BRISTOL COLLEGE OF SCIENCE
AND TECHNOLOGY

An Introduction to

Elliott 803 Autocode

By O. B. CHEDZOY

1st MAY 1963 DEPARTMENT OF MATHEMATICS

ELLIOTT 803 AUTOCODE

ERRATA

p.12	line 8	the square root sign should extend over the whole expression $b^2 - 4ac$
p.17	line 3	- Ø246.1235ØØ should read Ø-246.1235ØØ
	line 17	-ØØ241ØØ should read ØØ-241ØØ
p.24 - 27		Due to the unequal size of the printed characters in the text, the results of the calculations in sections 24, 25 and 26 are approximate indications only of how they would appear on a teleprinted sheet.
p.25	line 10	"Marcy" should read "Many"
p.39	line 12	"hold" should read "holds"

Author's Note

The 803 Autocode is primarily used to enable "scientific" calculations to be prepared simply and effectively for the computer. In this description the full flexibility of the Autocode system has not been used; this has been done in order to clarify and accelerate the teaching of the significant powers of the computer and its programming. The full specification should be readily appreciated by anyone who has assimilated this introductory description.

In preparing this description I should like to acknowledge the origination of the Autocode programme by Elliott Bros. and their helpful comments.

I should also like to mention the great assistance given to the preparation of this introduction by the members of the College, particularly in the Department of Mathematics, for their criticisms consequent upon using previous issues for teaching purposes, together with corrections of detail.

1st May, 1963.

O.B.CHEDZOY
Department of Mathematics
Bristol College of Science
and Technology

Contents

Part Three

Part One

1 Introduction

Programming for any computer, and the 803 is no exception, takes the form of breaking down a problem into a list of instructions - called a programme - and allowing the computer to operate on this list of instructions in conjunction with certain numerical data.

For instance, if it is required to write a programme for evaluating the hypotenuse of a triangle, given the values of the other two sides, we would represent this by a mathematical formula

$$a = \sqrt{b^2 + c^2}$$

The evaluation of the formula can be broken down into steps which would follow this pattern:

(1) Multiply b by itself.
(2) Multiply c by itself.
(3) Add the result of (1) to the result of (2)
(4) Find the square root of (3) - and this is the answer a.

We have made 4 simple steps which, in effect, control the calculation. We also raise the questions - where do b and c come from? and what do we do with the answer a?

These are questions which are answered by any system of programming. A framework of rules to be obeyed is set up, and this is done in such a way that almost any problem can be made to fit the rules easily. It is this set of rules and their implementation which is involved in learning to programme the computer, not the problem itself which must be understood in any case.

Like all sets of rules, there are qualifications and extensions, and even places where they may be broken; these are embellishments which it is possible to learn after one has learned the basic rules.

2 Storage and Control

The most essential feature of any computer operating at contemporary speeds is a method of storing numbers. Even in the problem $a = \sqrt{b^2 + c^2}$ the numbers b and c must have been stored somewhere; when b and c were squared, b^2 and c^2 were stored, and even when a was determined, that had to be stored somewhere. If computation takes place at electronic speeds, then storage must clearly be internal also at electronic speeds.

In addition to the storage system for numbers, we must also have a system for holding the list of instructions. It is then possible to place this list of instructions within the computer's storage, together with some numbers for it to use. In all computers, instructions and numbers share the same storage system, although the way in which this is done is not of importance to the Autocode programmer.

Once we have the list of instructions and the numbers within the computer's store, the calculation may proceed automatically under its own control until it has finished the last instruction.

At the start, the control section of the computer looks at the first instruction, identifies it and obeys it. For instance, in the example we had just now, b would be multiplied by itself. Once this operation is complete, the control would ensure that the computer looks at the second instruction.

3 Identification of Storage

Although instructions are held in the same store as the numbers, we consider them as quite different when we are programming.

A normal method of referring to unknown values is by the use of Roman letters with suffices. This autocode system adopts this procedure and the locations of numbers are referred to as

A B C D etc.

The Roman letters are, as shown, in upper case.

This apparent limitation to 26 locations will be removed later - however, they are sufficient to write a fair number of programmes.

4 Instructions - Arithmetic on Numbers.

It is now possible to appreciate the basic forms which an instruction must take. For arithmetic, the four usual operations of addition, subtraction, multiplication and division operate on two numbers - called operands - to produce one result. Bearing in mind that we are going to refer to numbers by the locations in which they are stored, the following is a list of the four basic requirements of arithmetic.

Addition	A=B+C
Subtraction	A=B-C
Multiplication	A=B*C
Division	A=B/C

The 'Addition' instruction may be interpreted by the computer as follows

> "Take the number stored in location B, and the number stored in location C, add them together and then place the result in location A."

It is possible to take liberties with the right hand sides, however. A could be referred to in place of B and/or C as well as on the left hand side. Thus the instruction

A = A + A

means "take the number stored in location A, and the number stored in location A, add them together and place the result in location A." Here it is important to realise that the contents of any particular location are unaltered until the instruction is complete. The instruction

A = A + A

must therefore be interpreted shortly by "Double the contents of A."

A further extension of the arithmetic facilities is possible by the use of constants. Suppose it was required to use the value 16.5B. The appropriate instruction would be

A = 16.5 * B

if the result was required in location A. This technique may be applied to either or both of the operands in the four arithmetic instructions, and also to the setting instruction

A = B (or A = -B)

This merely has the effect of making the number in location A equal to that in B (or equal to -B).

5 Specification of Numbers

Certain restrictions apply to the writing of numbers in preparing a programme. It is necessary to adhere to these absolutely.

For Numbers values may be integers or fractional or mixed. 241.57 is permitted as is 24.157 and .24157. Scaling factors are not allowed, and 3×10^{-4} or $.3 \times 10^{-3}$ must be written as .0003. The maximum size of a number is virtually unlimited, but the first twelve significant digits (decimal point ignored) must not exceed 274, 877, 906, 944.

6 Instructions - Functions of Numbers

A number of mathematical operations are specified by functions - logarithms, exponentials, square roots and so on. These are listed below with their meanings

SIN	sine
COS	cosine
TAN	tangent
ARCTAN	arctangent ($\tan^{-1}$)
LOG	logarithm (base e)
EXP	exponential
SQRT	square root
INT	integral part
FRAC	fractional part
MOD	modulus

The instructions for evaluating a function are as would be expected

A = LOG B
(or A = – LOG B)

which means evaluate the logarithm (to base e) of the number in location B and place this value in location A.

The three trigonometric functions SIN, COS and TAN, operate on angles quoted as a fraction of two rightangles, i.e.

A = COS B

where B is equal to 1/3, gives A a value of .5 (60^0 = .3333 x 180^0). ARCTAN always gives angles in the first ~~two~~ ~~sectors~~. in radians
a least quadrants
$(-90^\circ \to +90^\circ)$.

The function

A = FRAC B

takes the fractional part of B and places it in location A. It should be noted that where

B = 12.23 , A = .23
B = -12.23 , A = -.23

7 Peripheral Equipment of a Computer

In a digital computer, as with any machine, it is necessary to provide it with communication channels so that it may be fed and may receive information. In the case of the 803, these channels take the form of paper tape, punched cards and magnetic tape, although here we are concerned with the medium of paper tape.

Paper Tape provides a means of recording, in a suitable continuous medium form, letters and digits which are punched on a keyboard. The detailed description of the system employed is unnecessary at this stage, since the immediate concern is the preparation of a programme and the data on which it will operate. To this end, we must explain how we give instructions to achieve the "input" of information and the "output" of results.

8 Instructions - Input of Data

READ A

means "Read the next number available at the tape reader and place this number in location A."

9 Specification of Numerical Data

The specification for values of numbers on the data tape is the same as that for numbers in the programme but in this case it is possible to quote a number with a scaling factor of a power of 10;

i.e. in the form	.1234/-2	or .5678/5
are equivalent to	.001234	or 56, 780

i.e. the number after the oblique indicates the power of ten which multiplies the fractional decimal number.

The instruction READ A refers to integral, fractional or mixed numbers as for constants, and also to fractions with a scaling factor.

It is important to realise the difference between constants which are included as part of the programme and the data which the programme itself causes the computer to accept. For instance, one solution of an equation

$ax^2 + bx + c = 0$ is known to be $x = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$

In this case, a programme of instructions may be written to evaluate x, given the values of a, b and c. The programme itself will contain two constants, 4 and 2, but the list of instructions will evaluate x for any value of a, b and c which may be given. Thus the programme would be arranged to call for the three values of a, b and c from the tape reader; the values of a, b and c are regarded as data.

10 Instructions - Output of Numbers

It is necessary to be able to pass the results from the machine's

store to an externally visible one. This is achieved via the paper tape punch, and a PRINT instruction:

PRINT A

which is interpreted by the computer as print out the contents of location A on the tape. (This instruction is incomplete as it stands - further detail will be given in Part II)

11 Instruction - Stop

At the end of a programme it will be necessary to include an instruction to tell the computer to stop. This instruction is simply

STOP

12 Example of a Short Programme (1)

Suppose we now look at the example which was first quoted - to evaluate the hypotenuse of a triangle, given the first two sides - i.e. evaluate

$$\sqrt{a^2 + b^2}$$

The lengths to the two sides are a and b and we want to write the programme to deal with any values of a and b which we may care to present to the computer on paper tape.

Our programme must first call for the values of a and b from tape and then carry out the calculation as determined earlier by the application of our special rules. The programme must then look like this: (and suppose 1st side = 4 and 2nd side = 3).

	Values in locations	
	A	B
READ A	4	-
READ B	4	3
A=A*A	16	3
B=B*B	16	9
A=A+B	25	9
A=SQRT A	5	9
PRINT A	5	9
STOP		

The value printed will be 5.

Part Two

14 Counters

There are, in fact, two types of numerical values which can be held in the computer store:

Numbers and Counters

The Numbers have already been described, and take any value, positive or negative, but counters may take integral values only (positive or negative).

We refer to the locations of Counters by the use of Upper Case Roman Letters, but we use different ones to those used for numbers. In this document the locations of Numbers are referred to as

A B C D etc.

and the locations of Counters as

I J K L etc.

15 Instructions - Arithmetic on Counters

The operations are similar to those on numbers, except that they do not include division.

Addition	I=J+K	
Subtraction	I=J-K	
Multiplication	I=J*K	
Setting	I=J	(or I=-J)

There is NO division operation for counters. J and K may be replaced by I, or by a constant, for example

I = I + 2

Any constants used in conjunction with counters must be integers.

16 Instructions - Functions involving Counters

Certain functions are available for use with counters. The most straightforward of these is

```
I = MOD J
```

which places the arithmetical value (i.e. irrespective of sign) of the number in location J in location I.

The other functions concern both counters and numbers

```
I = INT A
  and
B = STAND J
```

The first of these, I = INT A, places the integral part of the number in location A into counter location I as an integer.

Conversely, B = STAND J, places the integer held in location J into number location B in the appropriate form.

Under no other circumstances can counters be changed into numbers and instructions such as

```
A = B+J
```

will cause failure.

17 Instructions - Input and Output of Counters

The Instruction

```
READ I
```

is interpreted by the computer as 'Read the next number available on the data tape and place it, in integral form, in Counter location I.'

If, however, the number is not an integer, it will not go into a counter location, and will cause failure.

The integer to be read in must be written as 5 and not 5. or 5.0. The maximum value of an integer to be held in a counter is 274, 877, 906, 943.

The contents of a counter may be printed out by using the instruction

PRINT I

but this requires slightly more detail (Section 18).

18 Detail of PRINT Instructions

There are several types of print instructions, but three only are quoted below, and are sufficient for normal use.

(1) PRINT A, M:N
(2) PRINT A, M/
(3) PRINT I, M

In each case the following rules hold:

(i) The first character printed is the sign (- for negative numbers, space for positive numbers)
(ii) Non-significant leading zeros are suppressed (they are replaced by spaces)
(iii) Two spaces always follow the printing of a number.

In the following examples, the spaces are shown by the symbol ϕ.

(1) PRINT A, M:N

means print the contents of location A, with M decimal digits before the point and N digits after. The total number of characters printed is (M+N+4). (M+N) should not be greater than 9.

If A = 246.123473, then

PRINT A, 3:4 will cause ϕ246.1235$\phi\phi$ to be printed and

PRINT A, 4:4 will cause $\phi\phi$246.1235$\phi\phi$ to be printed, but

PRINT A, 2:4 will cause a question mark on the next line followed by the number printed out in the form A, 9/ (see below) indicating that it was larger than anticipated.

If A = -246.123473 then

PRINT A, 3:4 will cause -246.1235$\phi\phi$ to be printed, and

PRINT A, 4:4 will cause -246.1235$\phi\phi$ to be printed.

(2) PRINT A, M/

means put the contents of location A, in scaled form with M digits accuracy. The total number of characters printed is (M+8). M should not exceed 9.

If A = -246.12347, then

PRINT A, 7/ will cause -.2461235/ϕ03$\phi\phi$ to be printed, and

If A = .00002314, then

PRINT A, 3/ will cause ϕ.231/-04$\phi\phi$ to be printed.

(3) PRINT I, M

means print the value held by counter I as an M decimal digit number. The total number of characters printed is (M+3). M should not exceed 12.

If I = -241, then

PRINT I, 5 will cause -241$\phi\phi$ to be printed.

19 Instructions - Control of Layout

The layout of the printed values is important - as also is the correct heading or title at various points.

The instruction

TITLE
VALUES OF F(X)

(3 sp) X (10 sp) F(X)//

will print

VALUES OF F(X)

$\phi\phi\phi$X$\phi\phi\phi\phi\phi\phi\phi\phi\phi\phi$F(X)

The title is always terminated by //

So far, we have considered the motion of a printer across the

page. We can also instruct it to start a new line at the left hand end of the margin. This is effected by the instruction

LINE

However, on occasions it may be desirable to space several sets of results by more than one line. Instead of writing this instruction five times, the equivalent may be achieved by

LINES 5

It may be desirable to separate columns of figures being printed - this may be achieved by the instruction

SPACES 1

or

SPACES 7

causing 1 or 7 spaces to be printed.

The width of a teleprinter page is $8\frac{1}{2}$ inches, but only 69 characters may be printed across it - a character occupying $\frac{1}{10}$ inch. A line is $\frac{1}{6}$ inch high.

NOTE It is always highly desirable to give each problem a suitable title. Considerable attention must be paid to the layout of results for simple and effective use to be made of them.

20 Decisions within a Computer

So far, in the description and examples, we have considered only the case where a programme is a list of instructions to be sequentially obeyed. This, as you will learn later, does not represent the most efficient way in which a computer may be used - and indeed it is limiting.

For instance, the previous example of finding the root of a quadratic needs a decision to make it foolproof if $b^2<4ac$, then the root will be complex and we shall want to deal with the answer in a different manner.

All decisions may be said to rest on the judgment of series of problems each of which may be answered either "yes" or "no". It is a common case to have a number of possible answers to

a question - but these may be split into a series of subsidiary simple questions which, when taken as a whole, represent a complicated question.

A question which may have two possible answers requires normally, in any instruction routine, two possible courses of action. In a programme where all instructions are normally in sequence, the only possible alternative is to arrange for a break in the sequence of instruction - or, in computer terms a "jump."

However, when there is a break in the sequence of instructions, or a choice of courses of action, it is necessary to be able to specify the next instruction if the sequence is to be broken. This is effected by a numbering system, using Reference Points. Thus, if the sequence is to be broken, the instruction

JUMP @ 3

will say, in effect, jump to reference point 3.

21 Reference Points

Reference points are numbered from 1 upwards. Reference point 1 is usually the start of the programme. The other reference points are placed in association with instructions which are required for various entry and re-entry positions after using jump instructions, but need not occur in numerical order.

22 Jump Instructions

There are three types of Jump instruction - all of which refer to reference points as a new source from which to accept instructions. These are:

JUMP @n
JUMP IF A=B@n
JUMP UNLESS A=B@n

The first of these is an unconditional jump - i.e. it always causes the sequence of instructions to be broken and for the next instruction to be that at reference point n.

The second of the instructions, JUMP IF, means compare the two quantities which are given, and if that statement is satisfied, then the next instruction is that at reference point n, if the statement is not satisfied, then the next instruction is the one in immediate sequence. In this case the contents of location A are compared with those of location B. If, and only if, they are equal, will a jump be made.

The third of these instructions, JUMP UNLESS, treats the statement in the reverse manner; a jump is made if, and only if, it is not satisfied. In the above example, the contents of location A are compared with those of location B: if they are equal no jump is made; if they are unequal a jump to reference point n is made.

The examples given above use the idea of A = B as the condition to be investigated. The condition chosen may be any of those which have been detailed in the arithmetic instructions or function instructions. In addition, the = sign may be replaced by an unequal sign, either > or <. It is therefore possible to have a very large number of conditions to be satisfied.

Typical examples are:

A=B	A<47	A> 4.35
A=B/2.5	A<B+C	A> 4*B
A=4+B	A<LOG B	A> SIN B
I=J	I<O	I> 7
I=J*3	I<J-14	I> 6+J

Example

In the case of the example given in Section 13, the programme may now be extended to test for imaginary values in the solution of a quadratic equation - this is the case where $b^2 - 4ac$ is negative. (Example $4x^2 + 2x + 5 = 0$)

```
                                  Contents of Locations
                                  D     E
1)  READ A                        -     -
    READ B                        -     -
    READ C                        -     -
    D=B*B                         4     -
    E=A*C                         4     20
    E=4*E                         4     80
    JUMP IF D<E@2                 Test satisfied, next instruction
    D=D-E                         at Reference Point 2
    D=SQRT D
    D=D-B
    E=2*A                         Not obeyed in this example
    D=D/E
    PRINT D,1:4
    STOP
2)  TITLE IMAGINARY//             'IMAGINARY' printed
    STOP
```

23 Setting Instructions

The computer has to be fed its instructions via the tape reader; it is not psychic and it is necessary to tell it at the beginning of the tape the general dimensions of the problem it is about to be given, so that it can reserve the right number of locations for numbers, counters etc.

There are 5 setting instructions altogether; four of these are before the commencement of the instructions, the other is after the completion.

The four instructions at the beginning are as follows, where the four letters on the left indicate the type of setting.

```
SETS IJK
SETV ABC
SETF TRIG SQRT LOG EXP
SETR 16
```

The first line, SETS, means that the following locations for counters are being used; I, J and K. Each letter used in the programme as a counter location must be nominated.

The second line, SETV, means precisely the same thing referring to numbers; here A, B and C are used.

SETF is followed by a list of the functions which are used in the programme. Setting them ensures that they are properly incorporated.

Any of these three settings may be omitted from Setting. If for instance, there are no counters and no functions being used one needs to SETV only.

The last of these four settings is SETR - this is to indicate the number of reference points used in the programme. This must always be used, and of course there always is at least one reference point at the start. In this case SETR 16 means that the highest reference point is 16 - and although it is possible for some of the intermediate ones to be omitted this is not usually the case.

A START instruction must be placed at the end of the tape; for example, we use

START 3

where this particular instruction means that the start of the programme is at reference point 3. Normally, of course, the starting point will be at reference point 1, in which case the instruction is

START 1

24 Complete Example (1)

The following is a complete example for extracting the roots of any quadratic equation with full consideration of all possibilities, and for the layout and correct heading of the results.

	Programme	Notes
		No counters set
	SETV ABCDEF	6 numbers A-F used
	SETF SQRT	Function used
	SETR 4	Highest Reference Point
1)	TITLE ROOTS OF A QUADRATIC EQUATION//	
	LINES 2	Double line space
	READ A	
	PRINT A, 6/	Occupies 14 characters
	READ B	
	PRINT B, 6/	Occupies 14 characters
	READ C	
	PRINT C, 6/	Occupies 14 characters
	LINES 2	Double line space
	D=B*B	b^2
	E=A*C	
	E=4*E	4ac
	JUMP IF D<E@4	Test for Complex Roots
	JUMP UNLESS D=E@3	Test for Real Roots
2)	TITLE EQUAL ROOTS//	Occupies 11 characters
	SPACES 3	Makes it 14 characters
	D=-B	} Evaluates -b/2a
	E=2*A	
	F=D/E	
	PRINT F, 6/	Prints below b (14 characters)
	STOP	START instruction only at end
3)	TITLE REAL ROOTS//	Occupies 10 characters
	SPACES 4	Makes it 14 characters
	D=D-E	} $\sqrt{b^2-4ac}$
	D=SQRT D	
	E=-B	} $\frac{-b-\sqrt{b^2-4ac}}{2a}$
	F=E-D	
	F=F/A	
	F=F*.5	
	PRINT F, 6/	Prints below b (14 characters)
	F=D+E	} $\frac{-b+\sqrt{b^2-4ac}}{2a}$
	F=F/A	
	F=F/2	
	PRINT F, 6/	Prints below c
	STOP	

(Programme continued overleaf)

4) TITLE COMPLEX ROOTS//	Occupies 13 characters
SPACES 1	Makes it 14 characters (NB not SPACE)
D=-B	
D=D/A	-b/2a (Real Part)
D=D/2	
PRINT D, 6/	Prints below b (14 characters)
E=E-D	
E=SQRT E	$\frac{\sqrt{4ac-b^2}}{2a}$
E=E/A	
E=E/2	
PRINT E, 6/	Prints below c
STOP	
START 1	Starting Reference point at end of programme only.

Special Notes

1 Reference point 2 is never referred to - it is redundant, but it is not necessary to delete it.

2 The evaluation of -b/2a may be done in any order (here three different ways are used) allowed by the rules of arithmetic.

3 Care has been taken to align the answers immediately below the coefficients of the equation which are printed out each time. This is illustrated by the following three examples.

(i) $4x^2 - 20x + 25 = 0$

(ii) $2x^2 + 17x + 2 = 0$

(iii) $x^2 - 6x + 25 = 0$

The results for these three examples are shown below in exactly the form which would be produced by the computer.

Example (i)

ROOTS OF A QUADRATIC EQUATION

.400000/ 01 -.200000/ 02 .250000/ 02

EQUAL ROOTS .250000/ 01

Example (ii)

ROOTS OF A QUADRATIC EQUATION

.200000/ 01 .170000/ 02 .210000/ 02

REAL ROOTS -.150000/ 01 -.700000/ 01

Example (iii)

ROOTS OF A QUADRATIC EQUATION

.100000/ 01 -.600000/ 01 .250000/ 02

COMPLEX ROOTS .300000/ 01 .400000/ 01

25 A Simple Loop - Complete Example (2)

Many computer programmes contain loops; this is the description applied when a group of instructions within the programme is obeyed more than once in its execution by the computer. A very simple example of a loop is encountered in the following example, which is specified in problem form:

Problem

A set of positive values of x (all less than 1) exist on a data tape. At the end of these values a zero is punched.

Write a programme to evaluate the function

$$\text{Log}_e(6x-3+2/x)$$

printing out all values to 5 decimal places in tabular form, so that the tape will stop on encountering the zero value.

	Programme	Notes
		No counters set
	SETV AB	Numbers held in A and B only
	SETF LOG	Log function required
	SETR 3	3 reference points
1)	TITLE	Title with heading for each
	VALUES OF A FUNCTION	column
	(3sp) X (6sp) FUNCTION//	
2)	READ A	Read from tape, start of loop
	JUMP UNLESS A=0@3	If zero,
	STOP	Stop

```
3)  LINE                    Start a new line
    PRINT A, 0:5            Print A, 9 characters printed
    B=2/A
    B=B-3
    A=6*A                   Calculation
    B=A+B
    B=LOG B
    PRINT B, 2:5            Print B, 11 characters printed
    JUMP @2                 End of loop
    START 1
```

Special Notes

1 The accuracy of the numbers is given in the question; the extreme values of the function may be calculated.
2 Zero is used as the end-of-tape character.
3 There is no STOP as the last instruction, this is an unconditional JUMP back to the beginning of the loop.
4 The loop consists of the instructions between reference point 2 and JUMP @2, except the STOP instruction.
5 The effect of the jump instruction is usually shown in a programme by arrows. This simplifies the understanding of the programme by the writer and anyone else, but it is not of course necessary for the computer.
6 The appearance of the printing will be:

VALUES OF A FUNCTION

X	FUNCTION
.91692	1.54388
.87264	1.51022
.72538	1.41329
.61864	1.37238
.60813	1.37056
.59951	1.36943

26 A Simple Counting Loop - Complete Example (3)

The number of times which a loop is repeated may be simply controlled by a counter. This identifies the main role of counters - as a numerical control on the calculation.

Problem

Ten values of an angle A in degrees ($A<180^0$) exist on a data tape. Write a programme to tabulate sin A + cos A against A for each of these values, to 3 d.p. accuracy.

Programme

```
    SETS I
    SETV AB
    SETF TRIG
    SETR 2
1)  TITLE
    SIN A + COS A
    (4sp) A (7sp) F(A)//
    I=10
2)  LINE <-------------------+
    READ A                   |
    PRINT A, 3:3             |
    A=A/180                  |
    B=SIN A                  |
    A=COS A                  |
    A=A+B                    |
    PRINT A, 1:3             |
    I=I-1                    |
    JUMP UNLESS I=0@2 -------+
    STOP
    START 1
```

An illustration of typical output is

```
SIN A + COS A
    A        F(A)
 172.316   -0.857
 168.422   -0.779
 161.311   -0.627
 122.262    0.312
 121.989    0.318
 120.003    0.366
  81.472    1.137
  46.222    1.414
  41.119    1.411
  18.689    1.268
```

Miscellaneous Examples

(1) A data tape has four values, a, b, c and d. Prepare a programme to evaluate the expression

$$\frac{a^2 + bc - ad}{b + c}$$

printing the answer with 2 integer figures and 3 decimal places.

(2) A data tape carries a series of values of x ($x<1$), terminated by a zero value. Write a programme to evaluate the polynomial $4x^3 + 3x^2 + 7x - 5$, tabulating the answers to an accuracy of 3 d.p.

(3) Prepare a programme to accept three different numbers from a data tape and print them out in descending order of magnitude. (5 significant figures, 2 d.p.)

(4) A data tape has a value a followed by three possible values of b; this sequence of numbers is repeated until a is given a zero value. Evaluate log $(a - 3b^2)$ and tabulate your results; in the cases where the expression is indeterminate, the printing of a value is replaced by spaces. (The results will not exceed 9 and accuracy is required to 4 d.p.)

(5) A set of values on a data tape represent the number purchased of a certain product. Orders of up to a dozen are 10/- each; up to 6 dozen at 9/6d each; higher quantities at 9/3d each. Write a programme to evaluate the cost in £ of the numbers on the tape. The last value on the tape is zero.

(6) A data tape carries the hours worked each day by a certain employee. The standard working day is 8 hours, for which he is paid at 5/- per hour and overtime is paid at $1\frac{1}{3}$ rate. Write a programme to read the data and evaluate his gross pay for the week, in £.

(7) Extend the programme for question 6 to allow for $1\frac{1}{2}$ rate on Saturdays and double rate on Sundays, when no normal time is worked.

(8) Two values exist on a data tape represent angles A and B, both in degrees. Write a programme to tabulate tanA,

tan(A+B), tan(A+2B), tan(A+3B), up to tan(A+9B), against the angle.

(9) Write a programme to evaluate the square root of a number x read from a data tape, using a second data number as a first approximation a, and where a second approximation a_2 is given by

$$a_2 = a_1 - \frac{a_1^2 - x}{2a_1}$$

Print out the result a_2 and the difference $a_2 - a_1$.

(10) Extend the programme in question 9 so that a third approximation a_3 may be made using a_2 in place of a_1, and then a fourth and so on until the differences between two successive approximations is less than .0001.

Part Three

27 Extensions to Store Numbering - Numerical Suffices

In Section 3 we identified the various locations in the store supply by means of the Roman letters, A, B, C etc.

It is, however, necessary to extend the store numbering system, otherwise we are clearly limited to a total of 26 locations for numbers and counters. This extension is achieved in the usual way by employing suffices to the letters. Thus a value which might normally be referred to in mathematical notation as a_{15} becomes A(15), the suffix being quoted in brackets. Similarly b_7 becomes B(7) and c_{33} becomes C(33).

Instructions within a programme accept this suffix notation in exactly the same way as has been the case for no suffices; i.e. an addition instruction might be

```
A(15)=B(7)+C(33)
      or
E=B(14)+E
```

Note that the suffices must be integral and positive, but zero is permitted; in fact A(0) is identical to A.

28 Extensions to Store Numbering - Variable Suffices

We may wish, however, to refer to locations in the store which we are unable to identify at the time of writing other than by means of a general suffix i such as pertains to a_i. In the Autocode, we adopt a similar system, which refers to A(I).

As an example of the way in which a suffix I may be used, consider the case of 3,500 customers of a firm having outstanding accounts, the values being held in locations A(1) to A(3500) corresponding to the account numbers of the customers.

To find the value of any particular account, where the number of the account is available on a data tape, the programme need comprise two instructions

```
READ I
PRINT A(I)
```

The first instruction causes the number of the account (as an integer) to be read and placed in counter I. The contents of location A(357) - i.e. the value of account 357 - will be printed when 357 is on the tape.

NOTE All the suffices must be integral and positive or zero - therefore the suffices must refer to counters and not to numbers, even if the number happens to be an integer.

29 Extensions to Store Numbering - Complex Suffices

There is considerable flexibility in the way suffices may be described. For instance, in the problem quoted above, the accounts could be held in locations A(101) to A(3600) in which case the instruction would be

```
READ I
PRINT A(I+100)
```

The value of I is unaltered by such an operation - it is only the value of the suffix which is I+100.

Other forms of suffix which are permissible include

```
A(5I)
A(2I+3)
A(I+J)
A(4I+J)
A(I+3J)
```

All suffices must include combinations of counters or integer constants. The overall value of a suffix must never be negative.

30 A Simple Counting Loop using a Variable Suffix (1)

A frequent requirement in a problem is to operate successively on numbers in store locations. An obvious case of this is when there is a set of numbers on a data tape which must be placed in successive locations from A(0) upwards, until, say, a zero value is encountered to mark the end of the data.

The following programme would suffice

```
1)  I=0
2)  READ A(I)<----------------
    I=I+1                    |
    JUMP UNLESS A(I-1)=0@2 ---
    ..............
```

I=0 initially so that the first time the READ instruction is obeyed the value read will go into location A(0); I is then increased by 1 and the process repeated until the zero value is found.

31 A Simple Counting Loop using a Variable Suffix (2)

Alternatively the problem might be to read 50 numbers from the tape and place them in successive locations A(0) to A(49).

In this case the programme would be

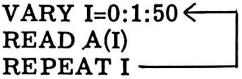
```
1)  I=0
2)  READ A(I)<----------------
    I=I+1                    |
    JUMP UNLESS I=50@2 -------
    ..............
```

The process is similar to that used in Section 30, but the control this time is exercised by the counter, now serving the dual purpose of limiting the number of times the loop is repeated and the placing in successive locations of the values.

32 The VARY Instruction, using Counters.

The VARY instruction is used to simplify the process of pre-

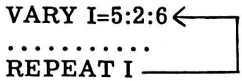
paring counting loops. As an example, the previous counting loop in Section 31 above, would become



where the instructions VARY and REPEAT mark the beginning and end of the loop.

The instruction VARY I defines the controls of the loop. In the above case, it means that I successively assumes the values of 0, 1, 2,.....,49: i.e. the loop starts by assuming that I=0 the first time and I increases by 1 each time the loop is repeated until the loop is completed 50 times in all. The REPEAT I instruction is associated with the previous VARY I instruction in the programme.

Similarly

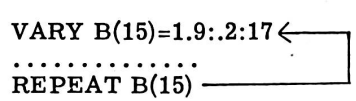


means that a loop will be started with I=5, and repeated for I=7, 9, 11, 13 and 15, making a total of 6 passages in all. Note that the values of I should not be interfered with during loop, unless it is expressly desired.

33 The VARY Instruction, using Numbers.

The VARY instruction may equally well be used for controlling loops where numbers are being varied in steps.

For instance



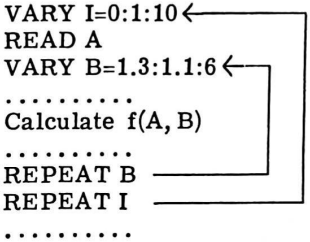
would be interpreted as B(15) assuming the value 1.9 for the first time, and on each repeat increasing by .2 to 2.1, 2.3,

etc. until the 17th time when the final value of B(15) will be 5.1. The number of times which the loop is performed is clearly integral and positive.

34 Loops within Loops

It is necessary and desirable at times to be able to control loops from more than one source. Consider the problem involved in calculating the value of a function f(a,b) for values of b of 1.3, 2.4, 3.5, 4.6, 5.7, 6.8 and for 10 values of a which are read from a data tape.

The programme for this would take the form



35 Setting Instructions - Extension of Store Numbering

The extensions of store numbering discussed earlier must be taken into account in the setting instructions for both counters and numbers. For instance, if counters I(0) up to I(10) are used, together with J, and numbers A, B(0) to B(22) and C(0) to C(15) are used the two pertinent setting instructions are

```
SETS I(10)J
SETV AB(22)C(15)
```

i.e. the highest suffix used only is quoted respectively after the counter or number.

If a variable suffix has been used, the highest value of the suffix to be encountered must be entered. If this value is not known, then a good estimate of the top value plus a small margin for error should be allowed.

36 Complete Example (4)

Problem

In a factory it is found that a machine operator's productivity is sensitive to the pressure under which he is required to work. It is found that the number of faulty items he makes per hour is given by .0011 n^3 where n is the number of attempted items per hour. (If n = 20, .0011 n^3 = 8.8 meaning that 9 must be scrapped).

The profit on a completed item (excluding labour costs) is £.25 and the loss on each scrap item is £.1. In order to design a bonus scheme, it is required to tabulate the following: Number of attempts per hour, Number of scrap, Number of good items, Total profit on good items, Total loss on scrap, Margin of overall profit. (n may be taken in the range 5 to 24 inclusive).

Method

The programme below illustrates the use of VARY and the INT and STAND instructions. The method uses an overall loop to consider each value for the number of attempts as a counter. This is turned into a number to evaluate .0011 n^3 and the integral part is placed in counter J and increased by 1. (No value of .0011 n^3 can be evaluated as a whole number, therefore the number of scrap items is one more than the integral value of .0011 n^3). This value is printed and the number of good items is found by subtraction. These two counters are turned back to numbers to evaluate the profit, loss and margin, thus enabling the table to be printed out as shown.

NOTE No input data are required in this case.

Programme to Calculate Productivity Estimates

```
    SETS IJK
    SETV ABCD
    SETF INT FRAC STAND
    SETR 1
1)  TITLE
    PRODUCTIVITY ESTIMATES


    ATTEMPTS  SCRAP  GOOD  PROFIT  LOSS  MARGIN//
    VARY I=5:1:20 <-----------------------------
    PRINT I, 5                                  |
    A=STAND I                                   |
    B=A*A                                       |
    B=B*A                                       |
    B=.0011*B                                   |
    J=INT B                                     |
    J=J+1                                       |
    PRINT J, 5                                  |
    K=I-J                                       |
    PRINT K, 5                                  |
    B=STAND K                                   |
    C=STAND J                                   |
    B=B*.25                                     |
    PRINT B, 4:2                                |
    C=C*.1                                      |
    PRINT C, 4:2                                |
    D=B-C                                       |
    PRINT D, 4:2                                |
    LINE                                        |
    REPEAT I ------------------------------------
    STOP
    START 1
```

Table produced by Programme for Productivity Estimates

PRODUCTIVITY ESTIMATES

ATTEMPTS	SCRAP	GOOD	PROFIT	LOSS	MARGIN
5	1	4	1.00	0.10	0.90
6	1	5	1.25	0.10	1.15
7	1	6	1.50	0.10	1.40
8	1	7	1.75	0.10	1.65
9	1	8	2.00	0.10	1.90
10	2	8	2.00	0.20	1.80
11	2	9	2.25	0.20	2.05
12	2	10	2.50	0.20	2.30
13	3	10	2.50	0.30	2.20
14	4	10	2.50	0.40	2.10
15	4	11	2.75	0.40	2.35
16	5	11	2.75	0.50	2.25
17	6	11	2.75	0.60	2.15
18	7	11	2.75	0.70	2.05
19	8	11	2.75	0.80	1.95
20	9	11	2.75	0.90	1.85
21	11	10	2.50	1.10	1.40
22	12	10	2.50	1.20	1.30
23	14	9	2.25	1.40	0.85
24	16	8	2.00	1.60	0.40

37 Subroutines

It sometimes occurs during a calculation that a set of instructions to perform a certain function has to be repeated since there appears to be no simple way of using a loop to effect this without considerable complication.

The instruction

SUBR 5

has the effect of making the computer take its next instruction at reference point 5 (as in JUMP @5), but on reaching an instruction

EXIT

the computer will take further instructions from immediately after the SUBR 5 instruction.

An example of the use of the subroutine function will clarify the method.

38 Complete Example (5)

Problem

Write a programme to evaluate y where $y = \sqrt{x^2 + 1} + \sqrt{4x^2 + 9}$ for x = 0 to x = 2 in steps of .2, to an accuracy of 3 d.p.

Programme

```
    SETV ABCDX
    SETF SQRT
    SETR 2
1)  TITLE
    EVALUATION OF A FUNCTION//
    VARY X=0:.2:11
    A=X
    B=1
    SUBR 2
    D=C
    A=2*X
    B=9
    SUBR 2
    C=C+D
    PRINT C, 1:3
    REPEAT X
    STOP
2)  C=A*A
    C=C+B
    C=SQRT C
    EXIT
    START 1
```

NOTE The subroutine commences at reference point 2 and evaluates $\sqrt{A^2 + B}$ and places the result in C each time it is entered. The first time it is used, the EXIT instruction refers back to D=C, and the second time it is used, the EXIT instruction refers back to C=C+D. Care must be taken to make sure that the data for the subroutine are in the correct locations and that the subroutine does not obliterate any useful results so far obtained.

39 Further Flexibilities of Autocode

In certain operations, further flexibilities of the autocoding system may be achieved by using the contents of a counter location where hitherto integers have been used for clarity.

JUMP Instructions

The three types of JUMP instruction all terminate in a specification of the reference point e.g.

```
JUMP @4
```

This reference point may be the contents of a counter location - for example

```
JUMP @J
```

and if counter J holds 7, the effect of the instruction is to be obeyed as

```
JUMP @7
```

This type of operation is used in the last complete example (6).

PRINT Instructions

The three print instructions may be quoted as

```
PRINT A, J:K
PRINT A, J/
PRINT I, J
```

i.e. the actual PRINT Instructions may have their number of positions governed by the contents of counter locations. Thus if counters J and K contain 3 and 5 respectively, then

```
PRINT A, J:K
```

will be interpreted as

```
PRINT A, 3:5
```

SUBROUTINE Instructions

It is desirable at times to specify subroutines by the contents of a counter location. For example

SUBR L

where L is a counter which contains 12, is interpreted as

SUBR 12

and will enter the subroutine which commences at reference point 12.

40 Complete Example (6)

It is essential in the writing of programmes for any problem to be able to see quite clearly how the problem is to be tackled. This programme is a fairly complicated example of a short programme.

Problem

The four main batsmen of a cricket side are called Smith, Jones, Green and Morgan. Their scores in each innings are recorded on a data tape, starting with those of Smith, and concluding with those of Morgan. All scores are recorded as the integers which they represent apart from not out scores which have 1000 added to them. After the last score for each batsman, a negative number is punched on the data tape.

Write a programme to accept this data tape and to tabulate the following for each batsman - Name; Number of Innings; Number of times not out; Total Runs scored; Highest Score; Average (total runs/times out). The figures for each batsman must be tabulated in order of decreasing averages.

Method adopted

The method adopted must clearly be to gather all the data before any print out commences. The table below gives an

illustration of the way in which the stores are allocated to the various numbers.

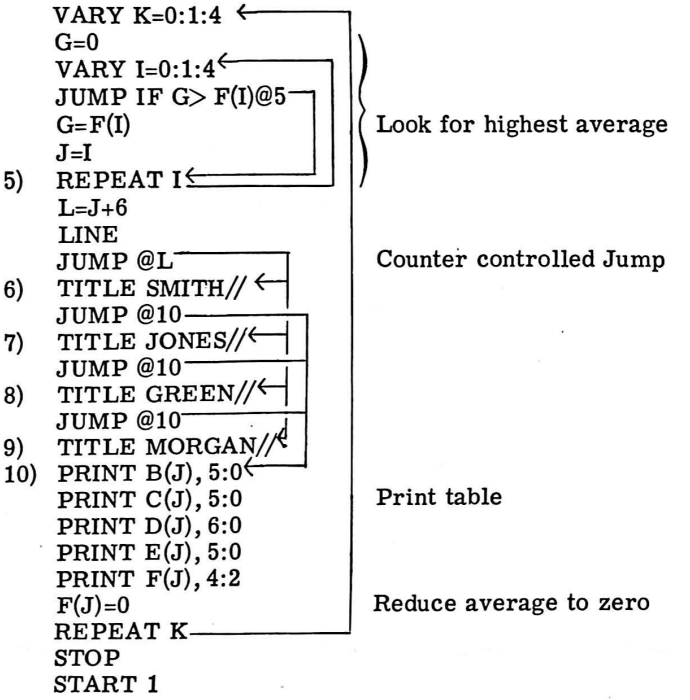
	Store	B	C	D	E	F
		Total Innings	Times not out	Total runs	Highest score	Average
Smith	0					
Jones	1					
Green	2					
Morgan	3					

These 20 numbers must be evaluated before any printing. All the data is in groups according to each of the four batsmen. Procedure to be adopted is to read Smith's data (first group) and then to adjust the totals as each number is read, after ensuring that all the totals start as zero. As soon as the negative number is encountered, the average is filled in. The process is repeated for the four remaining groups, the general suffix I taking values 0, 1, 2 and 3 successively.

After the conclusion of this part of the programme, the remainder of the programme commences with a search for the highest average, each average being compared with the highest so far found in G. When the highest has been found, the appropriate name is printed (selected by the JUMP @L instruction) followed by the appropriate values across the table. The process is then repeated for the remaining three times to provide the full table; notice that at the end of the print out for a batsman his average is made zero - this ensures that it is not considered again for inclusion in the table.

Programme to Calculate Cricket Batting Averages

Programme	Notes
SETS IJKL	
SETV AB(3)C(3)D(3)E(3)F(3)G	
SETR 10	
1) TITLE	
CRICKET BATTING AVERAGES	
INNS N.O. TOTAL HIGH AVGE//	
VARY I=0:1:4	
B(I)=0	
C(I)=0	Set totals to zero
D(I)=0	
E(I)=0	
2) READ A	Look for negative number
JUMP IF A<0@4	
JUMP IF A<1000@3	Look for "not out"
A=A-1000	
C(I)=C(I)+1	Increase "Times not out" by 1
3) B(I)=B(I)+1	Increase "Innings" by 1
D(I)=D(I)+A	Increase "Total"
JUMP IF E(I)> A@2	
E(I)=A	Check highest score
JUMP @2	
4) F(I)=B(I)-C(I)	
F(I)=D(I)/F(I)	Calculate average
REPEAT I	


VARY K=0:1:4
G=0
VARY I=0:1:4
JUMP IF G> F(I)@5
G=F(I)
J=I
5) REPEAT I
L=J+6
LINE
JUMP @L
6) TITLE SMITH//
JUMP @10
7) TITLE JONES//
JUMP @10
8) TITLE GREEN//
JUMP @10
9) TITLE MORGAN//
10) PRINT B(J), 5:0
PRINT C(J), 5:0
PRINT D(J), 6:0
PRINT E(J), 5:0
PRINT F(J), 4:2
F(J)=0
REPEAT K
STOP
START 1
Look for highest average
Counter controlled Jump
Print table
Reduce average to zero

Input Data for Cricket Batting Averages

13	
27	
85	
142	
1046	Smith
2	
45	
72	
1012	
-1	
96	
89	
12	
47	
13	
0	Jones
78	
81	
64	
1065	
-1	
33	
38	
47	
112	
101	Green
0	
2	
5	
1	
1000	
-1	
1061	
1002	
1049	
1008	Morgan
1021	
46	
32	
72	
-1	

Illustration of Output

CRICKET BATTING AVERAGES

	INNS	N.O.	TOTAL	HIGH	AVGE
MORGAN	8	5	291	72	97.00
SMITH	9	2	444	142	63.43
JONES	10	1	545	96	60.56
GREEN	10	1	339	112	37.67

Miscellaneous Examples on Part Three

1 A data tape contains 60 positive numbers, the second 30 being "weights" attributed to the respective observations of the first 30. Evaluate by programme the mean value per unit weight. (The final mean is less than 100 and should be printed to 3 dec. places).

2 Write a programme to evaluate the amount of income tax payable by persons with incomes varying from £1,000 to £2,000 in steps of £50 assuming the following:

(i) Allowances totalling £850.
(ii) Income Relief at 2/9 of income.
(iii) Reduced tax on £50 at £.125 per £.
(iv) 2nd Reduced tax on £200 at £.375 per £.
(v) Standard rate of tax at £.45 per £.

Tabulate tax against income, tax being printed to 2 dec. places.

3 Prepare a programme to read a set of 50 numbers from tape into the store. Arrange these numbers in order of descending magnitude - and print them out in that order. (You may assume that all numbers are different, but may be positive, negative or zero). They include fractional numbers to 2 dec. places accuracy).

4 A certain democracy, with 100 representative seats divided between two parties A and B, shows that the

distribution of seats approximately obeys the relationship

$\frac{V_A^3}{V_B^3} = \frac{S_A}{S_B}$, where V_A and V_B are the votes cast for the two parties, and S_A and S_B are the seats obtained respectively.

The results of the first 10 declarations are available on a data tape, party A first. Write a programme to predict the final seat distribution after each result tabulating the results to include - result number, total votes so far for each party, predicted seats so far for each party.

5 £1,000 is invested at a fixed rate of compound interest of $2\frac{1}{2}\%$ after tax. At the same time, there is a continuous devaluing of currency at the rates quoted for 20 years successively on tape (3% devaluation means that £1 will buy this year only 97% of what it bought last year). Prepare a programme to accept these values from tape, and tabulate for each year: the year number, the face value of the investment, the present purchasing power of the investment compared with the original investment. (3 dec. place accuracy required).

6 A data tape has integral values all lying between 20 and 40 inclusive, except the last one. Write a programme to tabulate the frequency with which each of these values occurs.

7 An index of industrial share prices is based upon the following investments of a total of £1,000, the base of the index being 100.

Investment	A	88 units of	£2.5
"	B	90 " "	£2
"	C	20 " "	£5
"	D	200 " "	£1.25
"	E	40 " "	£4
"	F	30 " "	£3

The value of the share prices for each day is prepared on a data tape in this order. Write a programme to

evaluate and print the % rise or fall of each share and the value of the new index. (2 dec. place accuracy required).

8 The temperature of a critical part of a nuclear reactor (operating at an optimum temperature of 400° C) is supplied direct to a computer (in this case it may be simulated by a data tape). Write a programme to accept these values and to print out the values together with reasons for the printing taking place, under the following conditions:

(i) Alarm conditions - when a tolerance of 40° C from the optimum is exceeded.
(ii) Warning conditions - when a change from the previous reading is more than 15° C.
(iii) Normal conditions - at every 20th reading, the 16th - 20th readings are printed consecutively.

9 Four teams play a tournament at football, the matches being played in the following order:

(1)	North v. South	(4)	East v. South
(2)	East v. West	(5)	West v. North
(3)	West v. South	(6)	North v. East

The number of goals scored in each of these matches are recorded in this order on the data tape.

Write a programme to list the teams in order of their success, together with the points obtained by each.

NOTE The success of a team is measured by points - 2 pts. for a win, 1 pt. for a draw, 0 pt. for a loss. Their overall success is the sum of these points.

10 A totalisator is in operation for a race with 5 starters. The bets places are recorded on tape with horse number (1, 2, 3, 4 or 5) followed by the number of units staked. Write a programme to accept all bets and record them until the "off" - this being indicated by a zero horse number.

To evaluate the dividend, 15% of the units staked on losing horses is deducted from the "pool", the remainder being paid as prize money equally distributed over the winning units. Extend the programme to print the dividend per unit staked should any of the five horses win. (3 d.p. accuracy required).

Produced by the NEOPRINT system

set on electric keyboards and printed by offset lithography by Unwin Brothers Limited

AT THE GRESHAM PRESS
Old Woking Surrey England