ALGAE I*

A Compiler for the IBM 704†

by

Edward A. Voorhees
Glenn L. Carter
Jeanne Hudgins
Chester S. Kazek
Karl G. Balke

University of California
Los Alamos Scientific Laboratory
Los Alamos, New Mexico

May 26, 1958

Revised - July 21, 1958

*This work performed under the auspices of the U. S. Atomic Energy Commission.

†This version is for use with a machine having: 8.192 words of core storage, 4 logical drums, 7 tape units, CAD instruction. A peripheral tape-to-printer or simulator is required.

1. Introduction

Discussions involving the subject of defining problems for interpretation and coding by known automatic-coding systems generally suggest that the techniques for stating the control (or logic) of the problem are frequently difficult to understand and difficult to use. It seems that the difficulty is one of discovering a suitable language with which to define problem control. In programming a problem for hand coding, the familiar flow diagram has been successfully used (when needed) to define the control of the problem. Such flow diagrams (as we draw them) cannot be presented directly to present-day computers. It is the purpose of this paper to propose a flow diagram representation using simple algebraic language which can be directly entered into the computer and to describe a compiler which accepts problems coded in this form.

The first part of this description concerns itself with the development of an idea which was earlier presented in a paper entitled "Algebraic Formulation of Flow Diagrams". The second section describes a compiler for the IBM 704 (called ALGAE I) which incorporates most of the techniques described in the first part. ALGAE I makes use of FORTRAN I to form the actual 704 code.

ALGAE I was written as a means to permit programmers to use and evaluate this idea of program control. As a result, any comments, criticisms, etc., from users of the system are not only welcome, but eagerly solicited.

2. General Description of the Algae Language.

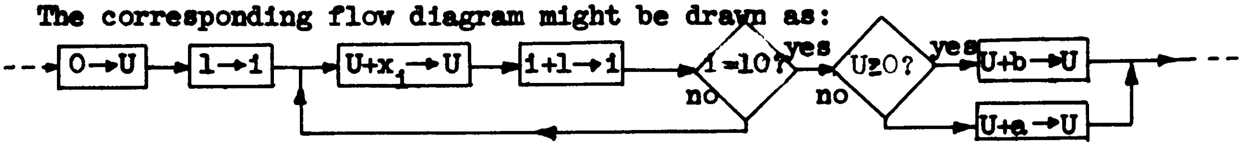
2.1 Basic Concepts.

Assume we have a flow diagram such that it could be used to hand-code a problem for any stored-program computer. Suppose we remove from the boxes all equations, statements of input-output tasks and other statements not directly related to the control and logic of the problem, and list them, with identification, elsewhere as reference material. We do not include in this list statements and questions pertaining to loops, numerical conditions, or switch and trigger conditions. The items remaining in the flow-diagram would form a statement or "part-picture" of the control for the problem. We will attempt to translate this control statement into a statement which can be easily written and entered into a computing machine.

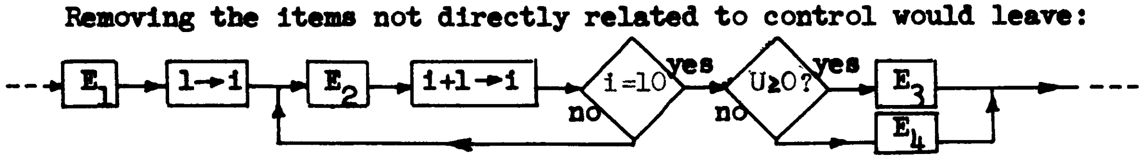
In order to illustrate the above discussion and extend the idea further, let us consider an elementary example. Suppose in a problem we wished to compute

$$U = \sum_{i=1}^{10} x_i + \left\{ \begin{matrix} a \ (\text{if } \Sigma < 0) \\ \text{or} \\ b \ (\text{if } \Sigma \geq 0) \end{matrix} \right\}$$

The corresponding flow diagram might be drawn as:



Removing the items not directly related to control would leave:



If a distinction is now made between loop ranges, such as the i-loop in the above example, and conditions for execution of equations, such as the U test above, we introduce the symbols

I_1 : $i = 1(1)10$ (meaning I takes successively the values 1 through 10 in increments of 1)

and C_1 : $U \geq 0$.

Our example would then be completely defined as follows:

Example 1

Control	Equations
$\ldots + E_1 + I_1E_2 + C_1(E_3,E_4) + \ldots$	E_1 : $U = 0$
I_1 : $i = 1(1)10$	E_2 : $U = U + x_i$
C_1 : $U \geq 0$	E_3 : $U = U + b$
	E_4 : $U = U + a$

where C_1 here is understood to mean " execute E_3 if C_1 is satisfied and execute E_4 if C_1 is not satisfied." If we were to define C_1^* as the negation of the condition C_1 , the last term in the control statement could be written as

$$\ldots + C_1E_3 + C_1^*E_4 + \ldots$$

provided it is known that when E_3 is executed E_4 would not also be executed (as would be the case if $\sum x_i = 1$ and $b = -2$). Clearly the C^* convention is not essential since the condition C_2 : $U < 0$ could be used instead. The meaning of I_1E_2 is evident and will be discussed more generally below.

2.2 Definition of Symbols.

Let us now define more completely the set of control statement symbols. These definitions make no restrictions on the characters with regard to their use in equation writing since control statements are assumed to be handled separately from equations.

C (assumed to have a subscript) represents a single condition for a two-way branch. Several conventions, defined in section 2.5, have been developed which use logical combinations of C's. C's are used for stating all conditions, except those inherent in range statements such as end-of-loop tests. C-type conditions include tests of numerical, logical,console switch, and trigger conditions. A more generalized, n-way branch is under development. Once a C has been defined, it may be used repeatedly throughout the problem.

E (assumed to have a subscript) represents a single equation, a continuous and closed set of equations (whenever one is done, all are done), an input-output task, program stop, or any other statement or closed set of statements not directly related to problem control. An E_i may also be used more than once.

G and H[†] (subscripted). This convention is designed to facilitate transfer of control. The symbol G (with a subscript) denotes a transfer to H (with the same subscript). There may be a many-to-one correspondence between G and H.

I,J,K,L,M,N (subscripted) represent single range statements or loop definitions. I_1 in Example 1 represents the range statement $i = 1(1)10$. More generally, the expression $I_j : a = b(c)d$ means that I_j defines a loop in which the subscript a varies from b to d in steps of c. These may be used repeatedly.

S (subscripted) represents a control statement. It allows the program control to be defined by many sub-control statements which are in turn connected by a single master control statement. An S_i , once defined, may be used only one time. This partitioning of the code into smaller units of control tends to clarify the various phases of the problem.

T (subscripted) is basically an S which may be used repeatedly within a problem. It is the Algebraic representation of a closed subroutine. Both the S_i and T_i can appear in the same positions as an E_i.

The plus sign,(+), has two meanings. The first is to indicate a direct flow of control , as, for example, $E_1 + E_2$, which means "execute E_1 , then E_2." E_1 and E_2 are terms of the control statement $E_1 + E_2$. A second meaning, when used with C's, is discussed in section 2.5.

The parentheses,(), are used for the phrasing or grouping of terms, for indicating ranges, and for special purposes to be described. The expression $(E_1 + E_2)$ is a single term.

The comma is used in special conventions, as described below.

[†] This set of symbols has not been shown to be essential to the system. It serves more as a convenience to the coder and may introduce some undesirable flexibilities unless used with caution.

2.3 The Basic Forms.

The symbols defined above may appear in seven basic forms. Section 2.4 sets forth rules to be used in combining these basic forms to symbolize more complex logical conditions.

Let us define the quantity X as any single subscripted E, S, or T. The basic forms may be written as:

1. X — The single quantity. In this case X may also be a subscripted G or H.

2. I_jX — Execute X loopwise for the range of the subscript indicated in the definition of I_j.

3. C_iX — Execute X only if the condition C_i is satisfied. In this form, X may be the subscripted transfer symbol G_i, resulting in a conditional transfer.

4. XC_i — Execute X, test C_i; if C_i is not satisfied, execute X again, test C_i and continue iterating until C_i is satisfied.

5. $C_j(X_a,X_b)$ — If C_j is satisfied, do X_a and skip X_b. If C_j is not satisfied, skip X_a and execute X_b. X_a and/or X_b may be the subscripted symbol G_i in this form.

6. $(X_a,X_b)C_j$ — Execute X_b, test C_j. If C_j is not satisfied, do X_a and X_b, test C_j and iterate until C_j is satisfied. In a control sense, this is an iteration loop in which the statement, X_a, is executed in every iteration except the first.

7. $I_j(X)C_k$ — After each cycle of the I_jX loop, test C_k and if satisfied, leave the loop and calculate the next term in the control statement. If C_k is not satisfied, do the next cycle, test C_k again, etc. Exit from the I_jX loop is made as in form 2 unless C_k is satisfied earlier.

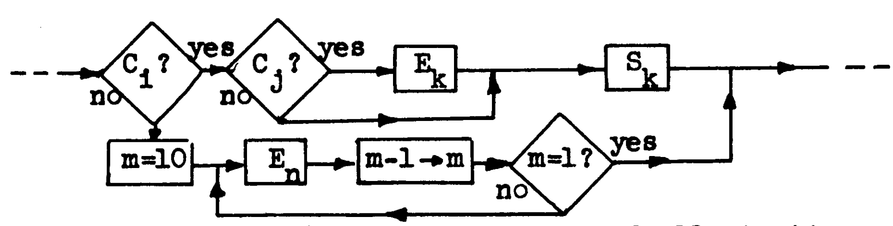
In each of the above forms involving I_j, any other range statement could be used (N_k, L_i, etc.). Note that all characters must be subscripted.

2.4 Combinations.

More complex operations may be expressed by combinations of the above forms in which the following rules are applied.

1. Any basic form may be combined with another by using the "continuation symbol" or plus sign (+). Thus, $E_i + S_k$ means " execute the equations in E_i as defined, then perform the sequence of operations defined by S_k."
2. Any basic form may be placed within another basic form (or within itself) in the position occupied by the symbol X of region 2.3. Thus $I_k(C_jE_i)$, form 3 within form 2, means "for each cycle of the $I_k(...)$ loop, calculate E_i only if C_j is satisfied; otherwise start the next cycle." Or, $I_k(J_iE_k)$, form 2 within itself, means " for each value of the subscript whose range is defined by I_k, execute the loop defined by J_iE_k." This is the familiar loop within a loop.

   Note that the basic form which replaces the X is set off by parentheses to indicate order of performance. Parentheses are not required in forms 5,6, or 7 since the form itself sets off the X's by parentheses and commas.
3. Any meaningful combination created by the above two rules may also be inserted within a basic form in place of an X. This "second-order" combination may be illustrated by the term $C_i(C_jE_k + S_k , I_kE_n)$, forms 3 and 1 combined using rule 1 and placed, with form 2, within form 5. The meaning of this term can best be illustrated by the flow diagram



   where I_k is defined, for convenience of illustration, as m= 10(-1)2 and C_i? asks the question, "Is the C_i condition satisfied?"

Additional compounding may be achieved by repeated use of the above rules.

2.5 C-Statements.

Further substitution may be made in the combinations and basic forms by replacing a single C condition with a C-Statement. Since the C's are defined as binary conditions, a logically arranged set of C's can have the effect of a single C. For this, we use the second definition of the symbol (+) and say that $C_i + C_j$ means C_i or C_j. Similarly, let us define C_iC_j as C_i and C_j. Substituting an expression consisting only of C's into a basic form or combination allows us to express conditions such as $(C_1 + C_2C_3)E_1$, meaning "if C_1 is satisfied or if C_2 and C_3 are both satisfied, execute E_1; otherwise skip E_1 and continue to the next term."

C-statements may also contain negated conditions by using the negation symbol (*). Thus $(C_1 + C_2C_3)^*$ would be the negation of the total situation defined in the above example. More complex statements of this type may be defined by the basic rules of Boolean Algebra.

When substituting a C-statement into a form or combination, parentheses are needed to encompass the statement only if the resulting terms are OR-ed at some point. Thus $C_3C_4 + (C_2+C_5)$ should be enclosed by parentheses as $(C_3C_4 + (C_2 + C_5))$, but the term $C_3C_4(C_2 + C_5)$ need not be.

To illustrate, re-define example 1 as:

$$U = \sum_{i=1}^{10} x_i + \begin{cases} a \text{ (if } \Sigma > 0, \Sigma \leq 3 \text{ and } x_i \geq 4) \\ b \text{ (if } \Sigma > 3 \text{ or } x_i < 4) \\ 0 \text{ otherwise.} \end{cases}$$

We could completely define this as follows:

Example 1a

Control	Equations
$\ldots + E_1 + I_1E_2 + C_1C_2C_3^*(E_4, (C_2^* + C_3)E_3) + \ldots$	$E_1 : U = 0$
$I_1 : i = 1(1)10$	$E_2 : U = U + x_i$
$C_1 : \Sigma > 0$	$E_3 : U = U + b$
$C_2 : \Sigma \leq 3$	$E_4 : U = U + a$
$C_3 : x_i < 4$	

2.6 Sample problem.

Let us now illustrate some of the preceding ideas with another example. Assume we wish to solve the Laplace equation for a 10 x 10 mesh with x = 1 on the boundaries and x = 0 in the interior. Then:

Example 2.

Control

$S_1 : I_1(J_1((C_1 + C_2 + (C_3\ C_4)^*)(E_1, E_2)))$

$S_2 : E_3 + I_2(J_2 E_4)$

$S_3 : S_1 + S_2 C_5 + E_5$

$I_1 : i = 1(1)10$	$C_1 : i = 1$
$I_2 : i = 2(1)9$	$C_2 : i = 10$
$J_1 : j = 1(1)10$	$C_3 : j \geq 2$
$J_2 : j = 2(1)9$	$C_4 : j \leq 9$
	$C_5 : C - 0.001 < 0$

Equations

$E_1 : x_{i,j} = 1$

$E_2 : x_{i,j} = 0$

$E_3 : C = 0$

$E_4 : E = (x_{i-1,j} + x_{i+1,j} + x_{i,j-1} + x_{i,j+1})/4$

$C = C + |E - x_{i,j}|$

$x_{i,j} = E$

E_5 : STOP

S_1 sets up the mesh, $S_2 C_5$ represents the main calculation, and E_5 stops the problem after the iteration is complete. By defining the additional range statements:

$$I_3 : i = 1(9)10 \quad \text{and } J_3 : j = 1(9)10$$

we could set up the mesh using no condition statements whatever:

$S_1 : I_3(J_1 E_1) + I_2(J_3 E_1) + I_2(J_2 E_2)$ which would probably improve the set-up from a time standpoint.

Note that the use of C and E as program variables is not restricted, even though the same symbols are used for control.

3. Algae Programming for the 704.

3.1 Basic Assumptions and Restrictions.

Since Algae I currently uses the Fortran compiler as an intermediate step in its assembly, it is assumed that the programmer has a sound knowledge of basic Fortran concepts and restrictions. In addition, the following rules apply:

1. All subscripts on the logical symbols (E,S,T,G, etc.) must be numerical and greater than zero. Subscripts on non-control symbols are restricted only by the Fortran regulations.
2. Formula numbers can be assigned by the Algae programmer only to FORMAT statements, and must lie within the range $1 \leq FN \leq 99$. All other formula numbering will be done by the Algae code under one of two options selected by the programmer.
3. Because of restriction 2, FREQUENCY statements cannot be used in Algae I.
4. Comments can appear only at the beginning of the deck or within the defined E's.
5. Identification punched in columns 73-80 will be reproduced on Algae I listings only if the cards are read onto tape by the off-line equipment.

3.2 Coding Conventions and Specific Restrictions.

The Fortran coding form is used to prepare the problem for keypunching. Specific instructions for defining each character are given below with the restrictions governing the use of this character in Algae I. Unless otherwise defined, subscripts must lie in the range $1 \leq i \leq 999$. There may be no more than 500 E's, each having 70 cards or less.

E_i Each E_i is a list, in Fortran language, of the "Things to be done" by one flow diagram "box" or closed set of equations. None of the statements listed within an E are tested by Algae I for accuracy. Rather, Algae I substitutes the entire set of equations into the sequence of Fortran statements whenever the specific E is requested by a control statement.

E's may contain comments at any point (they may consist entirely of comments), input-output statements, and/or the two control statements SENSE LIGHT i and PAUSE.* These are

*The control statement STOP should not be used, since it results in a logic error in FORTRAN.

the only control statements listed in chapter 4 of the Fortran manual that should ever be used by the Algae programmer. E's cannot contain FORMAT, DIMENSION, or EQUIVALENCE statements, and must be less than 70 cards in length. Multiple reference may be made to defined E's, as described in section 2.2.

Page 14 illustrates the method of defining E's. Note that the definition of E_{32} is terminated by the appearance of another identification symbol, E_{16}. E_{16} is in turn terminated by the symbol C_3. Fortran equations within an E may be continued from one card to the next as in normal Fortran notation.

C_1 There are two types of conditions in the Algae I program: Type I, illustrated by C_3 and C_5 on page 14 defines the test of a numerical condition. The term or expression within parentheses is compared to zero in a manner designated by the symbol appearing at the right of the parenthesis. If we denote the contents of the parenthesis by X,

(X) P asks, "Is X positive?" $(X > 0?)$

(X) N asks, "Is X negative?" $(X < 0?)$

(X) = asks, "Is X equal to zero?" $(X = 0?)$.

The symbols may be negated to obtain the remaining inequalities:

(X) P^*, "Is $X \leq 0$?"

(X) N^*, "Is $X \geq 0$?"

(X) $=^*$, "Is $X \neq 0$?"

In Algae I, the numerical condition (including parentheses) must occupy at most 48 card columns.

Type II conditions, illustrated by C_1, C_2, and C_4 on page 14, define the test of a trigger, sense switch, or sense light. It is distinguished from a type I condition by the absense of a left parenthesis as the initial non-blank character. A comparator is also used to indicate whether the condition is satisfied when the trigger is on or off.

Typical type II conditions are:

SENSE LIGHT 3 ON	S L 3 ON
SENSE SWITCH 6 OFF	SW 6 OFF
ACCUMULATOR OVERFLOW ON	A ON
DIVIDE CHECK ON	D ON
QUOTIENT OVERFLOW OFF	Q OFF

For the benefit of programmers who are in a hurry or dislike spelling, Algae I will accept the abbreviations at the right in the above listing. An S, A, D, or Q must appear in the first non-blank position to avoid an error, and the ON/OFF condition must be specified. Do not attempt to abbreviate the SENSE LIGHT i instruction when it is used within an E to turn a specific light on.

A C_i of the two above types must have a numerical subscript within the range $1 \leq i \leq 149$.

<u>C-Statements</u>, illustrated by C_{153} on page 14 and defined in section 1.5, enable the programmer to use repeatedly a complex condition with a minimum of writing. A C-Statement must have as its identification a numerically subscripted C_i with $150 \leq i \leq 200$. In Algae I, <u>a C-Statement cannot contain another C-Statement</u>, and must be complete on one card (no continuations).

A C Group has been defined as any closed grouping of conditions, including condition statements. Thus if a statement contains the expression $\ldots + E_3C_4 + C_1(C_2+C_3)^*C_{155}E_2 + \ldots$, C_4 and $C_1(C_2+C_3)^*C_{155}$ are C groups. The maximum number of C's which may appear in a C group is 70. In the second group above, this would include all C's within the statement C_{155} but not C_{155} itself.

<u>I_i- N_i</u>: Range statements (loop conditions) are identified by the numerically subscripted characters I,J,K,L,M,N. Examples of range statements are I_4 and M_2 on page 14. All range statements have the form a=b(c)d where:

- <u>a</u> is the subscript whose range is being defined,
- <u>b</u> is the first value to be taken by <u>a</u>,
- <u>c</u> is the increment of change, and
- <u>d</u> is the final bound on a. The final value of a is $\leq d$ if $c > 0$, or $\geq d$ if $c < 0$.

The following rules apply:

1. a may consist of any four Hollerith characters, the first of which is I,J,K,L,M, or N.
2. b and d may be any non-zero symbols recognized by Fortran as fixed point constants or variables. They can never assume negative values. Note that Fortran prevents the use of subscripted fixed point variables for b and d in DO's.
3. c may be any non-zero symbol (positive or negative) recognized by Fortran as a fixed point constant or variable. If d is greater than b, c must be greater than zero. If b is greater than d, Algae I will set up a reverse DO loop using a dummy variable. In this case, c must be preceded by a minus sign to indicate the reversal of the loop condition. The dummy variable is formed by repeating the first letter of a three times and adding the remainder of the term (see below). Care should be taken to avoid using this variable in the problem.

Some acceptable forms would be:

```
IVAL = IMAX(-IDELTA)IMIN        Dummy Variable:  IIIVAL
J32  = 45 (-3) JLOW             Dummy Variable:  JJJ32
M3K2 = 1 (INCR) 10
L    = L1 (3) 15.
```

It is not necessary for the symbol a to be the same as the identification symbol. For example, in I_4 on page 14 we define the range of the subscript JT. The subscript on the identification must lie in the range $1 \leq i \leq 999$. Only 150 range statements can be defined in a problem. Range statements, once defined, may be used repeatedly.

All 4 fields (a, b, c, and d) must be filled. Algae does not make the assumption that a missing term is equal to 1.

S_1 and T_1 These control statements are written for the 704 in the same manner as those defined in examples 1, 1a, and 2 (see page 14 for further illustrations). Note that all subscripts are numerical and non-zero. Subscripts may be written as smaller characters, but caution should be taken to keep them above the line to avoid keypunching errors. Care should also be taken to make commas distinctively different from the subscript 1.

S's and T's may be continued over several cards without the use of a continuation symbol in column 6, however a statement must have less than 1000 characters (not counting subscripts) and must have any subscripts completed on one line. One problem may contain a maximum of 450 S's and T's combined.

Subscripts on S_i must lie in the range $1 \leq i \leq 999$ but subscripts on T_j are restricted to the range $1 \leq j \leq 100$. To provide the multiple exit to a T sequence, the dummy variable LLxxx is formed, where xxx is the subscript on the defined T. Thus, T_{56} would produce a dummy variable LL56 which should not be used elsewhere in the code.

<u>G_i and H_i</u> These characters need never be defined and should appear only in S or T statements. In both cases the subscript <u>i</u> should lie between $1 \leq i \leq 200$. Violation of this restriction will not produce an error stop, but might lead to a swifter depletion of the allowed quantity of formula numbers.

3.3 Deck Ordering and Operation.

1. Deck ordering.

A problem designed for Algae I assembly may be considered to be divided into two sections. Section I must contain all

1. Control cards (see part 2 below)
2. Initial comments (all other comments must appear within E's)
3. Format statements
4. Dimension statements
5. Equivalence statements.

These cards may be arranged in any order in section I. All cards in section I must be loaded before the first card of section II. Section II consists of all the defined E's, S's, T's, C's and range statements, in any order.

2. Sense switches.

Sense switches 5 and 6 are used by the 8K ALGAE I. They have the following effect:

Sense switch 5:

If this switch is down, and if errors have been detected by the compiler, the FORTRAN code, if any, which has been produced will be listed on tape 9.

Sense switch 6:

If this switch is up, each formula in an E will be assigned a formula number. If the switch is down, formula numbers will be assigned only where necessary to define control of the problem.

3. Preparation of Algae System Tape.

The Algae System Tape is tape # 6. Because of the large number of cards involved, Algae should not be read by the card reader unless the installation does not possess enough tape units to permit the entire system to function from tapes. To write the system tape, place a blank tape on logical tape 6 and load the ALGAE 8K deck in the card reader. Press "LOAD CARDS." Note that it is not necessary to clear the machine or reset the console.

WARNING: DO NOT ATTEMPT TO WRITE TWO TAPES SIMULTANEOUSLY BY TURNING THEM BOTH TO 6. THE WRITING PROGRAM READS THE TAPE IT HAS WRITTEN, AND COMPARES IT WITH THE CONTENTS OF MEMORY. The 704 cannot read 2 tapes at once, and the program will continue to try to write a good tape indefinitely.

4. Operation

Input may be from cards or from a tape prepared on peripheral card-to-tape equipment. Each problem must be loaded separately, whether it is loaded by cards or by tape, as the input tape is destroyed in the process of compiling.

Machine set-up is as follows:

a) Tapes by logical tape number.
   1. FORTRAN
   2. blank
   3. blank or BCD input (see above)
   4. blank
   5. blank
   6. ALGAE
   9. tape-to-printer BCD tape
b) Printer. Share # 2 board.
c) Punch. Blank cards.
d) Card reader.
   1. ALGAE RREC 1 card.
   2. Problem decimal deck, if input is not on tape 3. Otherwise, no cards.
   3. No cards.

To initiate compiling, press the "LOAD CARDS" button. When the deck has been completely read in, and the heading printed on the printer, place ALGAE RREC 2 in the card reader and run it in to ready. If it is desired, the next deck to be compiled may follow RREC 2 in the card reader.

There are three program stops.

1. If a card being read in has a non-Hollerith character, the program will stop with 120_8 in the SR. The card which has just been read should be corrected, it and all unread cards replace in the card reader, and the "START" button pushed.
2. If compiling halts because of an error, a stop will occur with 5 in the address of the SR. Press "START" to load a self-loading card.
3. If there has been a machine error, there will be a stop with 1747_8 in the SR. Do not proceed until the trouble has been determined.

Note: The RREC 2 card brings into memory a code which transfers the FORTRAN output from tape 2 to tape 9. If the local FORTRAN has such a routine, do not use RREC 2.

Note: As soon as RREC 2 has been read, a new tape 3 with new BCD input may be hung.

PROBLEM ~SAMPLE~ DATE PAGE OF PROGRAMMER

FORTRAN STATEMENT

C ← For comment	Statement number (1–5)	Continuation (6)	FORTRAN STATEMENT (7–72)	Identification (73–80)
			EXAMPLES ØF ALGAE I DEFINITIØNS:	
C	E_{32}		CØMMENTS˚MAY˚APPEAR˚ANYWHERE˚WITHIN˚AN˚E	REMARKS
			Z = X**2 + 3.2 *Y + SQRTF(VM)	MAY˚BE
			T(I) = U(NPØS) **2 + V(NMIN) + W (N3) − SQRTF (Z **3) / 4.0* N1	PUNCHED
		x	+ N2	HERE –
	E_{16}		PAUSE	PRINTED
C			BE˚SURE˚TØ˚IDENTIFY˚EACH˚CØMMENT˚BY˚A˚C˚IN˚	ØFF-LINE
C			CØLUMN˚1.	BY˚ALGAE
	C_3		(X ** 2 −3.0*Y(I) + 7.29) P*	
	C_5		(Y−2) =	
	C_1		DIVIDE˚CHECK˚ØN	
	C_2		SENSE˚LIGHT˚3˚ØFF	
	C_4		S˚SW˚4˚ØN	
	C_{153}		$C_1 + (C_3 C_4^* + C_5) C_6^*$	
	I_4		JT = 3(15)4	
	M_2		M3 = MAX (−3) MIN	
	S_4		$C_1 (E_3 + E_4 C_2 + I_4 (M_2 (E_6 + I_3 E_4)) + T_3 + C_1 (S_5, S_6) + J_3 (E_4 + I_2($	
			$C_1 C_6 E_7 + S_3))) + E_{16}$	
	T_{71}		$K_2 (E_1 + (E_2, E_3) C_4) + C_1 G_2$	
			(The above statements are not intended to form a consistent set.)	

3.4 Sample Problems

On the following pages will be found two sample problems coded in Algae. The first problem is correctly coded. Most questions arising about the use of forms, control cards, etc., may be answered by careful perusal of this listing.

In the second code, "Laplace", on page 25 ff., intentional errors have been made so that the reader may become familiar with the diagnostic procedures used in Algae I. For a more complete discussion of diagnostics, see section 4. of this writeup.

```
ALG  C     ALGAE DEMONSTRATION CODE
     C     ARITHMETIC-GEOMETRIC MEAN COMPUTER
           DIMENSION     XP(25), YP(25), X(25), Y(25), FXY(25, 25)
           EQUIVALENCE  (X, XP), (Y, YP)
      1    FORMAT     (2I3)
      2    FORMAT     (35 H1          ALGAE DEMONSTRATION CODE     /     45 H
          A           ARITHMETIC-GEOMETRIC MEAN COMPUTER     /     8 H0      I
          B=    I3,    4 H, J=    I3)
      3    FORMAT     (36 H0     X(I)          Y(J)        F(I,J)      )
      4    FORMAT     (3 E 13.4)
      5    FORMAT     (36 H0     F(I,J) FROM PREVIOUS ITERATION     /     (8
          AE 13.4))
      6    FORMAT     (35 H0    SUMX          SUMY           AGM     /  3E13.4)
     CE1   COMPUTE 2 PI.  SET UP DXP, DYP.                                      E001.
           CAPI = ICAP                                                          E001.01
           CAPJ = JCAP                                                          E001.02
           TWOPI = 2.0 * 3.14159                                                E001.03
           DX = TWOPI / CAPI                                                    E001.04
           DY = TWOPI / CAPJ                                                    E001.05
      E23  T = I-1                                                              E023.01
     C     COMPUTE THE VALUES OF X PRIME.                                       E023.
           XP(I) = T * DX                                                       E023.02
      E24  T = J-1                                                              E024.01
     C     COMPUTE THE VALUES OF Y PRIME.                                       E024.
           YP(J) = T * DY                                                       E024.02
      E3   AM = AM1                                                             E003.01
           GM = GM1                                                             E003.02
      E4   X(I) = SINF (XP(I))                                                  E004.01
     C     COMPUTATION OF X.                                                    E004.
     CE5   FIRST COMPUTATION OF Y.                                              E005.
           Y(J) = COSF (EXPF (YP(J)))                                           E005.01
     CE6   SECOND COMPUTATION OF Y.                                             E006.
           Y(J) = COSF (EXPF (-YP(J)))                                          E006.01
      E7   AM = X(I)                                                            E007.01
     C     PREPARE AGM COMPUTATION.                                             E007.
           GM = Y(J)                                                            E007.02
     CE8   AGM COMPUTATION.                                                     E008.
     C     ARITHMETIC MEAN.                                                     E008.
           AM1 = (AM + GM) / 2.0                                                E008.01
     C     GEOMETRIC MEAN.                                                      E008.
           GM1 = SQRTF (AM * GM)                                                E008.02
      E2   AGM = AM1                                                            E002.01
      E9   FXY(I,J) = AGM                                                       E009.01
      E10  FXY(I,J) = 0.0                                                       E010.01
      E11  FXY(I,J) = -1.0                                                      E011.01
      E12  FXY(I,J) = 1.0                                                       E012.01
     CE13  INITIALIZE SUMS.                                                     E013.
           SUMX = 0.                                                            E013.01
           SUMY = 0.                                                            E013.02
      E14  SUMX = SUMX + ABSF (XP(I))                                           E014.01
      E22  PRINT 6, SUMX, SUMY, AGM                                             E022.01
      E15  SUMY = SUMY + ABSF (YP(J))                                           E015.01
     CE16  SET TO COMPUTE AGM OF SUMS.                                          E016.
           AM = SUMX                                                            E016.01
           GM = SUMY                                                            E016.02
           SENSE LIGHT 1                                                        E016.03
      E18  READ 1, ICAP, JCAP                                                   E018.01
           PRINT 2, ICAP, JCAP                                                  E018.02
      E19  PRINT 4, X(I), Y(J), FXY(I,J)                                        E019.01
      E20  PRINT 3                                                              E020.01
      E21  PRINT 5, ((FXY(I,J), I = 1, ICAP), J = 1, JCAP)                      E021.01
      E25  SENSE LIGHT 2                                                        E025.01
     CE26  IGNORE TEST.                                                         E026.
```

```
N1    I = ICAP (-1) 1
M2    J = 1 (1) JCAP
C1    (X(I)) =
C2    (Y(J)) =
C3    (X(I)) N
C4    (Y(J)) N*
C5    SWITCH 6 OFF
C7    ACCUM OVERFLOW ON
C151  (C1 C4 C2* + C2 C3* C1*)
C8    S LITE 1 OFF
C9    (ABSF (AM1 - GM1) / GM1 - .0001) N
C10   SENSE LT 2 ON
T1    (E3, E8) C9 + E2
T2    E1 + N1 E23 + M2 E24
S1    C3* C1* C4 C2* (E7 + T1 + C7 (E10, E9), C151 (E12, C3 C4*
      (E11, E10)))
S2    T2 + N1 E4 + C8 (M2 E5, M2 E6) + C5* E20
      +N1 (M2 (S1 + C5* E19)) + E21
S3    C7 E26 + H1 + E18 + H2 + S2 + C10* (S4, G1)
S4    E25 + T2 + E13 + N1 E14 + M2 E15 + E16 + T1 + E22 + G2
```

ALGAE OBJECT CODE FROM FORTRAN TAPE 2

```
C       ALGAE DEMONSTRATION CODE
C       ARITHMETIC-GEOMETRIC MEAN COMPUTER
        DIMENSION      XP(25), YP(25), X(25), Y(25), FXY(25, 25)
        EQUIVALENCE  (X, XP), (Y, YP)
 1      FORMAT       (2I3)
 2      FORMAT       (35 H1          ALGAE DEMONSTRATION CODE       /      45 H
       A            ARITHMETIC-GEOMETRIC MEAN COMPUTER      /      8 HO       I
       B=    I3,    4 H, J=     I3)
 3      FORMAT       (36 HO      X(I)          Y(J)        F(I,J)       )
 4      FORMAT       (3 E 13.4)
 5      FORMAT       (36 HO      F(I,J) FROM PREVIOUS ITERATION      /       (8
       AE 13.4))
 6      FORMAT       (35 HO     SUMX           SUMY           AGM      /  3E13.4)
        IF     ACCUMULATOR OVERFLOW                              0144, 0145 C007
 0144 CONTINUE
C       IGNORE TEST.                                                        E026.
 0145 CONTINUE
 0112 CONTINUE
        READ 1, ICAP, JCAP                                                  E018.01
        PRINT 2, ICAP, JCAP                                                 E018.02
 0113 CONTINUE
        LL002=0002                                                          T002
        GO  TO  0111
 0132 CONTINUE
        DO0133       III    =      1,  ICAP,      1                        N001
            I=      1+  ICAP-III                                           N001
        X(I) = SINF (XP(I))                                                 E004.01
C       COMPUTATION OF X.                                                   E004.
 0133 CONTINUE
        IF     (SENSE LIGHT 1)                                   0135, 0134 C008
 0134 CONTINUE
        DO0129            J=      1,  JCAP,      1                         M002
C       FIRST COMPUTATION OF Y.                                             E005.
        Y(J) = COSF (EXPF (YP(J)))                                          E005.01
 0129 CONTINUE
        GO  TO  0136
 0135 CONTINUE
        DO0130            J=      1,  JCAP,      1                         M002
C       SECOND COMPUTATION OF Y.                                            E006.
        Y(J) = COSF (EXPF (-YP(J)))                                         E006.01
 0130 CONTINUE
 0136 CONTINUE
        IF     (SENSE SWITCH 6)                                  0137, 0138 C005
 0137 CONTINUE
        PRINT 3                                                             E020.01
 0138 CONTINUE
        DO0139       III    =      1,  ICAP,      1                        N001
            I=      1+  ICAP-III                                           N001
        DO0131            J=      1,  JCAP,      1                         M002
        IF(X(I))                                           0125, 0102, 0102 C003
 0102 IF(X(I))                                             0101, 0125, 0101 C001
 0101 IF(Y(J))                                             0125, 0100, 0100 C004
 0100 IF(Y(J))                                             0124, 0125, 0124 C002
 0124 CONTINUE
        AM = X(I)                                                           E007.01
C       PREPARE AGM COMPUTATION.                                            E007.
        GM = Y(J)                                                           E007.02
        LL001=0002                                                          T001
        GO  TO  0110
 0117 CONTINUE
        IF     ACCUMULATOR OVERFLOW                              0118, 0119 C007
 0118 CONTINUE
```

```
      FXY(I,J) = 0.0                                              E010.01
      GO  TO  0120
 0119 CONTINUE
      FXY(I,J) = AGM                                              E009.01
 0120 CONTINUE
      GO  TO  0126
 0125 CONTINUE
      IF(X(I))                                  0105, 0107, 0105 C001
 0107 IF(Y(J))                                  0105, 0106, 0106 C004
 0106 IF(Y(J))                                  0121, 0105, 0121 C002
 0105 IF(Y(J))                                  0122, 0104, 0122 C002
 0104 IF(X(I))                                  0122, 0103, 0103 C003
 0103 IF(X(I))                                  0121, 0122, 0121 C001
 0121 CONTINUE
      FXY(I,J) = 1.0                                              E012.01
      GO  TO  0123
 0122 CONTINUE
      IF(X(I))                                  0108, 0115, 0115 C003
 0108 IF(Y(J))                                  0114, 0115, 0115 C004
 0114 CONTINUE
      FXY(I,J) = -1.0                                             E011.01
      GO  TO  0116
 0115 CONTINUE
      FXY(I,J) = 0.0                                              E010.01
 0116 CONTINUE
 0123 CONTINUE
 0126 CONTINUE
      IF    (SENSE SWITCH 6)                          0127, 0128 C005
 0127 CONTINUE
      PRINT 4, X(I), Y(J), FXY(I,J)                               E019.01
 0128 CONTINUE
 0131 CONTINUE
 0139 CONTINUE
      PRINT 5, ((FXY(I,J), I = 1, ICAP), J = 1, JCAP)             E021.01
      IF    (SENSE LIGHT 2)                           0147, 0146 C010
 0146 CONTINUE
      SENSE LIGHT 2                                               E025.01
      LL002=0001                                                  T002
      GO  TO  0111
 0140 CONTINUE
C     INITIALIZE SUMS.                                            E013.
      SUMX = 0.                                                   E013.01
      SUMY = 0.                                                   E013.02
      DO0141       III   =     1,  ICAP,      1                  N001
          I=     1+  ICAP-III                                    N001
      SUMX = SUMX + ABSF (XP(I))                                  E014.01
 0141 CONTINUE
      DO0142             J=     1,  JCAP,      1                 M002
      SUMY = SUMY + ABSF (YP(J))                                  E015.01
 0142 CONTINUE
C     SET TO COMPUTE AGM OF SUMS.                                 E016.
      AM = SUMX                                                   E016.01
      GM = SUMY                                                   E016.02
      SENSE LIGHT 1                                               E016.03
      LL001=0001                                                  T001
      GO  TO  0110
 0143 CONTINUE
      PRINT 6, SUMX, SUMY, AGM                                    E022.01
      GO  TO  0113
 0147 CONTINUE
      GO  TO  0112
 0111 CONTINUE                                                   T002
C     COMPUTE 2 PI.  SET UP DXP, DYP.                             E001.
```

```
      CAPI = ICAP                                                     E001.01
      CAPJ = JCAP                                                     E001.02
      TWOPI = 2.0 * 3.14159                                           E001.03
      DX = TWOPI / CAPI                                               E001.04
      DY = TWOPI / CAPJ                                               E001.05
      DO0149      III    =      1,  ICAP,      1                     N001
          I=      1+  ICAP-III                                       N001
      T = I-1                                                         E023.01
C     COMPUTE THE VALUES OF X PRIME.                                  E023.
      XP(I) = T * DX                                                  E023.02
 0149 CONTINUE
      DO0150            J=      1,  JCAP,      1                     M002
      T = J-1                                                         E024.01
C     COMPUTE THE VALUES OF Y PRIME.                                  E024.
      YP(J) = T * DY                                                  E024.02
 0150 CONTINUE
      GO TO(0140 ,0132), LL002
 0110 CONTINUE                                                       T001
      GO  TO  0151
 0152 CONTINUE
      AM = AM1                                                        E003.01
      GM = GM1                                                        E003.02
 0151 CONTINUE
C     AGM COMPUTATION.                                                E008.
C     ARITHMETIC MEAN.                                                E008.
      AM1 = (AM + GM) / 2.0                                           E008.01
C     GEOMETRIC MEAN.                                                 E008.
      GM1 = SQRTF (AM * GM)                                           E008.02
      IF(ABSF (AM1 - GM1) / GM1 - .0001)          0153, 0152, 0152 C009
 0153 CONTINUE
      AGM = AM1                                                       E002.01
      GO TO(0143 ,0117), LL001
```

STORAGE FOR VARIABLES APPEARING IN DIMENSION OR EQUIVALENCE SENTENCES

	DEC	OCT		DEC	OCT		DEC	OCT		DEC	OCT		DEC	OCT
FXY	32717	77715	XP	32767	77777	X	32767	77777	YP	32742	77746	Y	32742	77746

STORAGE FOR VARIABLES WHICH DO NOT APPEAR IN DIMENSION OR EQUIVALENCE SENTENCES

	DEC	OCT		DEC	OCT		DEC	OCT		DEC	OCT		DEC	OCT
I	32092	76534	III	32091	76533	ICAP	32090	76532	GM	32089	76531	GM1	32088	76530
DY	32087	76527	DX	32086	76526	CAPJ	32085	76525	CAPI	32084	76524	AM	32083	76523
AM1	32082	76522	AGM	32081	76521	JCAP	32080	76520	J	32079	76517	LL001	32078	76516
LL002	32077	76515	SUMX	32076	76514	SUMY	32075	76513	T	32074	76512	TWOPI	32073	76511

EXTERNAL FORMULA NUMBERS WITH CORRESPONDING INTERNAL FORMULA NUMBERS AND OCTAL LOCATIONS

EFN	IFN	LOC	EFN	IFN	LOC	EFN	IFN	LOC	EFN	IFN	LOC	EFN	IFN	LOC
1	3	00000	2	4	00000	3	5	00000	4	6	00000	5	7	00000
6	8	00000	144	10	00002	145	11	00002	112	12	00003	113	19	00037
132	22	00044	133	26	00064	134	28	00072	129	31	00103	135	33	00106
130	36	00117	136	37	00121	137	39	00123	138	42	00134	102	47	00164
101	48	00166	100	49	00172	124	50	00174	117	55	00207	118	57	00211
119	60	00216	120	62	00220	125	64	00223	107	66	00227	106	67	00235
105	68	00241	104	69	00247	103	70	00253	121	71	00255	122	74	00260
108	76	00265	114	77	00270	115	80	00273	116	82	00275	123	83	00275
126	84	00277	127	86	00301	128	90	00317	131	91	00317	139	92	00322
146	102	00374	140	106	00402	141	112	00425	142	115	00437	143	121	00455
147	126	00474	111	128	00476	149	138	00545	150	142	00566	110	144	00576
152	146	00577	151	149	00603	153	153	00626						

SUBROUTINES OBTAINED FROM LIBRARY

	DEC	OCT		DEC	OCT		DEC	OCT		DEC	OCT		DEC	OCT
(DBC)	491	00753	(CSH)	1008	01760	(BDC)	1089	02101	(FIL)	1119	02137	(LEV)	1132	02154
(RTN)	1159	02207	(SPH)	1639	03147	SQRT	1731	03303	COS	1753	03331	SIN	1756	03334
EXP	1810	03422												

```
9A      TOV  10A
        TRA  11A
10A     BSS
11A     BSS
D)403   LXD  C)G0,4
12A     BSS
13A     CAL  *
        XIT  (LEV)
        ETM
        CAL  (DBC)
        SLW  1
        CAL  (CSH)
13D1    NTR  8)1,0,81
14A     ETM
        NTR  ICAP
        NTR  JCAP
        LTM
15A     CAL  *
        XIT  (RTN)
16A     CAL  *
        XIT  (LEV)
        ETM
        CAL  (BDC)
        SLW  1
        CAL  (SPH)
16D1    NTR  8)2
17A     ETM
        NTR  ICAP
        NTR  JCAP
        LTM
18A     CAL  *
        XIT  (FIL)
D)504   LXD  C)100,1
D)404   LXD  C)G3,4
19A     BSS
20A     CLA  2)
        STO  LL002
        LXD  LL002,2
        SXD  C)G2,2
21A     TRA  128A
22A     BSS
23A     LXD  2)+1,4
        CLA  ICAP
        STD  26A2
        PXD  0,4
        STO  III
24A     CLA  2)+1
        ADD  ICAP
        SUB  III
        STO  I
        LXD  I,2
        SXD  C)G0,2
25A     CLA  XP+1,2
        SXD  6)+4,4
        TSX  SIN,4
        LXD  6)+4,4
        STO  X+1,2
26A     BSS
26A1    TXI  *+1,4,1
        SXD  III,4
26A2    TXL  24A,4
27A     MSE  97
        TRA  28A
        TRA  33A
28A     BSS
29A     LXD  2)+1,4
        CLA  JCAP
        STD  31A2
30A     CLA  YP+1,4
        SXD  6)+4,4
        TSX  EXP,4
        TSX  COS,4
        LXD  6)+4,4
        STO  Y+1,4
31A     BSS
31A1    TXI  *+1,4,1
31A2    TXL  30A,4
32A     TRA  37A
33A     BSS
34A     LXD  2)+1,4
        CLA  JCAP
        STD  36A2
35A     CLS  YP+1,4
        SXD  6)+4,4
        TSX  EXP,4
        TSX  COS,4
        LXD  6)+4,4
        STO  Y+1,4
36A     BSS
36A1    TXI  *+1,4,1
36A2    TXL  35A,4
37A     BSS
38A     PSE  118
        TRA  42A
39A     BSS
40A     CAL  *
        XIT  (LEV)
        ETM
        CAL  (BDC)
        SLW  1
        CAL  (SPH)
40D1    NTR  8)3
41A     CAL  *
        XIT  (FIL)
42A     BSS
43A     LXD  2)+1,4
        SXD  C)201,4
        CLA  ICAP
        STD  92A2
        PXD  0,4
        STO  III
44A     CLA  2)+1
        ADD  ICAP
        SUB  III
        STO  I
        LXD  I,4
        SXD  C)G0,4
45A     LXD  2)+1,1
        SXD  C)G1,1
        CLA  JCAP
        STD  91A2
        LXD  I,2
        SXD  C)100,2
D)30I   LXD  C)100,1
D)20I   LXD  C)G1,2
46A     CLA  X+1,4
46A1    TZE  47A
        TPL  47A
        TRA  64A
47A     CLA  X+1,4
47A1    TZE  E)20R
48A     CLA  Y+1,2
48A1    TZE  49A
        TPL  49A
        TRA  E)10R
49A     CLA  Y+1,2
49A1    TZE  64A
50A     BSS
51A     CLA  X+1,4
        STO  AM
52A     CLA  Y+1,2
        STO  GM
53A     CLA  2)
        STO  LL001
        LXD  LL001,4
        SXD  C)G3,4
54A     PXD  0,2
        STO  J
        TRA  144A
55A     BSS
56A     TOV  57A
        TRA  E)P
57A     BSS
58A     CLA  3)
        STO  FXY+1,1
59A     TRA  62A
E)P     SXD  C)G1,2
        SXD  C)100,1
60A     BSS
61A     CLA  AGM
        STO  FXY+1,1
62A     BSS
63A     TRA  D)418
E)10R   SXD  C)G1,2
        SXD  C)100,1
E)20R   SYN  E)10R
64A     BSS
65A     CLA  X+1,4
65A1    TZE  66A
        TPL  68A
        TRA  68A
66A     CLA  Y+1,2
66A1    TZE  E)T
        TPL  E)T
        TRA  68A
E)T     SXD  C)G1,2
        SXD  C)100,1
67A     CLA  Y+1,2
67A1    TZE  68A
        TPL  71A
```

```
        TRA 71A
68A     CLA Y+1,2
68A1    TZE E)V
        TPL 74A
        TRA 74A
E)V     SXD C)G1,2
        SXD C)100,1
69A     CLA X+1,4
69A1    TZE 70A
        TPL 70A
        TRA 74A
70A     CLA X+1,4
70A1    TZE 74A
71A     BSS
72A     CLA 3)+1
        STO FXY+1,1
73A     TRA 83A
74A     BSS
75A     CLA X+1,4
75A1    TZE 80A
        TPL 80A
E)13    SXD C)G1,2
        SXD C)100,1
76A     CLA Y+1,2
76A1    TZE 80A
        TPL 80A
77A     BSS
78A     CLS 3)+1
        STO FXY+1,1
79A     TRA 82A
80A     BSS
81A     CLA 3)
        STO FXY+1,1
82A     BSS
83A     BSS
        TRA 84A
D)418   LXD C)G0,4
84A     BSS
85A     PSE 118
        TRA 90A
86A     BSS
87A     CAL *
        XIT (LEV)
        ETM
        CAL (BDC)
        SLW 1
        CAL (SPH)
87D1    NTR 8)4
88A     ETM
        NTR X+1,4
        NTR Y+1,2
        NTR FXY+1,1
        LTM
89A     CAL *
        XIT (FIL)
90A     BSS
91A     BSS
91A1    TXI *+1,2,1
        TXI *+1,1,25
91A2    TXL 46A,2
```

```
92A     BSS
        LXD C)201,2
92A1    TXI *+1,2,1
        SXD C)201,2
        SXD III,2
92A2    TXL 44A,2
93A     CAL *
        XIT (LEV)
        ETM
        CAL (BDC)
        SLW 1
        CAL (SPH)
93D1    NTR 8)5
94A     LXD 2)+1,2
        CLA ICAP
        STD 97A2
        LDQ JCAP
        MPY 2)+2
        ALS 17
        STD 98A4
        CLA ICAP
        SUB 6)+3
        STD 98A3
        ADD 2)+2
        STD 98A1
        CLA ICAP
        STD 97A3
95A     BSS
96A     BSS
97A     ETM
        NTR FXY+1,2
        LTM
97A1    TXI *+1,2,1
97A2    TXL 97A,2
97A3    TIX 97A3+1,2
98A1    TXI 98A1+1,2
        SXD 97A2,2
98A3    TIX 98A3+1,2
98A4    TXL 95A,2
99A     LTM
100A    CAL *
        XIT (FIL)
101A    MSE 98
        TRA 102A
        TRA 126A
102A    BSS
103A    PSE 98
104A    CLA 2)+1
        STO LL002
        LXD LL002,2
        SXD C)G2,2
105A    TRA D)11P
106A    BSS
107A    CLA 3)
        STO SUMX
108A    CLA 3)
        STO SUMY
109A    LXD 2)+1,4
        CLA ICAP
        STD 112A2
```

```
        PXD 0,4
        STO III
110A    CLA 2)+1
        ADD ICAP
        SUB III
        STO I
        LXD I,2
        SXD C)G0,2
111A    CLA XP+1,2
        SSP
        FAD SUMX
        STO SUMX
112A    BSS
112A1   TXI *+1,4,1
        SXD III,4
112A2   TXL 110A,4
113A    LXD 2)+1,4
        CLA JCAP
        STD 115A2
114A    CLA YP+1,4
        SSP
        FAD SUMY
        STO SUMY
115A    BSS
115A1   TXI *+1,4,1
115A2   TXL 114A,4
116A    CLA SUMX
        STO AM
117A    CLA SUMY
        STO GM
118A    PSE 97
119A    CLA 2)+1
        STO LL001
        LXD LL001,4
        SXD C)G3,4
120A    TRA D)31U
E)1N    SXD C)G1,2
        SXD C)100,1
121A    BSS
122A    CAL *
        XIT (LEV)
        ETM
        CAL (BDC)
        SLW 1
        CAL (SPH)
122D1   NTR 8)6
123A    ETM
        NTR SUMX
        NTR SUMY
        NTR AGM
        LTM
124A    CAL *
        XIT (FIL)
125A    TRA 19A
126A    BSS
127A    TRA 12A
D)11P   LXD C)100,1
128A    BSS
129A    CLA ICAP
        LRS 18
```

```
        ORA 6)
        FAD 6)
        STO CAPI
130A    CLA JCAP
        LRS 18
        ORA 6)
        FAD 6)
        STO CAPJ
131A    LDQ 3)+2
        FMP 3)+3
        STO TWOPI
132A    CLA TWOPI
        FDP CAPI
        STQ DX
133A    CLA TWOPI
        FDP CAPJ
        STQ DY
134A    LXD 2)+1,4
        CLA ICAP
        STD 138A2
        PXD 0,4
        STO III
135A    CLA 2)+1
        ADD ICAP
        SUB III
        STO I
        LXD I,1
        SXD C)G0,1
136A    CLA I
        SUB 2)+1
        LRS 18
        ORA 6)
        FAD 6)
        STO T
137A    LDQ T
        FMP DX
        STO XP+1,1
138A    BSS
138A1   TXI *+1,4,1
        SXD III,4
138A2   TXL 135A,4
139A    LXD 2)+1,4
        CLA JCAP
        STD 142A2
        PXD 0,4
        STO J
140A    CLA J
        SUB 2)+1
        LRS 18
        ORA 6)
        FAD 6)
        STO T
141A    LDQ T
        FMP DY
        STO YP+1,4
142A    BSS
142A1   TXI *+1,4,1
        SXD J,4
142A2   TXL 140A,4
143A    TRA 143A+3,2
        TRA 22A
        TRA 106A
D)31U   LXD C)100,1
D)21U   LXD C)G1,2
144A    BSS
145A    TRA 149A
146A    BSS
147A    CLA AM1
        STO AM
148A    CLA GM1
        STO GM
149A    BSS
150A    CLA AM
        FAD GM
        FDP 3)+2
        STQ AM1
151A    LDQ AM
        FMP GM
        SXD 6)+4,4
        TSX SQRT,4
        LXD 6)+4,4
        STO GM1
152A    CLA AM1
        FSB GM1
        SSP
        FDP GM1
        STQ 1)+1
        CLA 1)+1
        FSB 3)+4
152A1   TZE 146A
        TPL 146A
153A    BSS
154A    CLA AM1
        STO AGM
155A    TRA 155A+3,4
        TRA 55A
        TRA E)1N
2)      OCT +000002000000
        OCT +000001000000
        OCT +000031000000
3)      OCT +000000000000
        OCT +201400000000
        OCT +202400000000
        OCT +202622077174
        OCT +163643334272
6)      OCT +233000000000
        OCT +000000077777
        OCT +000000000000
        OCT +000001000000
        OCT +000000000000
8)1     BCD 1     (
        BCD 12I3)
        OCT -377777777777
8)2     BCD 1     (
        BCD 135 H1
        BCD 1
        BCD 1   ALG
        BCD 1AE DEM
        BCD 1ONSTRA
        BCD 1TION C
        BCD 1ODE
        BCD 1  /
        BCD 1  45 H
        BCD 1
        BCD 1     A
        BCD 1RITHME
        BCD 1TIC-GE
        BCD 1OMETRI
        BCD 1C MEAN
        BCD 1 COMPU
        BCD 1TER
        BCD 1  /
        BCD 1  8 HO
        BCD 1     I
        BCD 1=   I3
        BCD 1,   4
        BCD 1H, J=
        BCD 1   I3)
        OCT -377777777777
8)3     BCD 1     (
        BCD 136 HO
        BCD 1    X(
        BCD 1I)
        BCD 1     Y
        BCD 1(J)
        BCD 1    F(
        BCD 1I,J)
        BCD 1   )
        OCT -377777777777
8)4     BCD 1     (
        BCD 13 E 13
        BCD 1,4)
        OCT -377777777777
8)5     BCD 1     (
        BCD 136 HO
        BCD 1    F(
        BCD 1I,J) F
        BCD 1ROM PR
        BCD 1EVIOUS
        BCD 1 ITERA
        BCD 1TION
        BCD 1   /
        BCD 1   (8
        BCD 1E 13,4
        BCD 1))
        OCT -377777777777
8)6     BCD 1     (
        BCD 135 HO
        BCD 1   SUM
        BCD 1X
        BCD 1    SU
        BCD 1MY
        BCD 1
        BCD 1AGM
        BCD 1  /  3
        BCD 1E13,4)
        OCT -377777777777
```

```
ALG  C      ALGAE LAPLACE EQUATION ROUTINE -- TYPE 1 MESH
     C           THIS CODE SOLVES THE LAPLACE EQUATION ON A 10 X 10 MESH
     C           WITH BOUNDARIES SET TO 1.
            DIMENSION      X(10, 10)
       1    FORMAT  (30H1ALGAE LAPLACE EQUATION SOLVER/  43H      10 X 10 MESH,
           A WITH BOUNDARIES SET TO 1)
       2    FORMAT  (10 E 12.4)
       3    FORMAT  (36H0ITERATION ON LAPLACE EQUATION MESH  / (10E11.3)
     C
     C      THIS COMPILING WILL DEMONSTRATE ERROR FORMATS.
     C
      S2    E3 + I2 (J2 E)
      S1    I (J1 ((C1 + C2 + (C3 C4)*) (E1, E2))
      S3    S1 + (S2 C6 E7) C5 + E5 + E8
      I1    I = 1 (1)
      I2    I = 2 (1) 9
      J1    J = 1 (1) 10, AND SO ON
      J2    J = 2 (1) 9
      C1         (I-1)
      C2         (I-10) = EQUAL
      C3         (J-2) N*
      C4         (J-9  P*
      C5         C - 0.001) N
      C6         (SW 6 ON)
     CE1    SET UP BOUNDARY VALUES OF MESH                                     E001.
            X(I,J) = 1.0                                                       E001.01
     CE2    SET INTERIOR TO ZERO                                               E002.
            X(I,J) = 0.0                                                       E002.01
      E3    C = 0.0                                                            E003.01
     CE4    CALCULATE DIFFERENCES,                                             E004.
            E = (X(I-1,J) + X(I+1,J) + X (I,J-1) + (XI,J+1)) / 4.0             E004.01
     C      SUM DIFFERENCES,                                                   E004.
            C = C + ABSF (E - X(I,J))                                          E004.02
     C      AND SET NEW VALUE OF X(I,J).                                       E004.
            X(I,J) = E                                                         E004.03
      E5    PRINT 1                                                            E005.01
            PRINT 2,  (X)                                                      E005.02
      E6    PAUSE                                                              E006.01
      E7    PRINT 3,  (X)                                                      E007.01
                                                              S1      IMPROPER       I0         CODE  07241
                                                              S1   HAS TOO MANY (,S             CODE  Ø7042
                                                              C1   IMPROPER STATEMENT           CODE  10222
                                                              C2   IMPROPER STATEMENT E         CODE  10254
                                                              C4   IMPROPER STATEMENT           CODE  10535
                                                              C5   IMPROPER STATEMENT C         CODE  10346
                                                              C6   IMPROPER STATEMENT           CODE  10222
                                                              S1      IMPROPER       WO         CODE  04715
                                                              S2          UNDEFINED  E0         CODE  04624
                                                              S3   HAS IMPROPER TERM  E7        CODE  04451
                                                              S3          UNDEFINED  E8         CODE  04624
```

```
C     ALGAE LAPLACE EQUATION ROUTINE -- TYPE 1 MESH
C          THIS CODE SOLVES THE LAPLACE EQUATION ON A 10 X 10 MESH
C          WITH BOUNDARIES SET TO 1.
      DIMENSION     X(10, 10)
 1    FORMAT  (30H1ALGAE LAPLACE EQUATION SOLVER/  43H      10 X 10 MESH,
     A WITH BOUNDARIES SET TO 1)
 2    FORMAT  (10 E 12.4)
 3    FORMAT  (36H0ITERATION ON LAPLACE EQUATION MESH  / (10E11.3)
C
C     THIS COMPILING WILL DEMONSTRATE ERROR FORMATS.
C
 0107 CONTINUE
 0104 CONTINUE
 0102 C = 0.0                                                          E003.01
      DO0103           I=      2,      9,      1                      I002
      DO0101           J=      2,      9,      1                      J002
 0101 CONTINUE
 0103 CONTINUE
      IF(SW 6 ON)                                    0105, 0105, 0105 C006
 0105 CONTINUE
 0106 PRINT 3,   (X)                                                   E007.01
      CODING ERROR                                         0107, 0107 C005
 0108 CONTINUE
 0109 PRINT 1                                                          E005.01
 0110 PRINT 2,   (X)                                                   E005.02
                                              HALT                    FIX AND RE-TRY          CODE  02563
                                                             0011     DIAGNOSTICS
```

4. Error Procedure and Description of Diagnostics

Algae I is designed to detect errors in source programs presented to it, and to give as part of its output as much information as is possible concerning those errors.

The compiling process may be divided into two phases. In Phase 1, the source program is read into the 704 DPM, statements are checked for duplication and improper subscripts, and diverse tables needed in Phase 2 are set up. In Phase 2 the FORTRAN source program is compiled. Detection of an error in Phase 1 prevents the execution of Phase 2. Detection of an error in Phase 2 prevents the execution of FORTRAN.

Errors may be divided into four major classes. These are:

- I. Machine errors.
- II. Phase 1 source program errors.
- III. Phase 2 source program errors which are relatively limited in their effect upon the code.
- IV. Phase 2 source program errors whose nature makes further compiling impossible.

Class I errors may cause a program stop at 1747_8. An identification number, ALPHA, is printed on the line printer together with a brief description of the failure. Class II, III, and IV errors cause a statement to be printed on the off-line listing, containing an identification number, ALPHA, and a brief description of the source program error. Class II errors will cause the next problem to be loaded at the end of Phase I. Class III errors will cause the next problem to be loaded at the end of Phase 2. Class IV errors will cause the next problem to be loaded whenever they are detected. Before leaving a problem the total number of source program errors which have been detected is printed both on the line printer and off-line listing.

Information printed about errors is listed in the following format: H ID COMMENT CHAR TYPE ALPHA

The H position contains the word HALT if processing of the problem is interrupted for any reason. ID is the identification of the objectionable statement when available. COMMENT is a brief description of the error. CHAR is the term of the statement, if any, to which exception is taken. TYPE indicates whether the error is a machine failure or source program mistake. ALPHA is the octal location at which the error was detected. By looking up this number in the following ALPHA table, a slightly expanded description may be obtained.

There are two symbols which may occur in the CHAR column on the listing which are the result of coding errors, but which do not appear in the source program at any point.

The first of these is W_0. This character is inserted at one stage in the code when a character has been determined to be improper in some respect. The insertion of W_0 permits Algae I to continue compiling, and also permits the programmer to trace the effect of his errors on the resulting code.

The second is an E with subscript ≥ 1000. This symbol is the identification of the entire coded contents of a parenthetical expression. It sometimes happens that a program will give rise to a vacuous parenthesis level, or other similar error. Then, when it is desired to incorporate this grouping into the rest of the code, the E_{1000} symbol is printed.

The format of the ALPHA table is as follows:

ALPHA	COMMENT	CLASS of error
	Nature of error	What to do about it

ALL ALPHA's are in octal.

5.0 Summary on Referencing

The following table gives, in condensed form, most of the restrictions on the Algae I source program.

Statement Type	Subscript Range	max. no. defined	no. of times each may be used	no. of cards / no. of char.	Remarks
S_i	$1 \leq i \leq 999$	500	1	70 / 1000	Control statement
T_i	$1 \leq i \leq 100$	100	100	70 / 1000	Subroutine
C_i	$1 \leq i \leq 149$	149	Indefinite	1 / 48 including blanks & ( )	Simple Condition
C_i	$150 \leq i \leq 200$	51	Indefinite	1 / 66	C-statement. May not contain another C-statement.
$I_i - N_i$	$1 \leq i \leq 999$	150	Unlimited	1 / See p. 10	Range statement
G_i	$1 \leq i \leq 200$	200†	Unlimited	0 / 0	Transfer statement
H_i	$1 \leq i \leq 200$	200†	1	0 / 0	Point of transfer entry.

†i may be greater than 200, but in large problems the available formula numbers may be exhausted.