STICHTING
MATHEMATISCH CENTRUM
2e BOERHAAVESTRAAT 49
AMSTERDAM

DR 3

The language of automatic calculating machines

H.D.Huskey.

1949

Printed at the Mathematical Centre, 49, 2e Boerhaavestraat, Amsterdam.

The Mathematical Centre, founded the 11-th of February 1946, is a non-profit institution aiming at the promotion of pure mathematics and its applications. It is sponsored by the Netherlands Government through the Netherlands Organization for the Advancement of Pure Research (Z.W.O), by the Municipality of Amsterdam, by the University of Amsterdam, by the Free University at Amsterdam, and by industries.

THE LANGUAGE OF AUTOMATIC CALCULATING MACHINES

by

Harry D. Huskey

0. INTRODUCTION

0.1. A calculating machine does sequences of arithmetic operations on numbers. It is automatic if it carries out long sequences of such operations without manual assistance. In order to carry out such sequences it is necessary for the device to do other things besides the four basic arithmetic operations of addition, subtraction, multiplication and division. There must be a means of reading in initial data and producing recorded answers. The device must be able to store numbers and instructions which specify the sequence of operations that is to be performed. In order to minimize the human effort in preparing a problem it is usually desirable that the number of instructions involved in a computation be only a fraction of the number of arithmetic steps that must be performed. This requires that the device must be capable of repeating sequences of instructions either a given number of times, or a sufficient number of times, so that some prescribed condition involving the computed quantities can be satisfied. To carry out a computation in an efficient manner it may be necessary to do other non-arithmetic operations. For example, it may be expedient to rearrange a number of items in order of size; or it may be necessary to store many items in a compact manner. Several items may need to be stored in one memory position and the device must be capable of selecting individual items in such an instance.

0.2. A substantial number of calculating machines have been built which are capable of doing automatically all of the things listed above. Although these machines perform elaborate computations involving even billions of steps , it is first necessary to spell out in fine detail all the steps in the process and establish the pattern for carrying out these steps. To illustrate this point consider the National Bureau of Standards Western Automatic Computer (otherwise referred to as the SWAC). This machine is capable of doing nearly a million arithmetic operations per minute (actually 937,500) where each such operation might consist of adding any two numbers in its memory and placing the answer in any other position of the memory. On the other hand it carries out only seven basic operations which are listed in the following table.

Addition	Extract
Subtraction	Input
Compare	Output
Multiplication	

Table 1. SWAC Commands

Some of these operations are self-explanatory. Compare is used to control the pattern of the computation, that is, to count processes or to stop a cycle when a variable reaches a prescribed value. Extract is used to split numbers into parts. In detail, the command structure is more complicated than the above indicates. For instance, there are two types of multiplication, rounded or fractional and integral, input and output provide for communication with various input and output devices, as well as with a magnetic drum auxiliary memory, and extract also provides for multiplication by powers of 2 (the device uses the binary number system).

0.3. In order for a computation to be performed on a calculator such as the SWAC the whole process must be written out in precise detail in terms of the language of the calculator, that is, in terms of the commands that the device can execute. After such a machine has been in use for some time certain patterns of commands useful in many different problems are used as units (these are called sub-routines and from the point of view of the user the language of the device has been extended. At the present time most automatic calculator installations have heirarchial systems of subroutines available to the user. These make it possible to operate the device as if it had a floating point arithmetic system, or even was capable of complex arithmetic; vector, matrix, interpolation, or integration processes, for example, are called into play by use of an efficient abbreviated coded instruction system.

0.4. The drawback in the above modes of operation is the relatively long time consumed by the calculator in the "bookkeeping" processes associated with the problem. Thus, the device is much slower in these modes of operation than the basic arithmetic speed would indicate.

0.5. Calculators which have been designed more recently have incorporated a few commands which make some of the above operations more efficient. For example, commands to facilitate repetition of processes, means to improve the processing of quantities involving indices, and means to aid in the organizing of heirarchies of subroutines are present in some of the more recently designed calculators.

0.6. This presentation considers the task of problem preparation from the point of view of what is normally used to specify a computation instead of from the point of view of what is the next step to improve present-day calculators. Inspection of some sample problems provides background for the system worked out in subsequent pages. Following the sample problems some basic principles are discussed.

EXAMPLES OF PROBLEM SPECIFICATION.

0.7. In this section a few problems will be given. They may not be a representative sample, but they should serve to indicate some of the problems which arise in designing a calculating system.

0.8. Problem 1.

$$z = 1(1)95,$$
$$y = 0.05(0.05)7.00,$$
$$x = z/137,$$
$$s = (1-y^2)^{\frac{1}{2}} - 1,$$
$$U = (1+y^2)^{\frac{1}{2}}/y,$$
$$f(z,y) = y^{2+2s} e^{\pi xU} \left| \Gamma(1+s+iyU) \right|^2.$$

0.9. Problem 2.

$r = 0(0.1)1.5$,
a_i and m_i $(i = 1,2,3)$ are given constants,

$$f(s) = \sum_i a_i/(1+m_i s^2)^2, \quad i = 1(1)3,$$

$$D(r) = (2r/\pi) \int_0^\infty sf(s) \sin rs \, ds.$$

0.10. Problem 3.

$$U(x,o) = U_0,$$
$$U(0,t) = U_1,$$
$$U_t = kU_{xx} + e^{-1/U}, \quad 0 \le x \le x_0, \quad 0 \le t \le t_0,$$
where U_0, U_1, and k are given constants.

0.11. Problem 4.

$r = -1+(0.05)1-$,
$h,k = 0(0.1)2.6$,

$$L(h,k,r) = 1/2\pi(1-r^2)^{\frac{1}{2}} \int_h^\infty \int_k^\infty \exp\left[-(x^2+y^2-2rxy)/2(1-r^2)\right] dx \, dy$$

0.12. Problem 5.

$$x'' = (g/w)(T-0.0682\, C_D\, rV^2) \cos Z,$$
$$y'' = (g/w)(T-0.0682\, C_D\, rV^2) \sin Z - g,$$
$$\tan Z = y'/x',$$
$$V = (x'^2 + y'^2)^{\frac{1}{2}},$$
g and T are given constants and
$w(t)$, $C_D(V)$, $r(y)$, are given in tabular form.
Trajectories are desired for 11 sets of initial conditions.

0.13. Problem 6. Given two matrices (a_{ij}) and (b_{ij}), find the product (c_{ij}). Or, find (c_{ij}) such that

$$c_{ij} = \sum_{k=1}^{n} a_{ik}b_{kj}, \quad i = 1(1)n, \quad j = 1(1)n.$$

0.14. Problem 7. Find the solutions x_j of the set $\sum_{j=1}^{n} a_{ij}x_j = b_i$ where $i = 1(1)n$.

0.15. Except for 6 and 7 these problems are derived from problems described in the Quarterly Progress Reports of the National Bureau of Standards. There is as yet no hope of designing a system which will solve the above problems as they are stated without manual attention. In most of the problems a method or algorithm must be devised which will produce a solution in a reasonable number of steps. The aim here is that no matter what procedure is devised by a numerical analysist, it should be easy to cause the calculator to carry out his procedure.

0.16. In the above examples there is no attempt to give complete specifications. Usually, it is desired that the answers have a given number of significant figures, that sufficient digits be carried at intermediate stages to provide this accuracy, and that if the initial data does not justify answers with the specified precision this fact be so indicated. Usually the number of digits carried at intermediate steps is not just a sufficient number to produce the required accuracy, but depends upon the word length as used in the calculator involved. Since multiple precision operations in most existing computers slow down the computation by substantial factors, the most frequent practice in floating point techniques is to attempt single precision, retain the maximum number of figures at each step (significant or not), and hope that the answers mean something. Clearly, if word length could be changed from problem to problem an advantage would be gained in terms of memory capacity and perhaps in arithmetic speed.

0.17. Most automatic calculators use typewriter or tabulator output and usually there is no provision for exponents or subscripts. The input equipment has the same limitations. Unless something is done about terminal equipment, the problem must be specified in terms of symbols which can be handled by the input equipment. That is, in most cases it must consist of the usual typewriter symbols (preferably lower case) plus what can be obtained on a few extra or special keys.

1. GENERAL PRINCIPLES

1.1. The design of an optimum system necessarily depends upon knowing precisely what the system is to do. If the coding problem is to be eliminated then the system must be very complex, not only to handle the great number of variations possible in the expression of a problem but also to accomodate the vagaries of the individuals who prepare problem specifications. The ideal is to have as few rules as possible for the person who writes problem specifications; on the other hand, in order to confine the system to a reasonable size, a number of restrictions will need to be imposed.

1.2. Minimize redundancy.

This principle is based on the expectation of having a reliable calculator and that the giving of the problem to the device is essentially a copying process. If manual transcription of input data is involved the possibility of duplication for checking purposes is not ruled out, however, only one set of data is finally inserted into the computing device.

1.3. In present-day computers the formula a + 2b illustrates a redundancy problem. The "2" must be assigned to a storage location register which is usually capable of storing ten such digits. Furthermore, whenever this number is to be used an address (3 to 5 more digits) must be used to specify it.

1.4. Use of typewriter symbols.

This makes it possible to prepare the specification of problems at other locations rather than with the problem preparation equipment associated with the calculator. As illustrated in the above examples, it is advantageous to have a so-called mathematical typewriter with some of the more common mathematical symbols on special keys.

1.5. Length of addresses.

Of all the storage locations corresponding to addresses appearing in the code for a problem, some will be used most frequently, some less frequently, and so on, until there is a class each member of which is used only once. Therefore, addresses that are used most frequently should be represented by a single character, those next most frequently by two characters, and so forth.

1.6. On the other hand, the fact that information is handled in blocks represented by single digits (or perhaps by alphanumeric characters) encourages the division of the items to be handled into categories, those to be represented by one character, those by two, and so forth. In many cases there are several items of a particular type which are represented in the formula by characters with subscripts. These same characters with subscripts should be usable as the address of the corresponding item in the memory of the calculator.

1.7. Coded change of machine structure.

If the system has sufficient facilities to be efficient on a large class of problems then the amount of information required to specify any given state of the device must necessarily be large. On the other hand, a given problem will use some of the features of the calculator a lot, other features to a lesser extent, and some features perhaps not at all. In particular, while evaluating a formula all arithmetic operations will most likely be done in floating point arithmetic, whereas any tallying operations (which are still required even with B or index register systems) would need to be done in simpler and faster fixed point arithmetic.

1.8. Thus, formulas could be introduced by a particular symbol such as f. This would cause the single characters standing for the basic mathematical operations to thereafter be interpreted as specifying floating point operations. A combination like df could indicate that the operations were to be done in double precision floating point arithmetic. A letter such as i at the beginning of a formula might indicate that these same single characters denoting the basic arithmetic operations are to be interpreted as fixed point (integral) operations. This can be carried a great deal farther. For example, the same arithmetic symbols might indicate complex operations, vector operations (the number of components having previously been specified), or even matrix operations. How far one goes in this direction depends on (1) the class of problems expected, (2) the assignment of meanings to symbols, (3) the memory space available in the calculator in which to store what to do about each combination of symbols, and (4) the effort to be put into organizing the system.

1.9. Scanning the problem.

The type of input equipment available will probably determine whether the problem is to be scanned or not. With magnetic tape there is little difficulty in scanning the problem for preliminary information. If the problem is on punched cards then it is not so desirable to scan since this requires manual handling of the cards.

1.10. On the other hand, the information gained from scanning makes it possible to accomplish essentially the same result with a less complicated system.

1.11. For problems of substantial size and with magnetic tape input the calculator can scan the whole problem, calculate word lengths, and assign storage space more reliably than a person can.

1.12. Notation used in the problem specification.

The most important requirement is that there be no ambiguity. For example, if a/bc appears there must be no doubt about whether it means a/(bc) or (a/b)c. In a new field such as this there is a temptation to introduce new notations specifically designed for the job; however, it must no be forgotten that conventional notations have stood the test of time and since people are familiar with them the pedagogical problem is of less consequence. On the other hand there is no particular reason to allow the person who prepares the problem specification to write both x = 4 and 4 = x indiscriminantly. With some additional complexity the machine can cope with one such pair of statements, but if the cal-

culator must accept various ways of writing many types of statements then too large a price is paid in terms of memory space and speed of operation.

1.13. Thus, the aim here will be to use problem specifications which are compact, clear, and non-redundant.

1.14. With these principles in mind the next step is to define some terms which are used in specific senses and then to present the problem preparation instructions or coding manual for the calculator.

2. DEFINITIONS AND NOTATION.

2.1. The items appearing in a problem specification are classified into a number of different groups. First the operational or connective symbols are classified into four groups, arithmetic, conditional, indicial, and terminal. The letters used in the problem specification will be classified as constants, data, indices, parameters, variables, specials, and temporaries. Indices will be further classified as regular and irregular. Each of these quantities shall now be defined.

2.2. Address. The memory stores characters. The character positions are numbered from one to the integer corresponding to the last character position. The integer corresponding to a character position will be called the address of any character placed in that position. The location of the initial character (usually a + or -) of an item will be referred to as the address of that item.

2.2.1. Address increment. That number which must be added to the address of an item of a class to find the address of the next item in that class is called the address increment. Where unambiguous this term will be shortened to increment.

2.2.2. Basic address. The address of the first item of a class of items will be called the basic address of the class.

2.3. Constants. These are numbers which are used in the computational process and which are invariant with respect to change of initial conditions (conditions given at the start of the problem).

2.4. Data. These are numbers involved in the computation but which may be changed from time to time (each time the problem is run, for example).

2.5. The equality symbol will be used with the meaning "replaces". For example, a=b means the value a is substituted for the value b.

2.6. Index. An index is a letter used to denote the elements of a sequence of consecutive integers where each integer corresponds to an element in a sequence of items. Indices are used for one or two dimensional arrays of items, for example a_{ij}.

2.6.1. First order indices. The i in a_i or a_{ij} will be referred to as a first order index.

2.6.2. Second order indices. The k in a_{jk} will be referred to as a second order index.

2.6.3. Forward index. A forward index is an integer used to denote an element of a sequence. Unity denotes the first member and successive integers (positive) denote successive members.

2.6.4. Reverse index. Similar to forward index except unity denotes the last member of the sequence (finite) and successive members of the sequence correspond to successive integers in decreasing order.

2.6.5. Marching index. These are used to denote the elements of a sequence as are normal indices. Incrementation of a marching index means to shift all elements so that effectively a given element goes into its predecessors possition. The first is thrown away and zero is inserted for the last element.

2.7. Marker. Items will have their addresses indicated in the memory by use of markers. The marker for items will be either + or - and it will be in the address position of the corresponding item. Classes of items will be terminated with markers of various orders.

2.7.1. Page markers. A sequence of items may be terminated by a page marker which is the symbol , (comma).

2.7.2. Chapter markers. A sequence of items or pages may be terminated by a chapter marker which is ; (semi-colon).

2.7.3. Book markers. A sequence of items, pages, or chapters, may be terminated by a book marker : (colon). A period marks the end of the problem.

2.8. Number. The term number as used here will mean a signed decimal digit number used with the following conventions: With no decimal point the number will be an integer, with the decimal point preceding the number it will be a pure fraction, and with the decimal point in an intermediate position the digits before the decimal will stand f

the power (positive or negative) of ten which multiplies the number and the figures after the decimal point are the significant figures in fractional form. Exponents may be in complementary form i.e. say, those above 50 are negative. Numbers may be of any length.

2.9. Operational symbols. The operational symbols include the arithmetic, indicial, conditional, and terminal symbols as defined below.

2.9.1. Arithmetic symbols. The arithmetic symbols are +,-, ·, and /.

2.9.2. Conditional symbols. The conditional symbols are =, ≠, <, ≤, >, and ≥.

2.9.3. Indicial symbols. These are (,), and ↑.

2.9.4. Terminal symbols. These are periods, commas, semi-colons, and colons.

2.10. Parameter. A parameter is a quantity which takes on a sequence of values and for each value of the parameter the corresponding portion of the formula is to be evaluated. Parameters are further classified as follows:

2.10.1. Regular parameter. Such a parameter changes from an initial value to a final value in equal increments.

2.10.2. Irregular parameters. All other parameters are called irregular.

2.11. Program loop. A program loop is a sequence of operations which is to be repeated until some specified condition is satisfied.

2.12. Section. The problem will be divided into two or more sections. The first (number one) is the preparation section. The other sections are numbered in order.

2.13. Temporaries. These are quantities which appear in only one section of the problem.

2.14. Variables. Generally, these are quantities which change in value during the course of the computation. (Constants, data, variables, temporaries and specials are handled in the same way in the calculator).

2.15. The Operational Block. Formulas are processed in units extending from one operational symbol to the next. The operational symbols need not be actually present; for example, in the formula a + 2b the implied multiplication between the 2 and the b terminates one block and initiates another. In this system parenthesis are not allowed in the formulas, only in the parameter specifications. Because of this and the principle that the problem specification should be typeable on

a typewriter, powers and functions will be indicated by special symbols. If a quantity is to be raised to a power it will be followed by the symbol p and then the exponent (either in numerical form or as the contents of an address). Functions will be indicated by the letter f followed by a subscript to identify particular functions. In this presentation functions will be required to be functions of a single variable.

2.16. The following table gives the general structure of the operation block:

Code	Name
PO	Present operation
FN	Function
FE	Function exponent
AN	Address or number
AE	Address or number exponent
FO	Future operation

Table 2. Operationnal Block.

2.17. In practice many of these components may be missing in a particular formula. For example, a+b= -3c is a possible formula with the underlined portion representing an operational block, or $+\sin^{3.14159}(x_{ij})^{v1}$-... can be handled in the form +f3p3.14159x1jpv1-... where f3 is the functionnal subroutine that computes the sine.

2.18. Functions.

In this presentation only functions of a single variable will be admitted in formulae. Functions of sequences or of several variables must be handled by subroutines. The limit on the types of functions of a single variable depends only on the number of subroutines that are available to the calculator.

2.19. Certain combinations of functions may be allowed in the formulae. To illustrate this the following arrangement has been chosen:

$$f^a(x^b)$$

where a and b are any numbers which the current arithmetic commands of the system can handle.

2.20. It will be assumed that all the subroutines in existence are available to the calculator (perhaps stored on magnetic tape) and that they are each identified by a code of the form f_i where $i=1,2,3,\ldots i_m$.

Each problem will generally use relatively few of the subroutines, so, those subroutines which are expected to be used most often should correspond to the fewest symbols of 1.

2.21. Assignment of meaning to symbols.

In order for the calculator to process the formula it is expeditious to classify the address symbols that will be used in the problem. Since certain types of symbols will occur most frequently in problems this assignment is done in terms of a basic standard assignment. For each problem it is only necessary to list the exceptions to the basic assignment. The following table gives the standard assignment.

Letters	Category	Meaning
a to e	c	constants
f	f	function
g	g	reverse marching index
h	h	reverse normal index
i to l	k	forward normal indices
m,n	m	forward marching index
o		not assigned
p	p	power
q	i	irregular parameter
r	r	regular parameter
s	s	problem section location
t to z	y	variables (include temporaries)

Table 3. Standard Assignment.

2.22. Any of the letters can be reassigned into any class with the exception of p, f, and s.

2.23. Reassignment is done by writing at the beginning of the problem specification a statement such as r:g, which means that from there on r will be a reverse marching index instead of a regular parameter.

2.24. Problem Section Markers.

The use of markers to partition data relating to the problem has been mentioned above. Markers may be used in the same way in the problem specification. On occasion it may be necessary to shift attention (transfer control) to a distant part of the problem specification, and the marker system is not an efficient way to do this job. For this purpose, and for general presentation purposes major parts (sections) of the

problem specification should be numbered using positive integers starting with unity, each such number being preceded by the character s and followed by a : (colon).

3. CODING MANUAL.

3.1. There is a set of standard operations that can be performed using the combinations of symbols discussed in this coding manual. These operations make it possible to program a large class of computational problems. If the problem is too complicated (for example, by involving three dimensional arrays of items such as a_{ijk}) then it will be necessary to study the structure of the system in order to produce a correct program for the problem. By making use of the structure of the system it is possible to program any sequence of operations which can be done on present-day computers. Generally, even this process will be more efficient since the subroutines that may be involved will themselves be written in terms of the extended language of the calculator.

3.2. There follows a general discussion of the structure of the code for the problem as a whole. This is followed by a more detailed discussion of the preparation section. Finally there is a discussion of the possibilities that are available in the other sections of the code.

3.3. General structure of the code.

The code consists of a sequence of operational blocks (see 2.6). The operational blocks are broken up into sections which are numbered as s1, s2, s3,... . The following table indicates the order in which the blocks must be arranged.

	Block example	Remarks
1.	s1:	Initial clear, marks the beginning of the preparation section.
2.	s5,	If there are more than two sections the letter s followed by one less than the number of sections must be placed here.
	,a:n,	Next must follow any reassignment of characters. This block means that a is to be a normal forward index.
	,a:n18,	Simultaneous with reassignment the maximum range in case of categories (see 2.11) g,h,k, and n and number of digits in all other categories may be specified.
3.	,a8ij,	A standard association of indices must be made so that such expressions as a5 7 (a_{57}) will be interpreted properly. The number of digits in each a is specified immediately following the character. All letters to be used in the problem specification must appear here in this preparation section so that the calculator will know how much space to allow for each item.
	,m9jk:v,	A character may be reassigned, indices noted, and the number of digits specified at once. There is no provision to handle indices which themselves carry subscripts.
4.	m4j:d= +3-56...	Finally, all initial data must be given, and this can follow directly after reassignment, index specification, and word length. Note that here the maximum word length is 4(a + or - and as many as three digits) but the various items may be shorter. In case of two dimensional arrays the info is given in the order a1 1, a1 2, a1 3,..., a2 1, a2 2,... .
5.	,qfi=....	The values of irregular parameters are given next. The subscript i and the length of each item (f) are also given here.
6.	,r3=0(0.2)10,	The range and increment for regular parameters are also given in this part.
7.	s2:	This marks the end of the preparation section and the beginning of the actual formula.
8.	...+a/...	Next follows the operational blocks of the formula. Any combination of the arithmetic symbols and terminal symbols along with the conditional symbol = (see 2.9) may be used here.
9.	,i;	Next there occurs indicial information, parameter changes, and conditional or unconditional transfer of control. The possibilities here will be discussed in subsequent sections.
10.	s3:	As convenient and to provide for transfer of control, section markers may be inserted as needed.
11.	p:a1,i.	A period marks the end of the problem. Usually this will be preceded by a print instruction. The example says print the values a, for each value of i. Format control can be incorporated by use of subroutines.

Table 4. Code Structure

3.4. The Formula.

After the preliminary specification the section marker s2: appears. Next the formula is written using the various operational symbols and function and power possibilities that have been listed above. In place of parenthesis new variables must be introduced. a/bc will mean a/(bc) whereas a/b+c will mean (a/b) + c.

3.5. If a quantity has a single numerical subscript it is written as a11, ai11, a11i, ai11k, etc. If a quantity carries to consecutive numerical subscripts they are to be separated by a space, a11 9, for example.

3.6. The Use of Markers to Terminate Program Loops.

In this system the expression ,i; will be interpreted to mean increment the index i and return to the beginning of the current segment as indicated by the presence of a semi-colon. If at anytime in the process a marker of second or higher order is picked up as an operand, then terminate the process, reset the index, and continue with the formula.

3.7. In case of the combination ,i;j; process the formula for values of i as above, increase j and repeat the process for all calues of i, etc. Increasing j by unity means move on to the next page, chapter, or book depending upon the order of j. Counting with respect to j is terminated by the appearance of a higher order marker than those which normally terminate the process with respect to i.

3.8. Marching Processes.

In the step by step solution of differential equations frequently a new value of a variable is computed in terms of two or more preceding values. All the successive values are not to be printed out but only those at particular points. This marching process is illustrated by the following:

$$x_{i+1} = f(x_i, x_{i-1}, x_{i-2}, \ldots, x_{i-k})$$

and the process of advancing one step can be indicated by i+1=i. To take this step it is necessary that the following transfers be made:

$$x_{i-k+1} = x_{i-k}$$

$$\ldots$$

$$x_i = x_{i-1}$$

$$x_{i+1} = x_i$$

A marching parameter is used to control this stepping process.

3.9. For example, the Newton-Raphson process for solving algebraic equations gives

$$x_{h+1} = x_h - f(x_h)/f'(x_h),$$

$$\left| x_{h+1} - x_h \right| \epsilon >h+1 = h;$$

As long as the difference of two successive approximations exceeds a limit, the x's will be shifted down and the process repeated. Since in the standard assignment it is already known that h is a marching index the above problem statement can be condensed as follows. First in section 1 the symbols ,xh, must appear indicating that the index h is associated with the variable x; then the above can be rewritten as

$$;x1 = x0 - f(x0)/f'(x0),\ z=x1 - x0,\ h,\ f2z < \epsilon;\quad .$$

where f2 denotes the subroutine which takes the absolute value.

3.10. Transfer of Control and Index Conventions.

The following table indicates the possibilities for transferring of control, and incrementation of indices.

	Operational Block	Effect
1.	, ---- ,	return to the preceding comma
2.	, ---- ;	return to the preceding semi-colon
3.	, ---- :	return to the preceding colon
4.	, ---- k, , ---- k; , ---- k:	go to section k, k is an integer
5.	,nT	return to the preceding T symbol n times and then continue, n is an integer.
6.	,n →kT	go to section k n times and then continue, n is an integer.
7.	,cT	return to the preceding T symbol (c) times where (c) is the contents of address c.
8.	,c→kT	similar to 6 and 7
9.	,i; ,i:	increase i by its increment and return to the preceding ; or :
10.	,i,j; ,i,j:	increase i and j by their respective increments and return to the preceding ; or :
11.	,i;j;	increase i and return to the preceding semi-colon, when a comma marker is reached in the data reset i and increase j and return to the preceding semi-colon. When a semi-colon marker is reached in the data reset i and j and continue.
12.	... ⩾ xT	if the contents of A exceeds x, or (x) if x is an address (letter), return to the preceding T symbol, otherwise continue.
13.	... < xT	if the contents of A are less than x or (x), as the case may be, return to the preceding T symbol, otherwise continue.
14.	... ⩾ x →kT	if the contents of A exceeds x or (x) go to section k, otherwise continue.
15.	... < x →kT	if the contents of A is less than x or (x) go to section k, otherwise continue.
16.	,h,	shift all the quantities with subscript h one position (march) and continue (see 3.8).
17.	,h; or ,h:	as in 16 but return to the preceding ; or :

Table 5. Transfer of Control.

3.11. Examples of problem specifications.

Some simple examples are now given to show how problems must be stated in order to be solved with the system described here.

3.12. Example 1. To solve the quadratic equation $ax^2+bx+c = 0$.

Problem statement:

s1: a = +01.37658, b= +99.88072, c= +00.31549, i2, x8i;

s2: z = bp2 - 4ac, xi = -b/2a - 1pizp0.5/2a, i.

In this example a new variable has to be introduced in order to take care of the expression $(b^2-4ac)^{\frac{1}{2}}$. According to the numerical convention (see 2.8) the true values of the numbers a,b, and c are obtained by taking the fractional parts and multiplying by 10^1, 10^{-1}, and 10^0, respectively.

3.13. Example 2. Evaluation of a polynomial.

s1: h11, a8h = +02.7734-01.0993+...etc. ...;

s2: ah + bx = b, h.

Upon execution of this program the iterative formula is repeated until the marker is found at the beginning of the a's.

3.14. Example 3. Vector-matrix multiplication $\sum_{i=1}^{20} a_{ij}b_i = c_j$, $j = 1,2,3,\ldots,20$.

s1: i20, j20, a7ij = ...given elements row by row with a comma at the end of each row except the last...; b7i = ... given elements..., c7j;

s2: cj + aijbi = cj, i; j.

3.15. In the system considered formulas must not have more than two indices. Accommodation of more indices, indices with exceptional values, of parenthesis in formulas, etc., requires more complexity in the processing system. It is thought that a more complex system capable of handling more general formulas than those illustrated above is a practical possibility in view of the present state of the computer art.