Chapter 1

THE ELEMENTS OF PROGRAMMING

1.1. Introduction

The complex calculating machines which have become increasingly well known during the last ten years may be classified as either 'analogue computers' or 'digital computers'.

In an analogue computer, numbers are represented by continuously variable quantities such as lengths, angles or voltages. The electrical or mechanical connexions of the computer are set up in such a way that these quantities satisfy the same mathematical relationships as govern the variables in the problem to be solved. The behaviour of the computer is thus made directly comparable to the behaviour of the physical or mathematical system under consideration.

In a digital computer, on the other hand, numbers are represented by sequences of digits. Although many modern machines do not in fact work in the decimal scale, it is often convenient to think of the numbers which they handle as being composed of decimal digits, stored in registers basically similar to the counting wheels of a desk calculating machine. Since a digital computer can only store rational numbers, the mathematical operations which it can perform are merely those of ordinary arithmetic.

This book is concerned only with digital computers, and for the sake of brevity they will generally be referred to merely as 'computers'. Although these machines are sometimes called 'electronic brains', their electronic nature is in one sense purely incidental. Their importance lies far more in the fact that they are automatic, being able to carry out lengthy sequences of numerical operations without human intervention. Since even the most complex calculations can always be broken down into a series of simple arithmetical steps, digital computers can be applied to many problems unsuitable for analogue solution.

The technique of making a computer carry out a particular calculation is known as 'programming'. This involves first breaking the calculation down into a sequence of arithmetical operations, and then preparing a series of 'instructions' which cause the computer to carry out the required operations in the correct order. Each instruction

specifies a particular step such as multiplication or addition, and it is the programmer's job to see that the instructions are couched in a language or 'code' which the machine can interpret. A whole set of coded instructions, arranged in the correct order, is known as a 'programme'. As there are almost as many instruction codes as there are machines, a programme made up for one computer may not be intelligible to another. It may, however, be used repeatedly on the machine for which it was designed, carrying out any number of similar calculations without the need for further programming.

A simple example of a computer with a permanently built-in programme is a desk calculating machine equipped with automatic division. Once the appropriate button has been pressed a series of subtractions, sign tests and shifts occur without further human intervention. Such an instrument, however, is like a musical box which will only play a single tune. The type of computer described in this book is more like a pianola, which can play any piece of music provided that it is given a set of 'coded instructions' in the form of a suitably punched roll of paper.

It would be out of place in a book of this kind to give a detailed account of the organization and instruction code of a real computer. It is difficult, however, to give an idea of what programming involves without taking particular examples. The remainder of this chapter will be devoted, therefore, to the programming of some numerical examples for an idealized computer with a very simple instruction code.

1.2. A simple calculation and its mechanization

While there are several different ways of solving sets of simultaneous linear algebraic equations, one of the simplest methods is that of successive approximation. Although this technique is only suitable for certain classes of equations, it is very easy to programme for a computer, and is chosen here for that reason.

The equations

$$\left.\begin{aligned} 10x+6y-z&=19,\\ 3x+9y+2z&=27,\\ x-4y+8z&=17,\end{aligned}\right\} \tag{1.1}$$

for instance, may be written in the form

$$x=(19-6y+z)/10, \tag{1.2a}$$

$$y=(27-3x-2z)/9, \tag{1.2b}$$

$$z=(17-x+4y)/8. \tag{1.2c}$$

If an initial 'trial solution' is chosen, such as $x=y=z=1$, equation (1.2a) gives a 'corrected' value for x,

$$x=(19-6+1)/10=1{\cdot}4,$$

which may be used in equation (1.2b) to give a new value of y,

$$y=(27-3\times 1{\cdot}4-2)/9=2{\cdot}311.$$

The 'corrected' values of x and y may then be substituted in the third equation (1.2c) to give

$$z=(17-1{\cdot}4+4\times 2{\cdot}311)/8=3{\cdot}106,$$

and the whole process repeated with the new values of x, y and z. It is obvious that if the successive values of the variables converge at all, they will converge to the correct solution of the original equations. In fact, successive iterations give

	Initial values	1st iteration	2nd	3rd	4th
x	1	1·400	0·824	0·983	1·002
y	1	2·311	2·035	1·997	1·999
z	1	3·106	3·040	3·002	2·999

The results in the last column are in very close agreement with the exact solution, which is $x=1$, $y=2$, $z=3$.

In the case of a general set of three equations

$$\left.\begin{aligned} a_1x+b_1y+c_1z&=d_1,\\ a_2x+b_2y+c_2z&=d_2,\\ a_3x+b_3y+c_3z&=d_3,\end{aligned}\right\} \tag{1.3}$$

the equations for the nth iteration are

$$\left.\begin{aligned} x_{n+1}&=(d_1-b_1y_n-c_1z_n)/a_1,\\ y_{n+1}&=(d_2-a_2x_{n+1}-c_2z_n)/b_2,\\ z_{n+1}&=(d_3-a_3x_{n+1}-b_3y_{n+1})/c_3.\end{aligned}\right\} \tag{1.4}$$

Alternatively, the process can be thought of as the calculation of a series of successive corrections to the original values of x, y and z. Thus equations (1.4) may be written

$$\left.\begin{aligned} x_{n+1}-x_n&=(d_1-a_1x_n-b_1y_n-c_1z_n)/a_1,\\ y_{n+1}-y_n&=(d_2-a_2x_{n+1}-b_2y_n-c_2z_n)/b_2,\\ z_{n+1}-z_n&=(d_3-a_3x_{n+1}-b_3y_{n+1}-c_3z_n)/c_3,\end{aligned}\right\} \tag{1.5}$$

and it will be noted that the numerators on the right-hand sides of equations (1.5) are merely the differences between the right- and left-hand sides of the original equations (1.3).

In order to carry out this calculation for a given set of coefficients, using equations (1.5), a human being would need a pencil and paper for writing down numbers, some kind of mental or mechanical apparatus for doing arithmetic, and a written or memorized scheme of analysis to follow. This scheme might be

(*a*) Form $(d_1 - a_1 x_n - b_1 y_n - c_1 z_n)/a_1$.
(*b*) Add this to x_n to obtain x_{n+1}.
(*c*) Form $(d_2 - a_2 x_{n+1} - b_2 y_n - c_2 z_n)/b_2$.
(*d*) Add this to y_n to obtain y_{n+1}.
(*e*) Form $(d_3 - a_3 x_{n+1} - b_3 y_{n+1} - c_3 z_n)/c_3$.
(*f*) Add this to z_n to obtain z_{n+1}.
(*g*) Go back to (*a*), testing for convergence on each occasion after the first.

It is apparent that a human being given such a 'programme' of instructions and a set of initial values x_0, y_0, z_0 could carry out the calculation purely mechanically, even if he had never heard of simultaneous equations.

In any pencil-and-paper calculation of this sort, the human computer is involved in three kinds of activity, namely,

(1) Transferring numbers between his paper and his arithmetic machine,

(2) Actually doing arithmetic,

(3) Looking at his list of instructions to see what to do next.

Of these three, it is probable that in many practical cases (2) consumes least time, even when carried out with the simplest type of desk calculating machine. Therefore, even if a high-speed machine which could add or multiply in a millisecond were available its use would not greatly decrease the total time taken by a calculation. A significant saving requires a reduction in the time spent on activities (1) and (3) as well, and this can only be achieved if the whole process is mechanized.

Now consider the features of a machine which could replace the human computer entirely. It would need the following elements:

(i) A 'store', corresponding to the human computer's sheet of paper, in which numbers could be held.

(ii) An 'arithmetic unit' capable of addition, multiplication, etc., together with facilities for transferring numbers between it and the store.

(iii) Means for feeding numbers into the store and displaying results.

(iv) A unit which would control the machine according to a list of 'instructions' supplied by a human operator.

Such a machine would have little advantage over a human being if only a single calculation had to be carried out, as it would probably take as long to prepare the instructions as to do the calculation by hand. However, if the instructions were made independent of the actual numbers appearing in the machine, they could be used repeatedly for carrying out similar calculations with different sets of data, the speed depending only on the rate at which the computer took in and obeyed the instructions.

These ideas form the basis of all automatic digital computers, and were first put forward by Charles Babbage, long before the days of electronics. Babbage envisaged an 'analytical engine' which was to be controlled by holes punched in cards, just as Jacquard had used punched cards to control the weaving of cloth. The specification of this purely mechanical computer contained most of the logical features found in present-day electronic machines, but although parts of it were built it was never completed.

1.3. A simple automatic computer

Although the 'programme' of instructions for solving a set of three simultaneous equations which was developed in the previous section may appear quite simple to a human being, it could only be fed into a machine which understood both algebra and the English language. To make an automatic computer a practical possibility it is necessary to devise a much simpler form for the instructions. To fix ideas, a simple computer using a numerical instruction code will now be described. Although this computer is an imaginary one, it could in fact be built without great difficulty.

The store of the machine consists of a hundred registers which for convenience are numbered 1–100. The number by which a particular register is known is called its 'address', while the decimal number stored in it is referred to as its 'content'. Thus the content of the register whose address is 59 might be the decimal number 3·14159.

The computer is equipped with an arithmetic unit similar to a desk calculating machine, which can carry out addition, subtraction, multiplication and division. This unit contains two special registers known as the multiplier register and the accumulator. The computer is able to read decimal numbers into its store from punched paper tape, and to print numbers on a typewriter. It has two tape-readers,

one for numerical data and one for instructions. The arrangement is shown diagrammatically in fig. 1.

When operating, the machine reads an instruction from the appropriate tape, obeys it, and then proceeds to the next instruction on the tape. The instructions consist of pairs of numbers, each pair consisting of a 'function number' f and an address s. The function

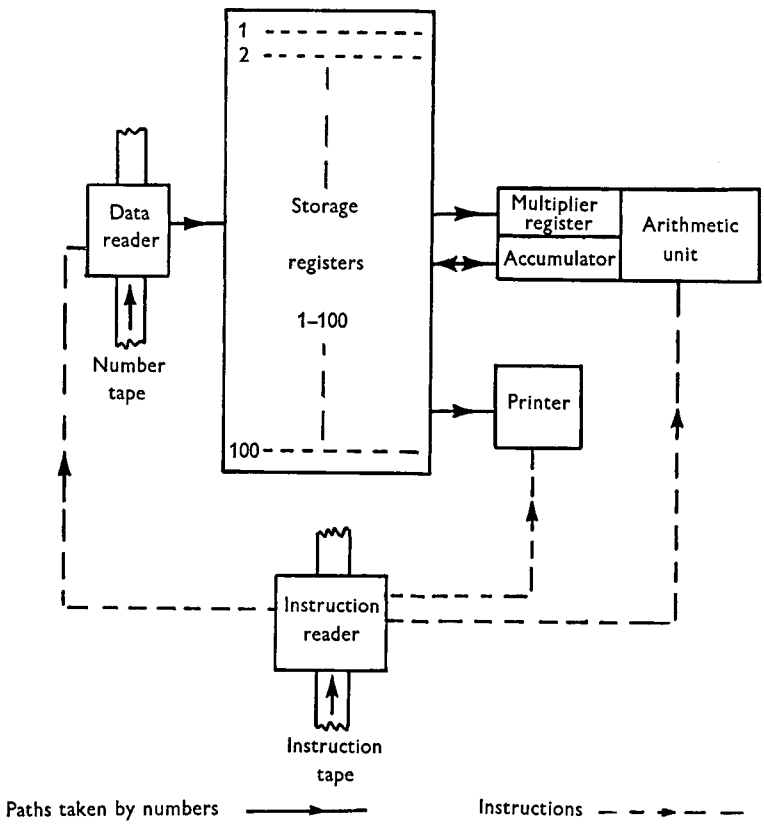

Fig. 1. Arrangement of a simple automatic computer.

number specifies the type of operation to be carried out, while the address specifies the register whose content is to be operated on. The functions described in the following table will be sufficient for a considerable proportion of numerical work. The symbol S indicates the content of the register s referred to by the address section of the instruction, M and A stand for the numbers stored in the multiplier register and the accumulator respectively, while a dash indicates the state of a register after an instruction has been obeyed. The contents

of registers not specifically mentioned remain unchanged, while the number previously stored in a register is obliterated when a new number is transferred to it.

Function number	Effect
1	Read one number into storage register s from the number tape. ($S' =$ tape entry; number tape moves on.)
2	Print the content of register s.
3	Stop. (The address part of this instruction is irrelevant.)
4	Copy the content of register s into the accumulator. ($A' = S$.)
5	Add to the content of the accumulator. ($A' = A + S$.)
6	Subtract from the content of the accumulator. ($A' = A - S$.)
7	Copy the content of register s into the multiplier register. ($M' = S$.)
8	Multiply the content of the multiplier register by the content of the register s, and add the result to the content of the accumulator. ($A' = A + MS$.)
9	As for function 8, but subtract the product from the number in the accumulator. ($A' = A - MS$.)
10	Copy the number in the accumulator into register s and clear the accumulator. ($S' = A$, $A' = 0$.)
11	Clear the accumulator. ($A' = 0$; the address part of this instruction is irrelevant.)
12	Divide the content of register s by the number in the multiplier register and place the result in the accumulator. ($A' = S/M$; the divisor is placed in the multiplier register merely for convenience.)

With this code, a sequence for dividing the number in register 72 by the content of register 49 and planting the result in register 86 would be

Function f	Address s	Effect
7	49	The content of register 49 is copied into the multiplier register.
12	72	The content of register 72 is divided by the number planted in the multiplier register by the previous instruction. The result appears in the accumulator, the previous content of which is obliterated.
10	86	The result is copied into register 86 and the accumulator cleared ready for the next step.

It should be noted that these instructions do not alter the numbers stored in registers 49 and 72.

The instructions for carrying out the process described by equations (1.5) can now be written entirely as a sequence of numbers. If the twelve coefficients $a_1, a_2, a_3; b_1, b_2, b_3; c_1, c_2, c_3; d_1, d_2, d_3$ are punched in that order on the data tape, and are followed by the trial solution x_0, y_0, z_0, the programme for a single iteration might be as follows:

1.1 to 1.15	Reading instructions: $a_1 \ldots a_3$ placed in registers 1...3, $b_1 \ldots b_3$ in 4...6, $c_1 \ldots c_3$ in 7...9, $d_1 \ldots d_3$ in 10...12, and x_0 y_0, z_0 in 13, 14, 15.
4.10	$A' = d_1$.
7.13	$M' = x_0$.
9.1	$A' = d_1 - a_1 x_0$.
7.14	$M' = y_0$.
9.4	$A' = d_1 - a_1 x_0 - b_1 y_0$.
7.15	$M' = z_0$.
9.7	$A' = d_1 - a_1 x_0 - b_1 y_0 - c_1 z_0$.
10.16	Result copied into spare register and accumulator cleared.
7.1	$M' = a_1$ (ready for division).
12.16	$A' = (d_1 - a_1 x_0 - b_1 y_0 - c_1 z_0)/a_1 = x_1 - x_0$ from equation (1.5).
5.13	$A' = (x_1 - x_0) + x_0 = x_1$.
10.13	New value of x copied into register 13 and accumulator cleared.
2.13	New value of x printed.
4.11 to 2.14	Similar sequences for calculating y_1.
4.12 to 2.15	Similar sequences for calculating z_1.

The reader will be well advised to work through this sequence in detail. Alternatives are possible, some of which might require fewer instructions. The use of storage registers 1–16 is a purely arbitrary choice.

1.4. The control unit: jump instructions

Machines controlled directly by punched cards or paper tape are quite feasible, and in fact have been produced commercially. They suffer from two drawbacks, however, which have led many computer designers to adopt a different system of control.

In the first place the speed of a machine which reads its instructions direct from punched tape or cards is limited by the mechanical nature of the input device. This will rarely work at more than 50 instructions per second, while electronic circuits can easily be made to operate at more than 1000 instructions per second. In the second place, such a computer cannot easily go back to repeat a series of instructions, nor can it use some numerical criterion to select one of two alternative courses of action unless it has several instruction readers. It is found in practice that facilities for repeating and selecting certain blocks of instructions are extremely useful in designing programmes.

Since instructions can be given the form of numbers, there is no reason why they should not all be placed in the main store of a computer, together with the numerical data, before a calculation is actually commenced. If this is done, it becomes necessary to add to the machine described in § 1.3 a new unit which will be called the 'control' unit, the function of which is to ensure that the instructions are selected from the store and obeyed by the computer in the correct order. This unit contains a register, known as the 'control register', which stores a single number C specifying the address of the instruction being obeyed. It is arranged that when the operation defined by an instruction is completed, the control number C is increased by one, so that the machine passes on to the next instruction. In symbolic terms, each instruction produces the effect $C' = C + 1$.

This method of controlling the computer does not, of itself, increase the overall speed of a calculation, since the instructions presumably have to be fed into the machine from cards or tape in the first place. It is now possible, however, to introduce instructions which alter the content of the control register, thus causing the machine to jump from one point in the programme to another. A simple type of jump instruction is defined as follows:

Function number	Effect
13	Set the content of the control register equal to s. $(C' = s.)$

It will be remembered that after a 'normal' instruction has been carried out, the control number C is increased by one, and the next instruction taken from register $C+1$. The effect of the instruction $13.s$ is to cause the next instruction to be taken instead from register s. As an example, the following sequence calculates $\frac{1}{1-x}$ from the expansion

$$\frac{1}{1-x}=1+x+x^2+x^3+\ \ldots,$$

where the number 1 is assumed to be available in register 18. The number x (<1) is initially stored in register 19, and this register is subsequently used to accumulate the terms of the series.

Address of instruction	Instruction	Effect
50	7.19	$M'=x$.
51	4.18	$A'=1$.
52	5.19	$A'=1+x$.
53	→ 10.19	On the first cycle, $(1+x)$ is placed in register 19.
54	4.18	$A'=1$.
55	8.19	On the first cycle, $A'=1+x(1+x)$.
56	**13.53**	Return to the instruction in register 53 $(C'=53)$.

The reader who follows through this sequence will soon see the power of a few instructions when they form part of a repeated cycle. The second time the instruction in register 53 is reached, its effect is to plant $1+x+x^2$ in register 19. The next cycle calculates $1+x+x^2+x^3$, and so on. A simpler method would be to calculate the individual terms x, x^2, x^3, etc., and sum them, but this would require a few more instructions.

The instruction introduced above can be used to jump forwards as well as back, and its effect can be compared with the jumps which occur in the game of 'Snakes and Ladders'. There is still no choice of next instruction, however, and it is clear that once a loop of instructions has been entered it can never be quitted. A more powerful type of instruction is the 'conditional' jump, whose operation depends on the sign of a certain number, usually the content of the accumulator. A typical example is defined as follows:

Function number	Effect
14	Set the content of the control register to s only if A is positive or zero. $(C'=s$ if $A\geqslant 0$, $C'=C+1$ if $A<0.)$