either *numerical*—numbers, financial accounts, statistical observations—or nonnumerical—letters, phrases, pieces of text, and also, for example, music; music as a sequence of sounds can be coded by a function of the amplitude of the sound, measured at regular intervals.[2] Thus algorithmic processing can be applied to an extremely wide range of types of elements. In fact, the fields of application of algorithms are so numerous that one can say that an algorithm can be provided for any problem for which one knows a method of solution. In what follows the word *data* will be used to describe any elements that can be coded and thus subjected to algorithmic processing.

The alphabet used most frequently for coding data inside the machine is the *binary* because this is the easiest to represent in physical equip-

ment. Hence the term *bit*, contraction of *binary digit*,[3] which can take either of only two values, represented by 0 and 1, respectively. Other forms of coding are of course possible. It seems that binary coding was not used to any great extent until after the construction in 1939 of the first Bell Laboratories machine, with which we deal shortly.

The starting values must somehow be given to the machine if it is to perform any processing on them, so there must be some device for communication between the machine and the outside world that enables this to be done; this is called the *input unit*. Similarly, the machine must be provided with some means for displaying its results, and therefore there must be an *output unit*. The two will, to a large extent, be made from the same components and are therefore usually referred to as the *input/output (I/O) units*.

We must note here that the machine carries out its processing quite independently of any meaning that the data may have; in effect, it works inside a formal world and knows nothing of any correspondence that may have been set up between data and the real world outside. Thus for the machine the word "greatness" in the de Gaulle text conveys no particular meaning. As Arsac has said [1970], "This separation of form and content characterizes all algorithmic processing." When we speak of information processing we must not forget that the word "information" is being used in a very special sense.

To sum up, an item of data has these characteristics:

1. It is coded with an alphabet of finite size.
2. So far as the machine is concerned, it carries no meaning.

Louis de Broglie wrote, "A calculating machine can provide its users with the results of its calculations, that is, with information." When it is simply a question of calculating the value of a function or the salary of an employee, or reserving a seat in a railway train or searching for a particular item in a file, or any similar task, the processing is performed with a specific end in view: given the data, to arrive at the result. For the user, who can attach whatever semantic or other significance to this result, what he gets is indeed information. The word *informatique* was coined by the French engineer Philippe Dreyfus to define the scientific activity that uses algorithms to process data in order to obtain information. The nearest English equivalent, *informatics*, has not come into general use, *data processing* being more frequent.[b]

I shall use the term machine, or information-processing machine, for any machine that can perform algorithmic processing. Such a machine can be characterized by the pair of attributes, *competence-performance*. Competence refers to the algorithm corresponding to the