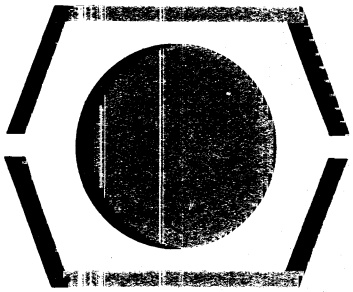


DECUS

PROGRAM LIBRARY

DECUS NO.	8-213
TITLE	4K ALGOL
AUTHOR	University of Grenoble Submitted By: Charles Conley
COMPANY	Digital Equipment Corporation Maynard, Massachusetts
DATE	April 11, 1969
SOURCE LANGUAGE	ALGOL

Although this program has been tested by the contributor, no warranty, express or implied, is made by the contributor, Digital Equipment Computer Users Society or Digital Equipment Corporation as to the accuracy or functioning of the program or related program material, and no responsibility is assumed by these parties in connection therewith.

IDENTIFICATION

Product Code: DEC-08-KAYA-D

Product Name: ALGOL-8

Date Created: March 31,1969

PDP-8 LIBRARY

PREFACE

This manual has been written for ALGOL users with a 4K or larger PDP-8.

The subset of ALGOL compiled by the system is SUBSET ALGOL 60 (IFIP)* with additional restrictions which are described in Appendix B.

The manual is essentially a user's handbook and is not intended as a ALGOL Primer. Chapter 1 is included as a general introduction to ALGOL-8. The reader who wishes to learn the ALGOL language in greater depth should refer to one of the several texts on the subject, such as "A Guide to ALGOL Programming" by Daniel D. McCracken (published by John Wiley and Sons, New York).

This compiler was written by J. C. Boussard, D. Clauzel, and X. Nguyen Dinh of IMAG (Institute of Applied Mathematics, University of Grenoble).

This manual is preliminary and subject to change without notice.

*Communications of the ACM Volume 7, No. 10, October 1964 pp 626-628 "Report on Subset Algol 60 (IFIP).
Communications of the ACM Volume 6, No. 1, January 1963, pp 1-17 "Revised Report on the Algorithmic Language, ALGOL 60."

CONTENTS

CHAPTER 1

INTRODUCTION

1.1	ALGOL-8 Program Form	1-1
1.2	ALGOL-8 Input/Output	1-3
1.3	Sample Programs	1-3
1.3.1	Arithmetic Operators	1-3
1.3.2	Compound Statements	1-7
1.3.3	Blocks	1-10

CHAPTER 2

REPRESENTATION OF ELEMENTS

2.1	Integers	2-1
2.2	Real Numbers	2-1
2.3	Basic Symbols	2-2
2.3.1	Long Basic Symbols	2-2
2.3.2	Short Basic Symbols	2-2
2.4	Identifiers	2-3
2.5	Constants	2-3
2.6	Variables	2-3
2.7	Subscripted Variables and Arrays	2-4
2.8	Statements	2-4
2.9	Labels	2-4

CHAPTER 3

EXPRESSIONS

3.1	Arithmetic Expressions	3-1
3.1.1	Constants	3-1
3.1.2	Variables	3-1
3.1.3	Operators	3-2
3.1.4	Mixed Mode Expressions	3-3
3.1.5	Standard Functions	3-4
3.2	Boolean Expressions	3-4
3.2.1	Forms	3-6
3.2.2	Limitations	3-7

CHAPTER 4

DECLARATIONS

4.1	Variables	4-1
4.2	Arrays	4-1
4.3	Switches	4-2
4.4	Procedures	4-3

CHAPTER 5

STATEMENTS

5.1	Assigned Statements	5-1
5.2	GOTO Statements	5-2
5.3	FOR Statements	5-2
5.4	Conditional Statements	5-4
5.5	Dummy Statements	5-5
5.6	Comments	5-5
5.7	Sample Program	5-7

CHAPTER 6

INPUT/OUTPUT STATEMENTS

6.1	I/O Statement Form	6-1
6.2	Output Format	6-2
6.3	Input Format	6-2
6.4	Text Output	6-2
6.5	New Line	6-3

CHAPTER 7

DIAGNOSTICS

7.1	Compile Time Diagnostics	7-1
7.2	Execution Time Diagnostics	7-6

CHAPTER 8

OPERATING INSTRUCTIONS

8.1	Compiler	8-1
8.1.1	Operation	8-1
8.1.2	Memory Usage	8-4
8.2	Operating System	8-4
8.2.1	Operation - No Functions Required	8-4
8.2.2	Operation - Functions Required	8-5
8.2.3	Memory Usage	8-5
8.3	Paper Tape Input Format	8-6
8.3.1	Compiler Input	8-6
8.3.2	Operating System Input	8-6

APPENDIX A

ALGOL-8 BASIC SYMBOLS

APPENDIX B
RESTRICTIONS

APPENDIX C

TABLE OF ERRORS

APPENDIX D
ALGOL-8 SUMMARY OF COMMANDS

APPENDIX E

NOTES ON ALGOL

Chapter 1

INTRODUCTION

An ALGOL-8 program includes elements (numbers and symbols), expressions (arithmetic and Boolean), statements, and declarations. Precise definitions of these parts of an ALGOL-8 program are contained in later chapters. Those elements which are needed to write simple ALGOL-8 programs are introduced in the following discussion.

1.1 ALGOL-8 Program Form.

ALGOL-8 programs start with a 'BEGIN' statement and conclude with an 'END' statement followed by a $ as shown below.

```
'BEGIN'
        }
        }          Body of ALGOL-8 Program
        }
'END'
$
```

The body of the ALGOL-8 program specifies the action to be performed. For example, the body of the program could contain statements to accept the input of data, perform some calculation, and output the results.

The type of data (integer or real numbers) must be specified before it may be used in an ALGOL-8 program. The declarations 'INTEGER' and 'REAL' are used to specify the type of number which is represented by a symbol, as seen below.

```
'BEGIN'
'REAL' A,B,C;
'INTEGER' I,J,K;
     .
     .
     .
'END'
$
```

The 'INTEGER' and 'REAL' statements, as well as all other ALGOL-8 statements are terminated by a semicolon.

Once the variables are declared they may be used in ALGOL-8 program calculations. For example the following ALGOL-8 program adds three numbers.

```
'BEGIN'
'INTEGER' A,B,C,SUM;
A:=5;
B:=27;
C:=15;
SUM:=A+B+C;
'END'
$
```

Notice in the above program the difference between the variables A, B and C and the variable SUM. A, B and C are integer constants given an explicit value by an assignment statement, such as A:=5;. (The colon must be used with the equal sign in an assignment statement.) SUM is an integer variable whose value is determined by running the program.

The constants of the previous program need not be assigned the identifiers A, B and C. The same result could be achieved through the following program where only SUM is declared as a variable.

```
'BEGIN'
'INTEGER' SUM;
SUM:= 5+27+15;
'END'
$
```

The preceding programs do not provide for the output of results or for the substitution of input by the user. This may be accomplished in ALGOL-8 through the READ and WRITE procedure statements.

1.2 ALGOL-8 Input/Output

ALGOL-8 programs may use either the Teletype console or the high-speed reader/punch as the input/output device for READ or WRITE statements. The device to be used and the information to be read or written is specified within parentheses after the input/output procedure statement. The integer 1 specifies the ASR 33 Teletype, 2 specifies the high-speed paper tape unit, as seen in the following examples.

READ (1,I)	The ALGOL-8 program will accept a value for the variable I from the Teletype keyboard or paper tape reader.
WRITE (2,A,B,C)	The ALGOL-8 program will punch the values for the variables A, B and C on the high-speed paper tape punch.

The variables must be declared before they are used in a READ or WRITE Procedure Statement.

1.3 Sample Programs

The following sample programs are provided to illustrate some of the features of ALGOL-8, and to acquaint the reader with the language in general.

1.3.1 Arithmetic Operators

ALGOL-8 programs may contain the arithmetic operators for addition (+), subtraction (-), multiplication (*), division (/), and exponentiation (↑). The following program combines the READ

```
'BEGIN' 'COMMENT' ARITHMETIC DEMO;
        'REAL' A,B,RSUM,RDIF,RPRD,RQUO;
        'INTEGER' I,J,ISUM,IDIF,IPRD,IQUO;
        WRITE (1, "TYPE TWO REAL NUMBERS");SKIP;
        READ (1, A,B);SKIP;
        RSUM:=A+B; WRITE (1,A," PLUS       ",B,"=",RSUM);SKIP;
        RDIF:=A-B; WRITE (1,A," MINUS      ",B,"=",RDIF);SKIP;
        RPRD:=A*B; WRITE (1,A," TIMES      ",B,"=",RPRD);SKIP;
        RQUO:=A/B; WRITE (1,A," DIVIDED BY",B,"=",RQUO);SKIP;
        WRITE (1, "TYPE TWO INTEGERS");SKIP;
        READ (1, I,J);SKIP;
        ISUM:=I+J; WRITE (1,I," PLUS       ",J,"=",ISUM);SKIP;
        IDIF:=I-J; WRITE (1,I," MINUS      ",J,"=",IDIF);SKIP;
        IPRD:=I*J; WRITE (1,I," TIMES      ",J,"=",IPRD);SKIP;
        IQUO:=I/J; WRITE (1,I," DIVIDED BY",J,"=",IQUO);SKIP;
'END'
$
```

Example A

```
TYPE TWO REAL NUMBERS
2,3

   0.200000$+01 PLUS          0.300000$+01=    0.500000$+01
   0.200000$+01 MINUS         0.300000$+01=   -0.100000$+01
   0.200000$+01 TIMES         0.300000$+01=    0.600000$+01
   0.200000$+01 DIVIDED BY    0.300000$+01=    0.666666$+00
TYPE TWO INTEGERS
2,3

     2 PLUS           3=      5
     2 MINUS          3=     -1
     2 TIMES          3=      6
     2 DIVIDED BY     3=      1
```

Example B

```
TYPE TWO REAL NUMBERS
89,765

   0.889999$+02 PLUS          0.765000$+03=    0.853999$+03
   0.889999$+02 MINUS         0.765000$+03=   -0.675999$+03
   0.889999$+02 TIMES         0.765000$+03=    0.680849$+05
   0.889999$+02 DIVIDED BY    0.765000$+03=    0.116339$+00
TYPE TWO INTEGERS
89,765

    89 PLUS          765=    854
    89 MINUS         765=   -676
    89 TIMES         765=  -1547
    89 DIVIDED BY    765=      0
```

Figure 1-1 ALGOL-8 Arithmetic Program

and WRITE statements with arithmetic operators. Two examples of program output are also included. Following the listing the program is described line by line.

Line 1 — The 'BEGIN' statement must be the first element of the ALGOL-8 program. The 'COMMENT' statement may be used to include comments to identify the program in the source listing. Comments are not typed during the running of the program.

Line 2 — All of the real number variables which are used within the ALGOL-8 program are declared with the 'REAL' statement.

Line 3 — The integer variables are declared with the statement 'INTEGER'.

Line 4 — The WRITE procedure statement is used to request user input. Any text included within double quotes is typed during the running of the program. The SKIP procedure statement is used to generate a new line for typed output. (It is equivalent to a carriage return and line feed on the Teletype console.)

Line 5 — The READ procedure statement accepts the user input to the ALGOL-8 program. The program will not continue until the user types two real numbers on the Teletype keyboard. Real number input may be terminated by typing any keyboard character other than a number, plus or minus sign, period, or dollar sign.

Lines 6,7,8,9	Once the input of two numbers is received, the ALGOL-8 program computes the various arithmetic combinations and types the results. The output of results is formatted by combining output of text (included within double quotes) and variables. Variables and text are separated within the WRITE statement by commas. Each line is terminated with a SKIP to allow the next output to start on a new line.
Line 10	The WRITE procedure statement on line 10 requests the user to input two integers. The SKIP generates a new line on which the input may be typed.
Line 11	The READ procedure statement accepts the user input of two integers. The integers may be terminated with any character other than a number.
Lines 12,13, 14,15	The arithmetic combinations of the integers are computed and typed in a similar manner to that for the real numbers above.
Line 16	The 'END' statement concludes the ALGOL-8 program statements.
Line 17	The ALGOL-8 program must be followed by a dollar sign as the first character on a line.

The running of the program produces the output shown in examples A and B of figure 1-1. The examples illustrate the differences in ALGOL-8 representation of real numbers and integers.

The input of values was in all cases terminated by a carriage return.

Real numbers are output in an exponential notation with the number separated by a dollar sign ($) from the exponential power (base 10) by which it is multiplied.

Examples:

-Ø.2ØØØØØ$ Ø1	equals	$-Ø.2ØØØØØ \times 1Ø^{1}$	equals -2.ØØØØØ
Ø.2ØØØØØ$-Ø5	equals	$Ø.2ØØØØØ \times 1Ø^{-5}$	equals Ø.ØØØØØ2

Integers are output as whole numbers, with up to four digits, and a minus sign if the number is negative.

Equivalent results for integer or real number input are obtained for the operations of addition and subtraction. Integer multiplication does not check for overflow, and as evident in example B, can yield incorrect results (e.g. 89 x 765 = -1547). Integer division differs from real division in that, since the result is not always an integer, the answer is always rounded off to the nearest integer.

As evident in example B, real number representations in ALGOL-8 are not always exact. For example, the number 89 is approximated as a real number in ALGOL-8 by the number 88.9999.

1.3.2 Compound Statements

The example of figure 1-2 uses the multiplication operator to generate the first ten powers of a number. The program employs a FOR statement to allow repeated execution of a grouping of ALGOL-8 statements. FOR statements have the form:

'FOR' E 'STEP' I 'UNTIL' F 'DO'S

where E is an assignment statement (e.g. V:=1)

I is an increment (e.g. 1)

F is a final value (e.g. 10)

S is a statement, compound statement, or block.

The expression after 'FOR' assigns an initial value to the variable. The statement following 'DO' is then executed for this assigned value. Once the statement is executed the variable is incremented by the value following 'STEP' and the statement following 'DO' is executed for the new value of the variable. The variable is repeatedly incremented and the statement is repeatedly executed for all values up to, and including, the value following 'UNTIL'.

In the program of figure 1-2, the 'DO' is followed by a compound statement. A compound statement is simply two or more ALGOL-8 statements enclosed in statement brackets ('BEGIN' and 'END'). This group of statements is executed as if it were one statement following the 'DO' There must be an equal number of 'BEGIN' symbols and 'END' symbols in every ALGOL-8 program.

The program simply requests a real number input and then repeatedly executes the compound statement which computes the powers of A. The statement X:=A*X computes the next power of A because X always equals the last power computed. Notice that X had to be set equal to 1 before the compound statement was executed.

```
'BEGIN' 'COMMENT' COMPUTE POWERS OF A;
        'INTEGER'V; 'REAL' A,X;
        WRITE(1,"FOR WHAT NUMBER WOULD YOU LIKE THE FIRST TEN POWERS?");
        X:=1;
        READ (1,A); SKIP;
        'FOR' V:=1 'STEP' 1 'UNTIL'10 'DO'
        'BEGIN'
                X:=A*X;
                WRITE (1,A," TO THE", V," EQUALS",X);
                SKIP;
        'END'
        SKIP;SKIP;
'END'
$
```

```
FOR WHAT NUMBER WOULD YOU LIKE THE FIRST TEN POWERS? 45.

   0.449999$+02 TO THE      1 EQUALS    0.449999$+02
   0.449999$+02 TO THE      2 EQUALS    0.202500$+04
   0.449999$+02 TO THE      3 EQUALS    0.911249$+05
   0.449999$+02 TO THE      4 EQUALS    0.410062$+07
   0.449999$+02 TO THE      5 EQUALS    0.184528$+09
   0.449999$+02 TO THE      6 EQUALS    0.830376$+10
   0.449999$+02 TO THE      7 EQUALS    0.373669$+12
   0.449999$+02 TO THE      8 EQUALS    0.168151$+14
   0.449999$+02 TO THE      9 EQUALS    0.756680$+15
   0.449999$+02 TO THE     10 EQUALS    0.340505$+17

FOR WHAT NUMBER WOULD YOU LIKE THE FIRST TEN POWERS? 987.

   0.986999$+03 TO THE      1 EQUALS    0.986999$+03
   0.986999$+03 TO THE      2 EQUALS    0.974169$+06
   0.986999$+03 TO THE      3 EQUALS    0.961504$+09
   0.986999$+03 TO THE      4 EQUALS    0.949005$+12
   0.986999$+03 TO THE      5 EQUALS    0.936668$+15
   0.986999$+03 TO THE      6 EQUALS    0.924491$+18
   0.986999$+03 TO THE      7 EQUALS    0.912472$+21
   0.986999$+03 TO THE      8 EQUALS    0.900610$+24
   0.986999$+03 TO THE      9 EQUALS    0.888902$+27
   0.986999$+03 TO THE     10 EQUALS    0.877347$+30
```

Figure 1-2 Program to Compute First Ten Powers

1.3.3 Blocks

The following sample program plots linear equations using a block to perform the actual plotting. An ALGOL-8 block is simply a compound statement which contains declarations.

```
    .
    .
    .
'BEGIN'
    'INTEGER' I,J,K;
    'REAL' A,B,C;
    'BOOLEAN' N,M;
     .
     .
     .
    Statement
    Statement
     .
     .
     .
'END'
  .
  .
  .
```

The declarations made within a block are not applicable outside the block. Thus local variables may be assigned which have a value while the program is executing a given block but which are not defined in an outer block. The use of the block is illustrated in the program example of figure 1-3.

The first part of the program requests and accepts the real coefficient and the real constant term of the linear equation using WRITE and READ procedure statements. The horizontal axis for the values of the equation is established with four WRITE statements. The FOR statement is then used to plot the equation using values of X from -10 to 10.

The block begins by declaring the variables which are used exclusively within the block. The use of local variables in large

programs allows the ALGOL-8 programmer to efficiently use the finite number of variables allowed in ALGOL-8, by freeing them upon exit from the block.

Boolean variables are used to test for points which are outside the range of the values.

Since the solution of the equation is a real number and the number of spaces must be an integer (the upper bound of a FOR statement must be an integer), the function ENTIER is used. ENTIER converts a real number, N, to the greatest integer, I (positive or negative), such that I is less than or equal to N (e.g. ENTIER (+5.9) =5; ENTIER (+6.Ø) =6; ENTIER (-4.32) = -5). The reader should note that ENTIER is neither truncation nor rounding off, although in some cases they will produce the same results.

A FOR statement within the block is used to plot the number of spaces up to the point. In this case, the upper limit of the FOR statement is an integer variable, rather than a specific integer, allowing different spacing for each point plotted.

The program loops to its beginning with a GOTO statement which permits the program to plot a series of equations without being restarted.

A sample running of the program is also shown in figure 1-3. Notice that although the equation is linear, in this case the graph only approximates linearity because "partial spaces" cannot be plotted.

```
'BEGIN' 'COMMENT' LINEAR PLOT PROGRAM;
        'INTEGER'  X;
        'REAL' A1,B;
   LIN: WRITE (1, "I'LL PLOT YOUR LINEAR EQUATION FOR YOU!");SKIP;
        WRITE(1,"WHAT IS THE COEFFICIENT OF X?");SKIP;
        READ (1,A1);SKIP;
        WRITE (1,"WHAT IS THE CONSTANT TERM?");SKIP;
        READ (1,B);
        SKIP;SKIP;SKIP;
        WRITE (1,"          -30        -20        -10         0");
        WRITE (1,"         10         20         30");SKIP;
        WRITE (1,"           ↑          ↑          ↑          ↑");
        WRITE (1,"           ↑          ↑          ↑");SKIP;
        'FOR'X:=-10 'STEP'1 'UNTIL'10 'DO'
        'BEGIN' 'COMMENT' PLOT BLOCK;
                'INTEGER' V,PLOT;
                'REAL' CONST;
                'BOOLEAN' B1,B2;
                WRITE (1,X);
                CONST:=X*A1+B+34;
                PLOT:=ENTIER (CONST);
                B1:=PLOT<2; B2:=PLOT>65;
                'IF' B1 'THEN' 'GOTO' POINT'ELSE'
                'IF'  B2 'THEN'PLOT:=66;
                'FOR'V:=1 'STEP'1 'UNTIL'PLOT 'DO'WRITE (1," ");
          POINT:WRITE (1,"*");SKIP;
        'END'
        SKIP;SKIP;SKIP;'GOTO'LIN
'END'
$

I'LL PLOT YOUR LINEAR EQUATION FOR YOU!
WHAT IS THE COEFFICIENT OF X?
0.66667

WHAT IS THE CONSTANT TERM?
2

          -30        -20        -10         0         10         20         30
           ↑          ↑          ↑          ↑          ↑          ↑          ↑
   -10                                      *
    -9                                      *
    -8                                       *
    -7                                        *
    -6                                        *
    -5                                         *
    -4                                          *
    -3                                          *
    -2                                           *
    -1                                            *
     0                                             *
     1                                             *
     2                                              *
     3                                               *
     4                                               *
     5                                                *
     6                                                 *
     7                                                 *
     8                                                  *
     9                                                   *
    10                                                   *


I'LL PLOT YOUR LINEAR EQUATION FOR YOU!
WHAT IS THE COEFFICIENT OF X?
```

Figure 1-3 Linear Pl ot Program

Chapter 2

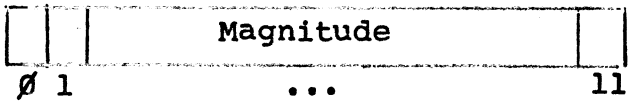
REPRESENTATION OF ELEMENTS

2.1 Integers

Integers are signed numbers (the positive sign is not always expressed), written without a decimal point, in the range ± 2040.

Division between integers produces a real result. In ALGOL-8, division between integers yields the closest integer. For example, if J=8 and K=3, the division of J by K gives 2.6666 and the result in ALGOL-8 is 3.

Integers are represented with one 12-bit word. The first bit (bit Ø) is the sign bit and bits 1 through 11 contain the integer magnitude.



BitØ = Sign

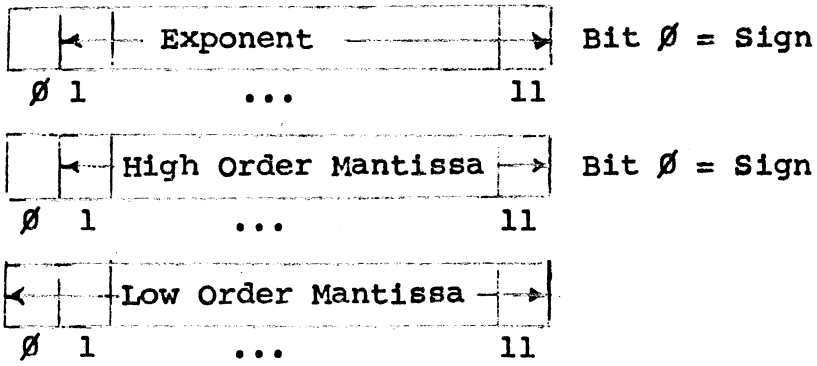
2.2 Real Numbers

Real numbers are signed numbers (the positive sign is not always expressed) of the decimal exponent form in the range ± 0.32$617 to ± 0.31$-616 ($ indicates the multiplication by a power of 10).

The forms represented by the following examples are allowed in ALGOL-8 programs:

-0.1234	.5	1.2$-2	(=.012)
5.0	-2.3	-0.1$10	(=-1,000,000,000)

Real numbers are represented by three 12-bit words. The first word represents the exponent and its sign. The second word represents the higher order bits of the mantissa, and the sign of the mantissa. The third word represents the lower order bits of the mantissa.



2.3 Basic Symbols

Basic symbols are the predefined elements of an ALGOL-8 program. Long basic symbols are typed enclosed in single quotes, while short basic symbols stand alone. Appendix A contains all the permitted Basic Symbols and their representation in ALGOL-8.

2.3.1 Long Basic Symbols

Some symbols and logical operators are represented by a string of two or more letters enclosed in single quotes. They are identified by their first two letters only, e.g., 'BE' and 'BEGIN' are equivalent.

Examples: 'BEGIN' 'INTEGER' 'SWITCH'

2.3.2 Short Basic Symbols

Other basic symbols are represented by a single character or, exceptionally, by two successive characters (a character followed by an equals sign).

Examples:	*	multiplication
	↑	exponentiation
	<=	less than or equal to

2.4 Identifiers

An identifier is a name given to a variable in an ALGOL-8 program. An identifier must begin with a letter and may contain only letters and numerals. The length of an identifier is not limited but only the first four alphanumeric characters are significant. For example, CAR15, CAR18 and CAR1 are interpreted as the same identifier. For this reason it is good practice to limit identifiers to four characters.

2.5 Constants

Integer constants (and the whole number part of real constants), must have an absolute value less than or equal to 2040, when their value is given explicitly in an ALGOL-8 program.

Examples:	Integers	Real Numbers
	27	1.0
	-2035	235.0$50
	5	-2.48

2.6 Variables

ALGOL-8 variables are quantities which are referred to by name (by its identifier) and are able to take on different values. Variables may be Boolean, integer or real type. Boolean variables may not be subscripted. The possible values of a Boolean variable are true and false.

Integer or real variables may be subscripted. (A subscripted variable is a real or integer one-dimensional array, as described in paragraph 2.7.)

Simple variables (not subscripted) have identifiers as described in paragraph 2.4 (e.g. A, SUM, INT5).

2.7 Subscripted Variables and Arrays

Subscripted variables permit the representation of many quantities with one identifier. A subscripted variable is one dimensional array (the only type of array allowed in ALGOL-8). It may be an array of integers or real numbers.

Subscripted variables have identifiers folllwed by brackets enclosing the subscript (e.g. SUM [I], X[J]). The subscript is an integer and its lowest value must be either Ø or 1.

NOTE: The brackets [= shift/K and] = shift/M on the ASR 33 Teletype Keyboard.

2.8 Statements

Statements are commands to the computer to carry out operations. ALGOL-8 statements must be terminated with a semicolon.

Examples:

```
'GOTO' START;
'IF' A=B 'THEN' X:=1 'ELSE' X:=Ø;
```

Compound statements are several statements grouped together and enclosed between the statement brackets 'BEGIN' and 'END'.

2.9 Labels

A label is used in an ALGOL-8 program to reference or identify statements in a program. It must be an identifier and must be followed by a colon.

Example:

```
START:  READ (1, I);
```

Chapter 3

EXPRESSIONS

3.1 Arithmetic Expressions

Arithmetic expressions may include constants, variables, operators and functions. For example, the arithmetic expression:

$$\frac{-B + \sqrt{(B^2 \quad -4AC)}}{2A}$$

may be written as an ALGOL-8 arithmetic expression

```
(-B + SQRT (B ↑ 2-4*A*C))/(2*A)
```

where 2 and 4 are constants; A, B, and C are variables; *, ↑ , -, and + are operators; and SQRT is a function.

3.1.1 Constants

Constants used in arithmetic expressions may be either integer or real.

Integers may be of the following forms (i.e. $I \leq 2040$).

```
5          -7          2018
```

Real numbers of the following form may be used in arithmetic expressions.

$\pm$ a.b$\pm$c

where a = whole number part (≤ 2040)

b = fractional part (6 digits or less)

c = exponent, base 10 ($-616 \leq c \leq 617$)

3.1.2 Variables

A variable used in an arithmetic expression is one or more alphanumeric characters which obey the following rules.

a) The first character must be a letter.

b) Spaces are ignored.

3) Only the first four characters are significant, and therefore must be unique.

3.1.3 Operators

The operators in arithmetic expressions are symbols representing the common arithmetic operations, plus left and right parentheses which serve as delimiters.

Arithmetic expressions in ALGOL-8 are evaluated from left to right with an an additional priority rule for operators.

Priority	Symbols
1	(,)
2	↑
3	*, /
4	+, -

Thus exponentiation (↑) is performed before either multiplication or division, which are performed before addition or subtraction. Parentheses have the highest priority and may be used to give priority to lower operators.

Operations of the same level are performed left to right.

Example: A*B + C/D - E*F↑G

= (A*B) + (C/D) - (E* (F↑G))

Two arithmetic operators must not be written side by side. Thus A/-B is not a legal ALGOL-8 arithmetic expression while A/ (-B) is legal. All signed exponents must be enclosed in parentheses to

separate the exponentiation sign (↑) from the sign of the exponent (+ or -).

Division (/) and exponentiation (↑) are considered real operators. For small powers of a variable such as X↑2, or X↑3, it is preferable to use X*X, or X*X*X, so that the functions need not be loaded (see paragraph 8.2). The exponentiation operator (↑) in ALGOL-8 is undefined when the variable is zero or a fraction. The programmer must therefore specially treat such cases. The following examples demonstrate the importance of correctly writing the arithmetic expression. Different placements of parentheses can yield very different results.

ALGOL EXPRESSION		ALGEBRAIC EQUIVALENT
A + B * X - C/ X↑2	=	$A + BX - \frac{C}{X^2}$
(A + B) * X -(C/X)↑2	=	$AX + BX - \frac{C^2}{X^2}$
(A + B) *(X - C) /X↑2	=	$\frac{AX + BX - AC - BC}{X^2}$
A + (B* X-C) / X↑2	=	$A + \frac{B}{X} - \frac{C}{X^2}$

3.1.4 Mixed Mode Expressions

Variables of real and integer type may be mixed in an expression. Integers are converted to reals before calculations. The form of the expression may affect the result.

Example: (A is real, I and J are integers)

a) A+I+J
I is converted to real and added to A, then J is converted to real and added to the sum.

b) A+(I+J)
I and J are added as integers converted to real and added to A.

3.1.5 Standard Functions

The following standard functions are provided in ALGOL-8. The argument must be a non-dimensioned real - variable except the argument of REAL. Constants are not allowed as arguments (e.g.: Y:=SIN (.5); is illegal, but PHI=.5; Y:=SIN (PHI); is legal).

SQRT (x)	-	Square Root of x
SIN (x)	-	Sine of x in radians
COS (x)	-	Cosine of x in radians
ARCT (x)	-	Arctangent of x
EXP (x)	-	Exponential of x
LN (x)	-	Logarithm (natural) of x
ENTIER (x)	-	Largest integer not greater than x
REAL (I)	-	Converts integer I to a real number
SIGN (x)	-	Integer { 1 for positive x 0 for zero x -1 for negative x
ABS (x)	-	Absolute value, or modulus of x

The program of figure 3-1 illustrates one use of functions in ALGOL-8, to generate a listing of values. A later example (paragraph 5.3) uses functions to generate a table of values. ALGOL-8 functions may be used within expressions and calculations as well, as long as the argument of the function is a real variable and not a constant.

3.2 Boolean Expressions

ALGOL-8 programs may use the following Boolean operators to form Boolean expressions.

'TRUE'	used to assign a logical value to a Boolean variable.
'FALSE'	used to assign a logical value to a Boolean variable.
'NOT'	used to negate a Boolean variable
'AND'	used to combine Boolean variables
'OR'	used to combine Boolean variables
'IMP'	implies - used to combine Boolean variables
'EQU'	equivalence - used to combine Boolean variables.

```
'BEGIN' 'COMMENT' FUNCTION COMPUTER;
        'REAL' A,B; 'INTEGER' I;
        SKIP;
        START: READ (1,A);SKIP;
        WRITE (1, "FOR A VALUE OF",A);SKIP;
        WRITE (1, "        SINE        COSINE        ARCTANGENT");SKIP;
        B:= SIN(A); WRITE (1,B);
        B:= COS(A); WRITE (1,B);
        B:= ARCT(A); WRITE (1,B);SKIP;
        WRITE (1, "      NAT LOG       EXPONENTIAL     SQR ROOT");SKIP;
        B:=LN(A); WRITE (1,B);
        B:=EXP (A); WRITE (1,B);
        B:= SQRT (A); WRITE (1,B);SKIP;
        WRITE (1,"   INTEGER PART     REAL PART       SIGN");
        WRITE (1,"            ABSOLUTE VALUE");SKIP;
        I:=ENTIER (A); WRITE (1,I);
        B:=REAL (I); WRITE (1,"        ",B);
        B:=SIGN (A); WRITE (1,B);
        B:=ABS (A); WRITE (1,B);SKIP;SKIP;
        'GOTO' START
'END'
$

0.01

FOR A VALUE OF   0.100000$-01
        SINE          COSINE         ARCTANGENT
   0.999982$-02   0.999949$+00   0.999965$-02
      NAT LOG       EXPONENTIAL     SQR ROOT
  -0.460516$+01   0.101004$+01   0.999999$-01
    INTEGER PART     REAL PART        SIGN          ABSOLUTE VALUE
      0            0.000000$+00   0.100000$+01   0.100000$-01

-12.5731

FOR A VALUE OF  -0.125731$+02
        SINE          COSINE         ARCTANGENT
  -0.672908$-02   0.999977$+00  -0.149142$+01
      NAT LOG       EXPONENTIAL     SQR ROOT
   0.253155$+01   0.346395$-05   0.354585$+01
    INTEGER PART     REAL PART        SIGN          ABSOLUTE VALUE
    -13           -0.129999$+02  -0.100000$+01   0.125731$+02

1.0$-610

FOR A VALUE OF   0.999961$-610
        SINE          COSINE         ARCTANGENT
   0.309548$-563   0.999999$+00   0.238813$-610
      NAT LOG       EXPONENTIAL     SQR ROOT
  -0.140457$+04   0.400000$+612   0.999980$-305
    INTEGER PART     REAL PART        SIGN          ABSOLUTE VALUE
      0            0.000000$+00   0.100000$+01   0.999961$-610

3.141592

FOR A VALUE OF   0.314159$+01
        SINE          COSINE         ARCTANGENT
   0.000000$+00  -0.999999$+00   0.126262$+01
      NAT LOG       EXPONENTIAL     SQR ROOT
   0.114472$+01   0.231406$+02   0.177245$+01
    INTEGER PART     REAL PART        SIGN          ABSOLUTE VALUE
      3            0.300000$+01   0.100000$+01   0.314159$+01
```

Figure 3-1 Function Computer Program

For all combinations of values for Boolean variables B1 and B2, the following table gives the values of the above combinations.

B1	B2	'NOT'B1	B1'AND'B2	B1'OR'B2	B1'IMP'B2	B1'EQU'B2
true	true	false	true	true	true	true
true	false	false	false	true	false	false
false	true	true	false	true	true	false
false	false	true	false	false	true	true

3.2.1 Forms

Boolean expressions may be of the following forms:

a) A single Boolean variable optionally preceded by the basic symbol 'NOT'.

'NOT'B1 where B1 is declared Boolean

b) Two Boolean variables connected by one of the four logical operators ('AND', 'OR', 'IMP', or 'EQU').

```
B3:=B1 'AND' B2;
B4:=B1 'OR'  B3;
B5:=B1 'IMP' B2;
B6:=B4 'EQU' B2;
```

where B1, B2, B3, B4, B5, and B6 are declared Booleans.

c) A relation comprising two simple integer arithmetic expressions connected by one of the six relational operators =, #, $\geq$, $\leq$, $>$, or $<$.

```
X+Y+Z>= 5
A#B          where A,B,X,Y, and Z are integers.
```

d) A relation comprising two simple real arithmetic expressions connected by one of the two relational operators $\prec$ or $\succ$.

X+Y+Z $\succ$ 5

A $\prec$ B+1 where A,B,X,Y, and Z may be integers or real numbers.

A Boolean expression can be used directly in a conditional statement or can be used to assign a logical value to a Boolean variable as seen in the two following examples.

```
'IF' I-J>K 'THEN' 'GOTO' START;
B1:=I-J≺5*K; 'IF' B1 'THEN' 'GOTO' START;
```

In the second example above, B1 is true when I-J $\prec$ 5*K;. B1 is false otherwise. The GOTO statement is performed when B1 is true.

3.2.2 Limitations

a) There are no Boolean arrays in ALGOL-8.

b) The multiple assignment of a Boolean expression value to a number of Boolean variables is not allowed. For example, "B1:=I>J; B2:=I>J;" is acceptable while "B1:=B2:=I>J;" is not acceptable.

c) The logical values 'TRUE' and 'FALSE' can be used only to assign logical values to Boolean variables.

B1:= 'TRUE'; is permitted, while

I-J> K:= 'FALSE' is not permitted.

Chapter 4

DECLARATIONS

Declarations are used to specify the properties of quantities used in an ALGOL-8 program and to associate these quantities with identifiers. Declarations must appear at the beginning of a block (i.e. before any statements which cause operations to be performed by the block).

4.1 Variables

Variables may be real numbers, integers or Boolean. Multiple variables of the same type may be declared in one statement by separating the variables by commas and terminating the statement by a semicolon.

Example:

```
'BEGIN'
        'INTEGER' N;
        'REAL' A,B,C;
        'BOOLEAN' B1,B2;
              .
              .
              .
'END'
$
```

4.2 Arrays

ALGOL-8 arrays may be either real arrays or integer arrays. (The type of array, integer or real, must be declared also.) The lower bound of the array must be either Ø or 1. The upper bound must be an unsigned integer or a simple integer variable, as seen in the example below. The bounds of the array are enclosed in square brackets and separated by a colon.

Example:

```
'BEGIN'
        'INTEGER' N;
        'REAL'  'ARRAY'  AX[1:10], AY[Ø:12];
           .
           .
           .
        'BEGIN'
              'INTEGER'  'ARRAY'  INTX[1:N];
           .
           .
           .
        'END'
'END'
$
```

As illustrated in the preceding example, if the upper bound of an array is an integer variable it must be declared and assigned a value in a preceding block prior to its use as an array bound.

4.3 Switches

Switches may be used in an ALGOL-8 program to conditionally transfer control to other areas of the program. The switch is established by the declaration 'SWITCH'. The switch is used to conditionally transfer control with a GOTO statement.

Example:

```
'BEGIN'
        'INTEGER' I;
        'SWITCH' CHANGE:=BPOS,BNEG,BZERO;
           .
           .
           .
        'GOTO' CHANGE [I];
        BPOS:Statement P1;
             Statement P2;
                .
                .
                .
        BNEG:Statement N1;
             Statement N2;
                .
                .
                .
        BZERO:Statement Z1;
              Statement Z2;
                .
                .
                .
'END'
$
```

The statements identified by the labels included in the switch declaration are executed depending upon the value of I. If I=1, the statements beginning with P1 (labeled BPOS) are executed. If I=2, the statements beginning with N1 (labeled BNEG) are executed. If I=3, the statements beginning with Z1 (labeled BZERO) are executed.

The statements to which control is transferred may be simple statements, compound statements or blocks.

If I≠1,2, or 3, the statement 'GOTO' CHANGE [I]; is undefined.

4.4 Procedures

There are no user procedures in ALGOL-8.

Chapter 5

STATEMENTS

The units of operation within the language are called statements.

Statements may be of the following types:

1) Assignment statements
2) GOTO statements
3) Conditional statements
4) FOR statements
5) Dummy statements

5.1 Assignment statements

An assignment statement relates a variable V to an arithmetic expression E by means of the symbol:=, as in the following example:

```
V:= E;
```

The above statement means that the value of expression E replaces the value of variable V.

Multiple assignment statements can be done when more than one variable is to be given the value of the same expression. You may write:

```
V1:= V2:= V3:= E;
```

If the type of the variable (i.e. integer or real) is not the same as the expression, the result of this expression will be converted into the type of the variable.

5.2 GOTO statements

A GOTO statement may be of the following forms:

a) 'GOTO' LABEL;

b) 'GOTO' SWITCH [INDEX] ;

where LABEL is used to reference a point in the ALGOL program either within the block of the GOTO statement or in an outer block, SWITCH is a declared switch designator, and INDEX must be a simple integer variable or an unsigned integer.

Example:

```
'BEGIN' 'COMMENT' BLOCK 1;
     'INTEGER' I;
START: READ (1,I);
           'BEGIN' 'COMMENT' BLOCK 2;
                'SWITCH' POINT := LOOP,EXIT;
                'GOTO' POINT [I];
           LOOP:WRITE (1,"LOOP");SKIP;
                'GOTO' START
           'END' BLOCK 2;
EXIT:WRITE (1,"EXIT");SKIP;
'END' BLOCK 1
$
```

In this example, 'GOTO' START leads from the inner block (2) into the outer block (1) 'GOTO' POINT [I] is equivalent to 'GOTO' LOOP for I=1, or 'GOTO' EXIT for I=2. If the value of I is less than 1 or greater than 2, the 'GOTO' POINT [I] statement is undefined.

5.3 FOR statements

The FOR statement is in the following form:

'FOR' V:= A 'STEP' B 'UNTIL' C 'DO' S

where the controlled variable V must be a simple integer variable, the elements A, B, and C can be in the following forms:

a) a signed (minus only) or unsigned integer

b) a simple integer variable

and S is a statement

The FOR statement can be self-imbedded since the statement S may itself be a FOR statement.

NOTE: The 'WHILE' basic symbol of the ALGOL language is not permitted in ALGOL-8. Thus ALGOL-8 programs may not use 'WHILE' to terminate FOR statements.

```
'BEGIN' 'INTEGER' I; 'REAL' F,R,C,S;
        WRITE (1,"   DEG        SIN                COS");
        SKIP;SKIP;
        F:=3.14159/180;
        'FOR' I:=0 'STEP' 10 'UNTIL' 90 'DO'
        'BEGIN' R:=I*F;
                C:=COS(R);
                S:=SIN(R);
        WRITE (1,I,S,C);SKIP
        'END'
'END'
$
```

```
 DEG        SIN              COS

  0    0.000000$+00    0.999999$+00
 10    0.173648$+00    0.984807$+00
 20    0.342019$+00    0.939692$+00
 30    0.499999$+00    0.866025$+00
 40    0.642787$+00    0.766044$+00
 50    0.766043$+00    0.642787$+00
 60    0.866024$+00    0.500000$+00
 70    0.939692$+00    0.342020$+00
 80    0.984807$+00    0.173648$+00
 90    0.999999$+00    0.953673$-06
```

Figure 5-1 A FOR statement used in a Program

5.4 Conditional statements

A conditional statement may be of the following forms:

a) 'IF' B1 'THEN' S1

b) 'IF' B1 'THEN' S1 'ELSE' S

c) 'IF' B1 'THEN' S2

where B1 is a Boolean expression of one of the forms given in Chapter 3.

S1 is an unconditional statement, S2 is a FOR statement, S is a statement.

The conditional statement can be self-embedded since the statement S may itself be a conditional statement.

The embedding levels of FOR statements and conditional statements are interdependent (controlled by the same stack). The total number of conditional and FOR statements that can be accommodated at one time is 18.

```
'BEGIN' 'COMMENT' CONDITIONAL DEMO;
        'INTEGER' I,J;
  START:READ (1,I,J);
        'IF'I>J 'THEN'WRITE (1,"FIRST NUMBER IS GREATER.");
        'IF'I<J 'THEN' WRITE (1,"SECOND NUMBER IS GREATER.")'ELSE'
        WRITE (1,"THEY ARE EQUAL.");
        'GOTO' START;
'END'
$
```

Figure 5-2 A Conditional Statement used in a Program

5.5 Dummy Statements

A dummy statement produces no executable code. It may be used to place a label.

The dummy statement is allowed just before the basic symbol 'END' to label it or to end the last statement of a compound statement or a block, by a semicolon.

Example:

```
'BEGIN'
      'REAL' A,B;
      'INTEGER' I,J;
      READ (1,A,I);
         .
         .
         .
      'BEGIN'
            'IF' I=J 'THEN' 'GOTO' EXIT;
               .
               .
               .
            B:=A*A*A;
      'END'
      EXIT:
'END'
```

Dummy statements

In this example, there are dummy statements before each basic symbol 'END'. The semicolon after "B:=A*A*A" is unnecessary and is considered a dummy statement. The last basic symbol 'END' is labeled by the dummy statement EXIT.

5.6 Comments

Comments are added to ALGOL-8 programs for the user's convenience. They may be used to identify the function of statements, blocks, etc., within an ALGOL-8 program. They produce no executable code.

There are two forms of comments:

a) Comments may be introduced between declarations or statements by using the basic symbol 'COMMENT'.

These comments may have any length and may contain all characters except the semicolon, which necessarily stops the comment.

b) Comments may be introduced after the basic symbol 'END' without using the symbol 'COMMENT'.

These comments are ended by one of the following three basic symbols:

; , 'END' or 'ELSE'

or the dollar sign at the end of the program.

The comment length is also unlimited but, in addition to the semicolon, the single quote is not an allowed character. (it serves to form the basic symbol 'END' or 'ELSE' which stops the comment).

Example:

```
'BEGIN' 'COMMENT' TEST DYNAMIC ARRAY;
        'INTEGER' I,J,K;
        READ (1,I,J);
        'COMMENT' CALCULATION OF SIZE;
        K:I+J;
              'BEGIN' 'COMMENT' K IS USED AS THE;
                      'COMMENT' UPPER BOUND OF THE ARRAY;
                        'ARRAY' TABLE [0:K];
                        'FOR' J:=0 'STEP' 1 'UNTIL' K 'DO'
                        .
                        .
                        .
                 'END' BLOCK 2
'END' TEST;
$
```

The semicolon which stops the comment following the last basic symbol 'END' of the program is optional.

5.7 Sample Program

The following sample program illustrates all of the statements introduced in this chapter. The program is designed to plot linear, quadratic, and cubic equations. A switch is used to handle the three different cases.

A conditional statement with Boolean variables is used to determine that the equation can be plotted by the program, i.e. it is linear, quadratic, or cubic. A FOR statement controls the number of executions of the block which plots the equation.

```
I WILL PLOT YOUR EQUATION FROM X=-10 TO X=10.
WHAT IS THE HIGHEST POWER OF X?
3
I'LL PLOT YOUR CUBIC EQUATION!
WHAT IS THE COEFFICIENT OF X CUBED ?
0.1
WHAT IS THE COEFFICIENT OF X SQUARED ?
0.5
WHAT IS THE COEFFICIENT OF X?
-5.
WHAT IS THE CONSTANT TERM?
-2.
```

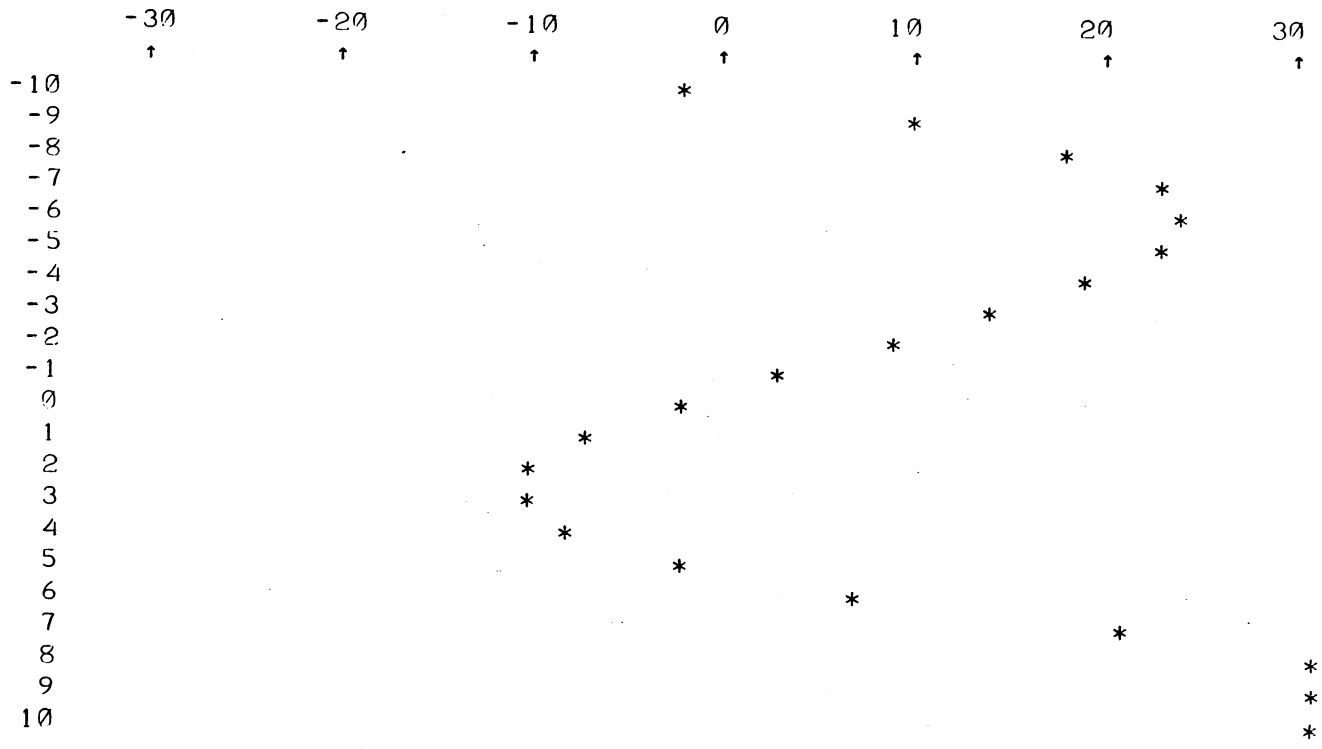


Figure 5-3 Sample Program Output

```
'BEGIN' 'COMMENT' PLOT PROGRAM;
        'INTEGER' XHIGH, X;
        'REAL'A1,A2,A3,B;
        'BOOLEAN'HIGH,LOW;
        'SWITCH'CONTROL:=LIN,QUAD,CUBIC;SKIP;
        WRITE (1,"I WILL PLOT YOUR EQUATION FROM X=-10 TO X=10."); SKIP;
   ASK: WRITE (1,"WHAT IS THE HIGHEST POWER OF X?");SKIP;
        READ (1,XHIGH);SKIP;
        HIGH:=XHIGH>3; LOW:=XHIGH<1;
        'IF'HIGH 'OR' LOW 'THEN'
        'BEGIN'
                WRITE (1,"I'M SORRY, I'M NOT SMART ENOUGH TO ");SKIP;
                WRITE (1,"PLOT", XHIGH,"TH DEGREE EQUATIONS!");SKIP;
                WRITE (1,"WOULD YOU LIKE TO TRY AGAIN?");SKIP;
                'GOTO' ASK;
        'END'
        'ELSE' 'GOTO'CONTROL [XHIGH];
   LIN: WRITE (1, "I'LL PLOT YOUR LINEAR EQUATION FOR YOU!");SKIP;
        A2:=A3:=0.0;
        'GOTO' ONE;
  QUAD: WRITE (1,"I'LL PLOT YOUR QUADRATIC EQUATION FOR YOU");SKIP;
        A3:=0.0;
        'GOTO' TWO;
  CUBIC:WRITE (1, "I'LL PLOT YOUR CUBIC EQUATION!");SKIP;
        WRITE (1,"WHAT IS THE COEFFICIENT OF X CUBED ?");SKIP;
        READ (1, A3);SKIP;
   TWO: WRITE (1, "WHAT IS THE COEFFICIENT OF X SQUARED ?");SKIP;
        READ (1, A2);SKIP;
   ONE: WRITE(1,"WHAT IS THE COEFFICIENT OF X?");SKIP;
        READ (1,A1);SKIP;
        WRITE (1,"WHAT IS THE CONSTANT TERM?");SKIP;
        READ (1,B);
 GRAPH: SKIP;SKIP;SKIP;
        WRITE (1,"          -30        -20        -10          0");
        WRITE (1,"         10         20         30");SKIP;
        WRITE (1,"           ↑          ↑          ↑          ↑");
        WRITE (1,"           ↑          ↑          ↑");SKIP;
        'FOR'X:=-10 'STEP'1 'UNTIL'10 'DO'
        'BEGIN''COMMENT' PLOT BLOCK (NO. SPACES TO ORIGIN=34);
                'INTEGER' V,PLOT;
                'REAL' CONST;
                'BOOLEAN' B1,B2;
                WRITE (1,X);
                CONST:=X*X*X*A3+X*X*A2+X*A1+B+34;
                PLOT:=ENTIER (CONST);
                B1:=PLOT<2; B2:=PLOT>65;
                'IF' B1 'THEN' 'GOTO' POINT'ELSE'
                'IF'  B2 'THEN'PLOT:=66;
                'FOR'V:=1 'STEP'1 'UNTIL'PLOT 'DO'WRITE (1," ");
          POINT:WRITE (1,"*");SKIP;
        'END'
        SKIP;SKIP;SKIP;'GOTO'ASK
'END'
$
```

Figure 5-4 Sample ALGOL-8 Program

Chapter 6

INPUT/OUTPUT STATEMENTS

ALGOL-8 programs may use either the ASR-33 Teletype or a high speed paper tape reader/punch as an input/output device.

6.1 I/O Statement Form

Input/Output statements in ALGOL-8 are of the form:

Function (Unit, Variable List)

where

Function	=	READ or WRITE
Unit	=	1 (ASR-33 Teletype) or 2 (high speed reader/punch).
Variable List	=	One or more simple or subscripted variables, separated by commas.

Example:

```
'BEGIN'
'INTEGER' S; 'INTEGER' 'ARRAY' A[1: 1Ø];
     .
     .
     .
    'FOR' S:=1 'STEP'1 'UNTIL' 10 'DO'
    READ (2, A[S]);
     .
     .
     .
'END'
```

The above READ statement means that the 10 values of the integer array A will be read from paper tape on the high speed reader. The ALGOL-8 program which executes this statement will advance the paper tape and accept the values of these 1^0 variables in the USASCII code.

6.2 Output Format

There are no ALGOL-8 format statements. The output formats for integers and real numbers are as follows:

Integers - space, sign and up to 4 digits.

e.g. -1234

Real numbers - 2 spaces, sign (if negative), Ø,decimal point, 6 digits, $, sign, 2 or 3 digits

e.g. -Ø.123456$+Ø7

6.3 Input Format

The input format for ALGOL-8 programs is free form (a number may be typed in any form as long as it does not exceed the range of the variable type). Input to the operating system may be in parity or non-parity USASCII format (paper tape or keyboard initiated). Blank tape, rubout and line-feed characters are ignored. Spaces are also ignored except in text strings (see 6.4 Text Output).

Input of data to the operating system in response to a READ procedure statement may be terminated by any character which is not a number, sign, decimal point or dollar sign (indicating the exponent). A ← (SHIFT/O) typed before a terminator deletes the item being input.

6.4 Text Output

Alphanumeric characters included within double quotes may be elements of the variable list of a WRITE statement. The WRITE statement will type all characters which are included in the double quotes. Thus the WRITE statement may output text

or format data by supplying spaces within quotes.

Example:

```
WRITE (1, "X= ", X, "   XPRIME= ", XP);
```

6.5 New Line

The output of text and data may be formatted on separate lines by the SKIP statement, which causes the Teletype carriage to be returned and a new line of output supplied. The following example prints the values of X, Y, and Z under their respective headings.

Example:

```
WRITE (1, "  X     Y     Z ");
SKIP;
WRITE (1 , X, Y, Z);
```

Chapter 7

DIAGNOSTICS

7.1 Compile time diagnostics

Whenever an error is detected during the compilation of an ALGOL-8 program, the compiler prints an error message.

After this detection, no recovery is possible. The user must correct the error, reinitialize the compiler, and restart.

An error message is printed as follows:

xx yy zz

xx is the diagnostics code, yy is the line number where the error has been detected and zz is the character number in this line.

Generally, line numbers and character numbers will quickly determine the origin of an error. But it happens that the discovery of an error may occur after the latter. For example, in the following line:

'BEGIN' 'INTEGER I, J, K,; 'REAL' A, B, C;

A quote is missing after INTEGER but it would not be found before the detection of a second quote after REAL. Thus the character number would indicate an error after REAL, rather than the actual error after INTEGER.

The following is a list of the error codes typed by ALGOL during compilation.

Ø Long Basic Symbol Error

*The first single quote of a long basic symbol is not found.

*The long basic symbol is not found.

1 Identifier Error

*An identifier is not declared in a statement.

*An identifier does not begin with a letter.

*A left bracket is omitted in a subscript variable of a statement.

2 Declaration Error

*A declaration is not ended by a semicolon.

*The long basic symbol 'REAL' or 'INTEGER' is followed by a long basic symbol other than 'ARRAY'.

*The program contains a Boolean array declaration which is not allowed in ALGOL-8.

*A left bracket or a right bracket is omitted in an array declaration.

*The colon separating the lower and upper bounds is omitted.

*The upper bound is an integer variable which is declared in the same block of the array identifier.

*The lower bound is neither Ø nor 1.

3 Multiple Declaration Error

*The same identifier is declared more than once in the same block.

4 Statement Error

*The long basic symbol beginning a statement is not one of these following long basic symbols:

'BEGIN'	(compound statement or block)
'IF'	(conditional statement
'FOR'	(for statement)
'GOTO'	(go to statement)
'END'	(dummy statement)

*The input-output procedure statement READ, WRITE, or SKIP is not found.

*A conditional statement is found just after the long basic symbol 'THEN'.

*The statement following the long basic symbol 'ELSE' does not correspond with the unconditional statement following the long basic symbol 'THEN'.

5 Statement End Error

*A statement is not ended by one of the following basic symbols:

Either ; or 'END' or'ELSE'

6 Colon and Equals Sign Error

*The colon and equals sign representing the basic symbol := are omitted in the following cases:

*In a switch declaration, after the switch identifier.

*In a FOR statement, after the controlled variable.

7 Index Error

*In an array declaration, the upper bound is neither an integer variable nor an unsigned integer.

*In a statement, a subscript is neither an integer variable nor an unsigned integer.

*In a GOTO statement with a switch designator, the subscript is neither an integer variable nor an unsigned integer.

*A right bracket is omitted after the subscript.

*In a FOR statement, the controlled variable is not an integer variable.

*In a FOR statement, the initial value, the step value or the final value is neither an integer variable nor a signed (minus only) or unsigned integer.

8 Constant Outside Range

*The value of an unsigned integer is greater than 2040 when used explicitly as an ALGOL-8 program constant.

*The integer part of a real number is greater than 2040 when used explicitly as an ALGOL-8 program constant.

9 Simple Boolean Expression Error

*One of the simple arithmetic expressions of a relation is not an integer type.

*In a Boolean assignment statement, the long basic symbol beginning the Boolean expression is not one of the following:

'TRUE', 'FALSE', or 'NOT'

*The variable following a logical operator ('NOT', 'AND', 'OR', 'IMP', 'EQU') is not a Boolean variable.

*In a relation, a relational operator is omitted.

10 GOTO statement Error

*The identifier following the long basic symbol 'GOTO' is neither a label nor a switch identifier.

*Labels referenced by the program remain undefined at the end of the program.

11 BEGIN - END Error

*There are too many long basic symbols 'END'. Each 'BEGIN' must be paired with an 'END'.

12 Capacity Overflow Error

*One of the limitations described in Appendix B is exceeded.

*More than 18 conditional and FOR statements were in use at one time in the user program.

*The arithmetic expression being compiled is too complex for ALGOL-8 (stack overflow).

13 Assignment Statement Error

*The assignment operator (:=) is found in the middle of an arithmetic expression.

*The equals sign is not found after a colon in a multiple assignment statement.

14 Real Number Error

*The exponent of a real number is not properly written.

15 Read or Write Error

*The procedure statement with parameters is neither READ nor WRITE.

*The first parameter of READ or WRITE is neither 1 nor 2.

*A comma which separates parameters is omitted.

*One of the READ parameters is not an identifier.

*One of the WRITE parameters is neither an identifier nor a character string enclosed between double quotes.

*The identifier parameter of a READ or WRITE procedure statement is neither real nor integer.

*A left bracket is omitted in a subscripted variable.

*The right parenthesis which ends the READ or WRITE procedure statement is omitted.

16 Arithmetic Expression Error

*An operator is missing between two operands.

*Two operands are found consecutively.

17 Parenthesis Number Error

*The number of left parentheses is not equal to the number

of right parentheses in an arithmetic expression.

18 Standard Function Activation Error

*The parameter of the standard function is not a simple variable.

*The left or right parenthesis is missing.

*The parameter is not a real number except for the function REAL.

7.2 Execution Time Diagnostics

Errors detected at the time of program execution in ALGOL-8 cause the following error message on the Teletype printer:

OPS n

The error codes (n) are given below. When an error condition is encountered at execution time, the ALGOL-8 program will halt. If recovery is possible (as noted below), the user may press the CONT switch of the computer console and the program will resume execution.

Error Code	Condition	Recovery
1	User array size exceeds the available space.	No recovery possible.
2	Division by zero has been attempted in the user program	Division is performed.
3	Integer input outside acceptable range	Input ignored.
4	Attempt to convert a real number to an integer which is outside the acceptable integer range.	The real number is converted to zero
5	Attempt to take square root of a negative number.	Square root is taken of the modulus.
6	Attempt to take logarithm of a negative number.	Logarithm is taken of the modulus.
7	Switch index in users program is undefined.	Index is taken as 1.
8	Users program demands functions which have not been loaded.	No recovery possible.

Chapter 8

OPERATING INSTRUCTIONS

8.1 Compiler

The compiler consists of 2 paper tapes -

ALGOL Compiler	DEC-08-KA1A-PB
ALGOL Compiler Reinitialization	DEC-08-KA2A-PB

The compiler will only run in Field Ø, since it uses the program interrupt facility. The compiler must be reinitialized with the second tape, after each compilation, before another compilation is attempted.

8.1.1 Operation

The operation of the compiler is given in the following steps which are summarized in the accompanying flow chart (figure 8-1).

a. load compiler with binary loader

b. load address 400 and start

c. the program types OPT -

   The options are,

   R - Input/Output on high speed reader/punch

   T - Input/Output on low speed reader/punch

d. place the source program in the appropriate reader, turn on the appropriate punch and type R or T.

e. start Low Speed Teletype reader (if T has been typed)

f. the program halts at loc. 262 after compilation is complete

g. error messages are typed on the Teletype console and the program halts at loc. 3142.

h. re-initialize the compiler if a further compilation

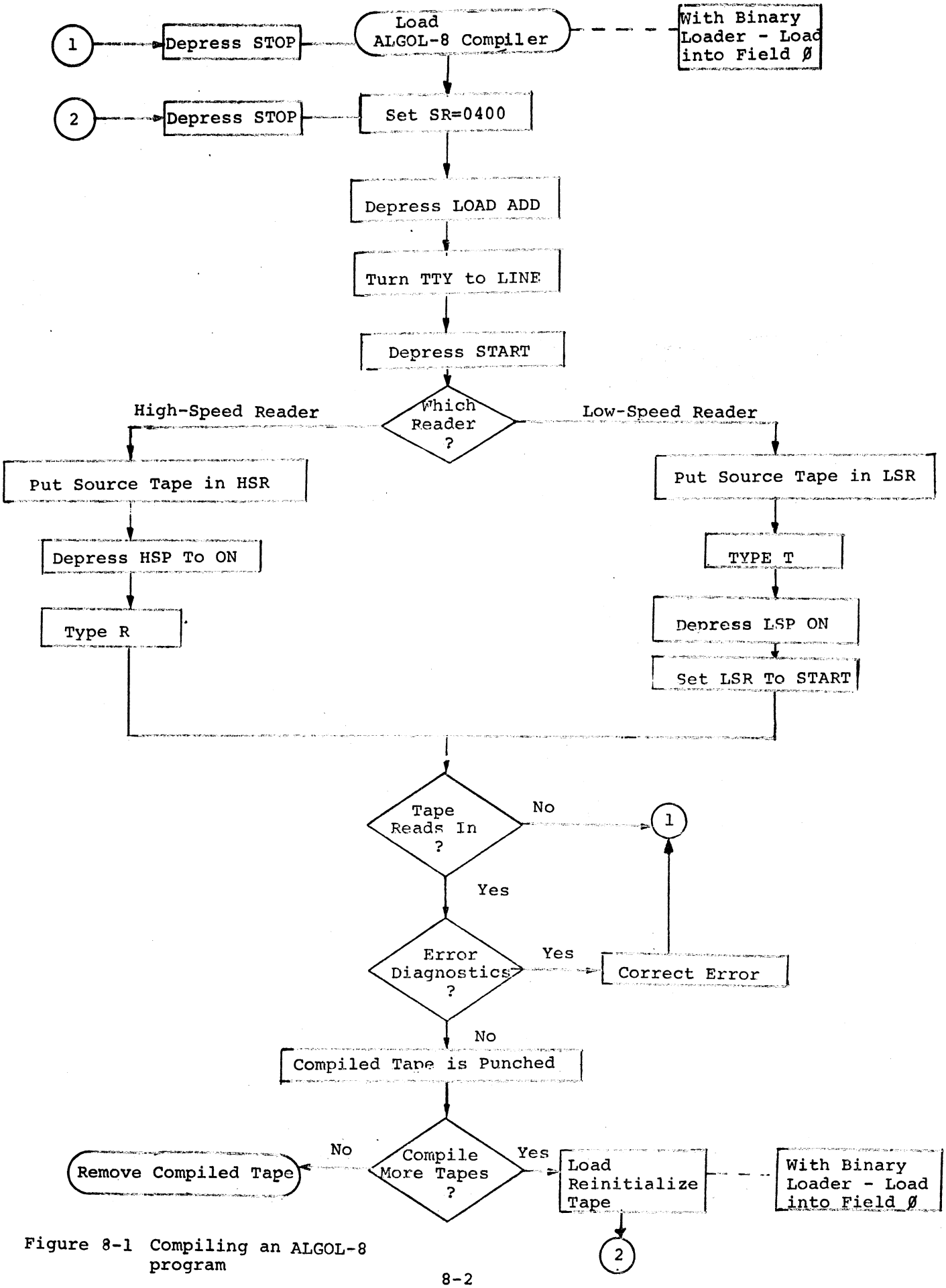
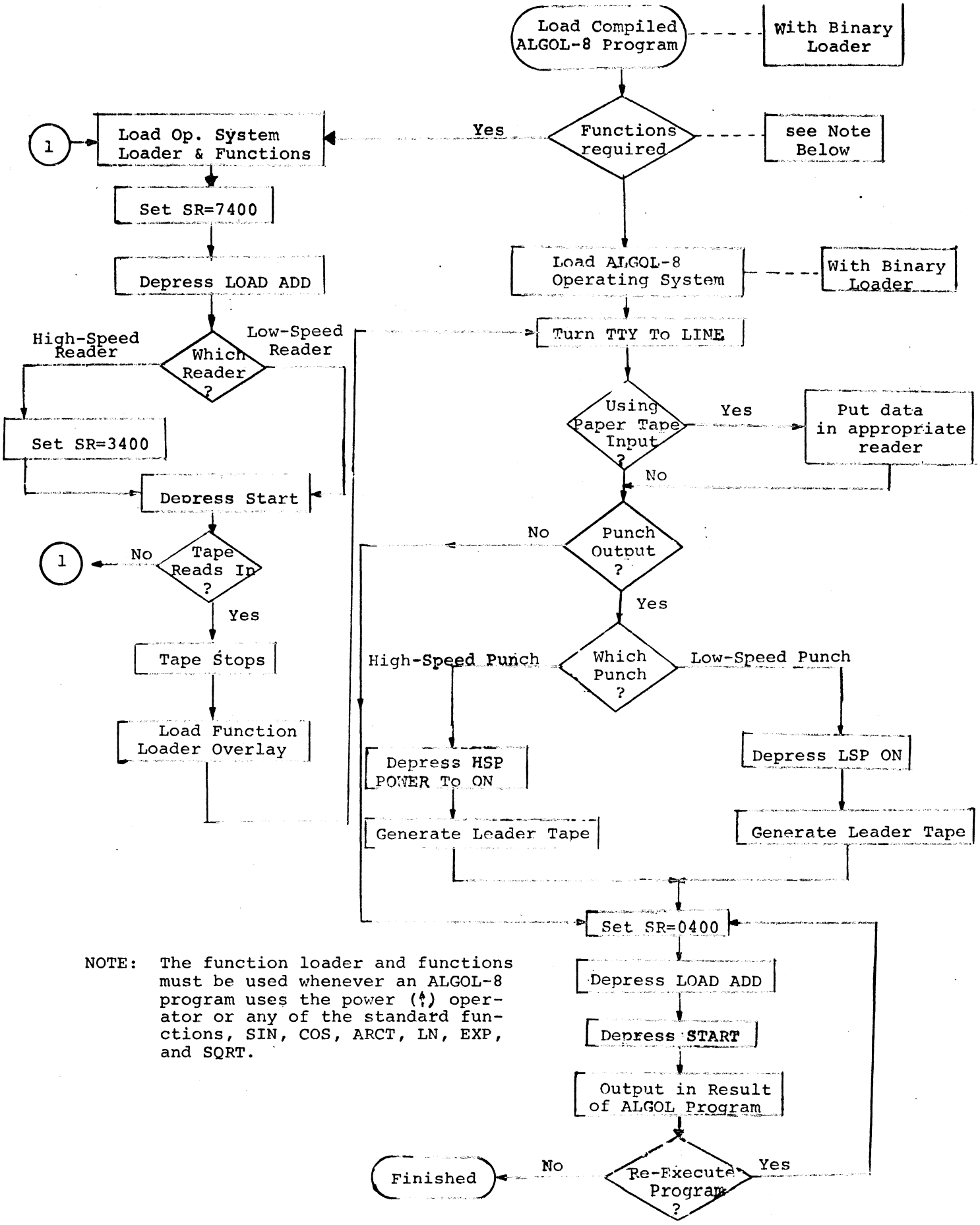


Figure 8-1 Compiling an ALGOL-8 program



NOTE: The function loader and functions must be used whenever an ALGOL-8 program uses the power (↑) operator or any of the standard functions, SIN, COS, ARCT, LN, EXP, and SQRT.

Figure 8-2 Running an ALGOL-8 Program.

is required, by loading the second tape with the Binary Loader.

8.1.2 Memory Usage

The compiler uses all memory locations from Ø-7577. The double input buffer is at loc. 4ØØ and the single output buffer is at 74ØØ.

8.2 Operating System

There are 3 tapes which make up the ALGOL-8 operating system

ALGOL OPERATING SYSTEM	DEC-08-KA3A-PB
ALGOL OPERATING SYSTEM, FUNCTION LOADER AND FUNCTIONS	DEC-08-KA4A-PB
ALGOL FUNCTION LOADER OVERLAY	DEC-08-KA5A-PB

The system is based on the floating point package (DEC-8-5-S). It will run in any memory field, but will not use more than one field.

The standard functions (SIN, COS, ARCT, LN, EXP, and SQRT) are loaded only if they are required by the compiled program. They must be loaded if the program calls them explicitly or if the exponentiation (↑) operator is used.

There are two alternative operating procedures:

8.2.1 Operation - No Functions Required

a. Load compiler output with Binary Loader.

b. Load Operating System (KA3A-PB) with Binary Loader.

c. Input data must be in the appropriate reader if required and, if punched output is required, the appropriate punch must be on.

d. Load address 400 and start.

e. The program may be restarted at location 400.

8.2.2 Operation - Functions Required

a. Load compiler output with Binary Loader.

b. Load Operating System, Function Loader and Functions (KA4A-PB) with Binary Loader.

c. The Binary Loader loads the operating system and the Function Loader. The tape will stop before the functions are loaded.

d. To load the functions, load address 7400, set bit Ø for appropriate reader (as in the Binary Loader) and start.

e. The necessary functions will be loaded and the reader will halt at location 7424. (A checksum error will cause the loader to halt at location 7511 with the checksum difference in the AC.) The reader will load only the needed functions and, in general, the reader will halt with the function tape only partly read.

f. Load the Function Loader Overlay (KA5A-PB) with the Binary Loader.

g. Go to step c of Paragraph 8.2.1.

8.2.3 Memory Usage

The Operating System uses locations 0-377 and 4600-7577.

Functions are loaded by page, downwards (towards 0) from location 4600.

The compiled program is loaded from 400 upwards. The pages between the compiled program and the functions (or location 4600, if no functions are required) are used for array storage.

8.3 Paper tape Input Format

Input to the compiler and data input to the operating system may be in parity or non-parity ASCII format paper tape. Blank tape, rubout and line-feed characters are ignored. Spaces are ignored except in text strings.

8.3.1 Compiler Input

A source program must be terminated by a $, which must be the first character of a line, followed by a carriage return.

8.3.2 Operating System Input

Input of a data item to the operating system i.e. by a READ procedure statement is terminated by any character which is not a number, sign, decimal point or $ (indicating the exponent).

APPENDIX A

ALGOL-8 BASIC SYMBOLS

Long Basic Symbols	Corresponding ALGOL-8 Representation
begin	'BEGIN'
end	'END'
real	'REAL'
integer	'INTEGER'
boolean	'BOOLEAN'
array	'ARRAY'
switch	'SWITCH'
if	'IF'
then	'THEN'
else	'ELSE'
for	'FOR'
step	'STEP'
until	'UNTIL'
do	'DO'
goto	'GOTO'
comment	'COMMENT'
true	'TRUE'
false	'FALSE'

Short Basic Symbols	Corresponding ALGOL-8 Representation
+	+
-	-
×	*
/	/
↑	↑
≐	=
≠	#
<	<
>	>
≤	< =
≥	> =
.	.
,	,
;	;
:	:
:=	:=
Power of 10	$
(	(

ALGOL-8 BASIC SYMBOLS

Long Basic Symbols	Corresponding ALGOL-8 Representation
¬	'NOT'
∧	'AND'
∨	'OR'
⊃	'IMP'
≡	'EQU'

Short Basic Symbols	Corresponding ALGOL-8 Representation
)	)
[	[
]	]
‘	"
’	"

*NOTE Shift K= [

Shift M=]

APPENDIX B

RESTRICTIONS

Limitations of the Language

1. There are no user procedures.
2. There are no Boolean arrays.
3. An integer or real array must be one dimensional.

   The lower bound must be either the digit Ø or the digit 1.

   The upper bound must be a simple integer variable or an unsigned integer.
4. The subscript of an array variable or of a switch designator must be a simple integer variable or an unsigned integer.
5. Variable identifiers are differentiated by the first four alphanumeric characters only.
6. A switch list must be a label list, the labels of which must be declared in the same block in which the switch declaration appears, or in an outer block.
7. The while long basic symbol is not recognized.

Limitations of Program Size and Form

1. 50 identifiers simultaneously valid. Identifiers are defined as simple variables, subscripted variables, switch designators and declared labels.
2. 10 arrays (real or integer).
3. 7 nested blocks.
4. 18 undeclared labels. Undeclared labels are those labels whose declaration occurs in the same block after their use, or in an outer block. Hence, labels of a switch list are undeclared labels.

APPENDIX C

TABLE OF ERRORS

0	Long basic symbol error
1	Identifier Error
2	Declaration Error
3	Multiple Declaration error
4	Statement error
5	Statement end error
6	Colon and equals sign error
7	Index error
8	Constant outside range
9	Simple Boolean expression error
10	GOTO statement error
11	Begin-End error
12	Capacity overflow error
13	Assignment statement error
14	Real number error
15	Read or write error
16	Arithmetic expression error
17	Parenthesis number error
18	Standard function error

APPENDIX D

SUMMARY OF COMMANDS

'BEGIN'	Opening statement parenthesis - used to start a program, compound statement or block.
'END'	Closing statement parenthesis - used to terminate a program, compound statement or block.
'REAL' A,B,C;	Real number variable declaration - used to identify the real number variables used within a program or block; in this case A, B and C are real variables.
'INTEGER' I, J, K;	Integer variable declaration - used to identify the integer variables used within a program or block; in this case I, J and K are integer variables.
'BOOLEAN' B1, B2;	Boolean variable declaration - used to identify the Boolean variables used within a program or block. In this case B1 and B2 are Boolean variables.
'ARRAY' A [1:10] , B [Ø:N] ;	Array declaration - used to specify the arrays used within a program or block. This

	symbol should be preceded by 'INTEGER' or 'REAL' to identify the type of array. In ALGOL-8, the lower limit must be Ø or 1, and the upper limit must be an unsigned integer (e.g. 10, above) or a simple integer variable (e.g. N, above). Only one-dimensional arrays are allowed in ALGOL-8.
'SWITCH' S:=L1, S2,STOP;	Switch declaration - used with 'GOTO' S [I] to branch an ALGOL-8 program, based on the value of an integer index, I. The index I may be an expression but its values must range from 1 upward. In the example given, the possible values for I must be 1, 2 and 3.
'GOTO' L;	GOTO statement - specifies that the statement to be executed next is the one identified by the label L. (L is an identifier which is separated from the statement which it precedes by a colon.)
'COMMENT' TEXT;	Comment statement - provides a means of including comments in a program to identify

parts and to clarify procedures. All characters typed after the 'COMMENT' and before the semicolon are included in the program listing. They are not typed during the running of a program.

'IF'R 'THEN' S; IF statement - R is a relation, such as X≥Y+5; S is a statement, such as 'GOTO' BIG;. If the relation R is true, the statement S will be executed. If the relation R is false, S is skipped and the next statement in sequence is executed. S may be a compound statement, bracketed by 'BEGIN' and 'END'.

'IF' R 'THEN' S1 'ELSE' S2; Conditional statement - if relation R is true, statement S1 is executed and S2 is skipped; if R is false statement S1 is skipped and S2 is executed. S1 must not be a conditional statement; however, S2 may be conditional, thus establishing nested conditional statements.

'FOR' V1:= A1, A2, A3 'DO' S1; FOR statement - statement S1 is executed repeatedly with the controlled variable V1 taking on each of the values of A1, A2, and A3. For example, S1 could be a block to compute the sine of an

angle V1, where A1, A2, and A3 are the desired values of V1.

'FOR'V1:=X1 'STEP'X2 'UNTIL'X3 'DO'S1; Step-until statement - statement S1 is executed repeatedly with the control variable V1 taking on values of X1 initially, and increasing to X3 by adding on the increment X2. For example, if S1 computes the sine of an angle, X1 (initial) could be Ø degrees, X2 (increment) could be 10 degrees, X3 (final) could be 90 degrees.

APPENDIX E

NOTES ON ALGOL

COMPILER

The ALGOL-8 compiler is analyzed with emphasis on the implementation, performance, and error diagnosis.

(1) The ALGOL-8 compiler will halt with every syntex error and does not continue to scan the rest of the source. This is typical of many ALGOL compilers.

(2) After each compilation load the first part of RE-INIT to type out the symbol table and load the second part of "RE-INIT to reinitialize the compiler. This is only necessary if the identifier printout is desired.

(3) Error diagnosis performed by the compiler is fairly good with the exception that the message typed out may sometimes appear vague and as a result becomes unclear, each error code covers many different messages. This results from the complexity of the programming language and the ability of the program to be contained in 4K.

The following observations were made during compilation time.

(1) Boolean expressions must be written as specified in section 3.2.1 of the manual

(2) I/O of Boolean identifiers and mixed mode constructions with Boolean mode identifiers are not allowed.

(3) All "END" statements must be terminated by (;).

(4) The following refers to dummy statements.

```
'BEGIN'
    .
    .
    .
'BEGIN'
'IF'    I = J 'THEN' 'GOTO' EXIT;
    .
    .
    .
    .
'END'
EXIT:
'END'
$
```

The above situation will always cause BAD 'GOTO' error message, because 'EXIT' is outside the block where 'GOTO' statement tries to link.

The correct form is as follows:

```
'BEGIN'
'IF' I=J 'THEN 'GOTO' EXIT;
    .
    .
    .
    .
EXIT:
'END'
```

(5) With regards to "GOTO" statement by referencing ALGOL-8 Manual page 5-2, Section 5.2 GOTO examples have shown different result.
GOTO example 1 O.K.
GOTO example 2 error 10 during compilation
refer to the attached sheet.

OPERATING SYSTEM

(1) DOT (.) is not allowed for integer input. It does not give any error message but simply treats it as a null data.

(2) WRITE (1, "INPUT␣␣␣␣␣␣␣ COSINE"); SKIP; will occasionally print out as: INPUTCOSINE. The seven spaces between the two words are not printed out.

(3) In the statement 'FOR' V: = 1 ' . . . .'UNTIL' N 'DO'
if N has value a of 0 or a negative number, all the statements within the range of 'DO' will be ignored, but no error message will appear.

(4) "VARIABLE ARRAY" program is attached for reference, because the "WRITE" statement under the control of "FOR" yields only 1 line result so that most of them will not be readable. The result is also attached.

ALGOL Functions including both Basic and Standard.

(1) "ENTIER" function test results are attached at the end of this section.

(2) Boolean operator 'AND' does not function properly at all times for example:
T = true
F = false
T 'AND' T→T Correct
T 'AND' F→T Wrong, should be F
F 'AND' F→T Wrong, should be F
Refer to results of Boole 1A & 2A.

(3) SINE function:
let x = argument
$.001 \leq x \leq .05$

Then error E for all the arguments in the range above is $.0000001 < E < .00001$
3/50 yields 5-digit accuracy.
47/50 yields 6-digit accuracy.

(4) COSINE function:

The range for argument x where $.001 \leq x < .025$ almost half have 6-digit accuracy and the other half have 5-digit accuracy with $-.000001$ as error.

For other arguments of other ranges, the perctage which guarantees 6-digit accuracy improves and the error is seldom greater than .000001.

(5) ARCTAN function:

The range for argument x where $.001 \leq x \leq .025$ few had an error of $-.000001$ and guarantee of 6-digit accuracy is given.

(6) Exponential function:
The range for the argument x where $0 \leq x \leq 14$ produced excellent results with the following exceptions:
e↑9. yields 8103.07 instead of 8103.08 The error is -.01.

e↑11. yields 59873.9 instead of 59874.1 with error 1.2.

e↑13 yields 442412. instead of 442413. with error -1.0.

(7) ABS function operates accurately.

(8) Natural log function:

The range of the argument x where $1. \leq x \leq 1.030$. One sixth yield 6-digit accuracy, whereas most of them yield only 5-digit accuracy and one tenth yield approximately 4-digit accuracy.

(9) Square Root function:

The range of argument where $1. \leq x \leq 40.$ All results have a 6-digit accuracy.

USERS' MANUAL CHANGES

(1) Page 1-2. 'INTEGER' SUM: (colon should be changed to semicolon after sum).

(2) Page 1-8. 'FOR' E 'STEP' I 'UNTIL' F 'DO's. It would be much better and less confusing if A semicolon (;) is inserted after S.

(3) Page 3-3. Division (/) is considered as Real Operator. (Integer division is allowed.)

(4) Page 4-1. Chapter 4. All declaration statements should be listed before their explanation, because all declaration statements must appear at the beginning of a block.

(5) Page 5-2. Section 5.3

Treat 'FOR' V: = A . . . . as 2.

(6) Page 5-4. The total number of conditional and FOR statements that can be accommodated at one time is eighteen.

This particular combination is not tested.

However, its equivalent:
'IF' . . . 'THEN' . . . . 'ELSE' was tested and it can accommodate at one time 9.

(7) Page 5-5. Section 5.5

B: = A*A*A; is a dummy statement.

(8) Page 5-6. In the example:

K: I + J; should be K: = I + J;

(9) Appendix B: At no time can a single ALGOL program contain more than 1Ø (TEN) different dimensioned variables.

Ex: A (1:N), B(1:N) . . . .

VARIABLE ARRAY

```
        'BEGIN'
        'COMMENT' ARRARY STATEMENT TEST;
        'INTEGER' N,V;
        'REAL' 'ARRAY' AX[1:N],BX[1:N];
        WRITE (1, "TEST FOR ARRAY STATEMENT") ;SKIP;
        WRITE (1, "TYPE A NUMBER IN, MAX=1Ø") ;SKIP;
        READ (1, N) ;SKIP;
        'FOR' V:=1 'STEP' 1 'UNTIL' N 'DO'
        'BEGIN'
        READ (2,AXCV1) ;SKIP;
        BX[V]:=1/AX[V];SKIP;
        'END';
        'FOR' V:=1 'STEP' 1 'UNTIL' N 'DO'
        WRITE (1,AX[V],BX[V]) ;SKIP;
        'FOR' V:=1 'STEP' 1 'UNTIL' N 'DO'
        WRITE (2,AX[V],BX[V]) ;SKIP;
        'END'
$
```

GOTO, EXAMPLE 1

```
          'BEGIN'
          'INTEGER' I, J;
          WRITE (1, "GOTO TEST");SKIP;
          READ (1, I, J);SKIP;
          'BEGIN'
          'IF' I=J 'THEN' 'GOTO' OK;
          WRITE (1, "NOT TOO GOOD") ;SKIP;
          'GOTO' EXIT;
OK:       WRITE (1, "O.K.") ;SKIP;
          'END';
EXIT:     WRITE (1, "EXIT") ;SKIP;
          'END'
$
```

GOTO, EXAMPLE 2

```
*L
          'BEGIN'
          'INTEGER' I, J;
          WRITE (1, "GOTO TEST") ;SKIP;
          READ (1, I, J) ;SKIP;
          'BEGIN'
          'IF' I = J 'THEN' 'GOTO' OK;
          WRITE (1, "NOT TOO GOOD") ;SKIP;
          'GOTO' EXIT;
OK:       WRITE (1, "O.K.") ;SKIP;
          'END'
EXIT:     WRITE (1, "EXIT") ;SKIP;
          'END'
$

*
```

```
        'BEGIN'
        'REAL' A,B;
        'INTEGER' I,J;
        WRITE (1, "DUMMY TEST") ;SKIP;SKIP;
        READ (1,A,I,J  ;SKIP;
        WRITE (1,"A,B,I,J ARE NOT DECLARED IN THE NEXT BLOCK") ;SKIP;
        WRITE (1,"NEXT IS A NEW BLOCK") ;SKIP;
        'BEGIN'
        'IF' I = J 'THEN'  'GOTO'  EXIT;
        WRITE (1,"WRITE 5 SKIPS") ;SKIP;SKIP;SKIP;
        SKIP;SKIP;
        B:=A*A*A;
        EXIT:
        'END'
        'END'
$

*
```

```
        'BEGIN'
        'REAL' A,B;
        'INTEGER' I,J;
        WRITE (1,"DUMMY TEST") ;SKIP;SKIP;
        READ (1,A,I);
        WRITE (1,"A,B,I,J ARE NOT DECLARED IN THE NEXT BLOCK") ;SKIP;
        WRITE (1,"NEXT IS A NEW BLOCK") ;SKIP;
        'BEGIN'
        'IF' I = J 'THEN' 'GOTO' EXIT;
        WRITE (1,"WRITE 5 SKIPS") ;SKIP;SKIP;SKIP;
        SKIP;SKIP;
        B: =A*A*A;
        'END'
EXIT:
        'END'
$

*
```