

cc-69-11

ALGOL: OS-3 User's Manual

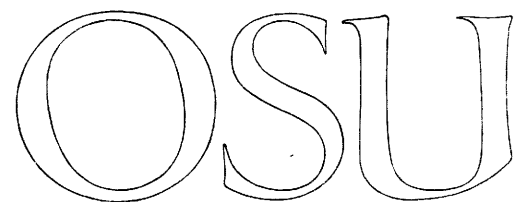
October, 1969

by

Jo Ann Baughman

Mary Lynn Berryman

Yvonne Yapp

The logo for Oregon State University, consisting of the letters 'OSU' in a large, outlined, serif font.

COMPUTER CENTER

Oregon State University
Corvallis, Oregon 97331

ALGOL: OS-3 User's Manual

October, 1969

cc-69-11

Jo Ann Baughman
Mary Lynn Berryman
Yvonne Yapp

Oregon State University
Computer Center
Corvallis, Oregon 97330

Copyright 1969, Oregon State University
Computer Center Publications
Corvallis, Oregon 97331

Printed by the Oregon State University
Department of Printing

PREFACE

ALGOL, algorithmic language, was developed during the years of 1958 and 1959 on an international basis. ALGOL has been defined as a language which is useful for the description of processes. These processes can be numerical or logical.

The purpose of this manual is intended to provide, for the user with a knowledge of ALGOL, some examples of entering and executing ALGOL programs on the CDC 3300 or 3500 under OS-3. No attempt has been made to teach ALGOL but only to illustrate the use of those commands which will assist the ALGOL programmer in the use of ALGOL under OS-3. This is not an all-inclusive volume.

We wish to acknowledge the support of the National Science Foundation grant GJ 51, NSF AUFENKAMP 370 Regional Center, which aided in the development of this manual. We also thank Gilbert Bachelor and Robert Brenne for their contributions to this publication.

ALGOL: OS-3 USER'S MANUAL

TABLE OF CONTENTS

	<u>Page</u>
Introduction	1
 PART ONE	
I. An Introduction to ALGOL from the Teletype.....	3
1. Using EDIT.....	3
2. An Example of Editing and Running an ALGOL Program.....	6
3. Saving and Running a Binary Version of a Program.....	10
4. Making a Paper Tape of a Program on a Saved File.....	12
II. An Introduction to ALGOL from Batch.....	13
1. To Run an ALGOL Source Deck.....	14
2. To Run an ALGOL Program that is on File.....	15
3. To Punch a Card Deck from a Program on File.....	16
4. To Copy a Source Deck onto a Saved File.....	17
 PART TWO	
I. Elements of Simple Program.....	19
1. Illustration on Use of 'BEGIN', 'COMMENT', 'END', and 'EOP'.....	19
2. Sample Program.....	20
II. Declarations.....	21
1. Type Declarations.....	22
2. Array Declarations.....	24
3. Switch Declarations and Calling.....	25
4. Procedure Declarations and Calling.....	28
5. Sample Program (on declaration).....	31
III. Operators and Values.....	35
1. Relational Operator.....	35
2. Logical Operator.....	35
3. Arithmetic Operator.....	35
4. Logical Value.....	36
5. Sample Program on Operators.....	36

IV.	Expressions.....	38
1.	Function Designators.....	38
2.	Arithmetic Expressions (Including IF clause).....	38
3.	Boolean Expressions.....	38
4.	Designational Expressions.....	39
5.	Standard Functions.....	39
V.	Statements.....	40
1.	Compound Statements.....	40
2.	Block.....	40
3.	Assignment Statement.....	43
4.	GO TO Statement.....	43
5.	Dummy Statement.....	43
6.	Conditional and Unconditional Statements.....	44
7.	Procedure Statements.....	48
8.	FOR Statement.....	48
9.	Input/Output and Format String.....	52
10.	Sample Run on All Statements.....	62

PART THREE

I.	Newton's Method.....	65
1.	Newton's Method for Cube Roots.....	65
2.	Newton's Method with Looping.....	71
II.	Two-Dimensional Array.....	74
III.	Economics Equilibrium Problem.....	79
IV.	Simpson's Rule for Integration.....	85

REFERENCES	89
INDEX.....	i

INTRODUCTION

The digital computer has astonishing capabilities as a tool for the experimental scientist. It is immediately evident that the use of such a tool requires thorough preparation. The computer can do nothing that the user in principle could not also do; it can only do it faster.

It is clear that we must write down the set of instructions that we want to give to the machine. However, we cannot use everyday language and expect the machine to receive instructions in the form of a letter or dictation. Instead we must adapt the language used to the capabilities of the machine.

Although the computing machine can be used in more general problems, in what follows we are chiefly concerned with computation--more precisely, arithmetic computation. Arithmetic formulas which contain numbers, names denoting yet unknown quantities, and functions (such as SIN, COS, and LN) have long been used to describe computational rules. Such formulas form a core embedded within a sequence of organizational statements which describe the flow of the computational process. Thus, for example, the execution of parts of the computation can be made to depend on certain conditions, or one can prescribe the number of times a part is to be repeated. Indeed, the power of the automatic computer comes from its ability to make precise decisions at definite places during the course of the computation in accordance with preassigned criteria.

Finally, the machine must receive specifications as to type and dimension of the initial data entering into the computation (input data) and the numerical values given as the result (output data). A complete set of instructions and rules written in such a manner that it uniquely defines the course of a computation from beginning to end, we will call a program.

The preparation of the program entails more than due consideration to the arithmetic and organizational capabilities of the

machine. The simple-minded intelligence of the computer requires that the language used be formed according to stringent rules. The present manual explains the standardized formal language ALGOL (ALGOriithmic Language) which arose out of an international effort. ALGOL programs are largely independent of the properties of individual machines and are conveniently readable by a wide circle of interested people. To an ever increasing extent algorithms and programs are being written and published in ALGOL.

In Part One the reader will find introductory examples illustrating the use of ALGOL from a remote Teletype and from batch (cards).

In Part Two the words from the ALGOL language are defined and examples are given with each. This section may be of benefit to students enrolled in an ALGOL class. Throughout the manual references will be made to the manuals listed at the end of the manual.

In Part Three complete examples are given with written explanations for the procedures used. The comments include explanations about the usage of ALGOL and also about the structuring of the programs. Each example has been run from the Teletype and appears exactly as it did on the printout from the Teletype unit.

Users interested in additional information should refer to the following Computer Center publications:

ccm-70-7, OS-3 Editor Manual, Dayton, January, 1970

ccm-70-8, OS-3 Reference Manual, Skinner, January, 1970

CDC, ALGOL Generic Reference Manual

PART ONE

I. AN INTRODUCTION TO ALGOL FROM THE TELETYPE

Since the user may make use of EDIT to create a program, an introduction to EDIT will be given, then reference will be made to special characters and keys on the Teletype. Reference will then be made to the ALGOL language that the compiler accepts. A detailed example of entering, editing, and running an ALGOL problem is given along with an explanation of all commands used.

1. USING EDIT

The OS-3 system library contains various routines such as the Fortran compiler, the Algol compiler, OSCAR, RADAR, and EDIT.

The EDIT program allows the user to generate, alter, or list files. In the EDIT program, a right bracket (]) sign indicates that you are in the EDIT command mode.

All editing operations are performed in a core memory work area. Information to be edited must be transferred into this area to be modified. Modified information must be copied out to a file before it can be used. Binary machine language object programs cannot be handled by EDIT.

All editing operations are performed in a core memory work area. Information to be edited must be transferred into this area to be modified. Modified information must be copied out to a file before it can be used. Binary machine language object programs cannot be handled by EDIT.

Turn Teletype to ON LINE.

Type a Control A (hold CONTROL key down and type an A) CS,A.
Type in your Job Number and User Code and push key marked
RETURN CR .

The Teletype should, if you typed a valid JOB and USER NUMBER,
block out your JOB and USER CODE and type the date and pound
symbol (#).

Type the word EDIT and push RETURN CR key. You are now in
the EDIT command mode.

#####

JANUARY 19, 1970 10:17 AM TERMINAL 042

#EDIT

]INPUT

00001:AL1

00002: THIS WILL BE AN ADDING PROGRAM.

00003:

00004: 'BEGIN'

00005: 'REAL' A,B,SUM;

00006: INPUT(60, '(' ')', A,B);

00007: SUM=A+B;

00008: OUTPUT(61, '(' * ')');

00009: OUTPUT(61, '(' ')', A,B,SUM);

00010: 'END'

00011: 'EOP'

00012:

The Teletype will type a right bracket (]), which indicates it
is waiting for an EDIT command. You type the command INPUT and
push the RETURN key CR . The Teletype will print 00001:.

Within EDIT are many commands, one of which is INPUT. This
command prepares a temporary storage for information. A
sequence number followed by a colon is provided for each
line. For more information on the EDIT mode see OSU Computer
Center manual ccm-70-7.

The Algol Compiler will use the first 8 characters preceding the first 'BEGIN' as identification. This example will be identified by the ALGOL Compiler as AL1.

You are now ready to enter your ALGOL program. If you make an error while in EDIT, a backward slash, `\`, (an upper case L) will cause the last character or blank to be ignored. An @ sign will cause the previous characters and blanks (on the same line) to be ignored.

Example: ABD C is equivalent to ABC. Example: ADB@ABC is equivalent to ABC.

Notice that in line 7 an error occurs. This will be picked up later by the compiler, (it should read `SUM:=A+B`). Let us assume at this point that you do not know the error exists.

You are now ready to save the program. The name of the file in which the program is saved may be different from the name of the program. In this example, the name of the program is AL1 and the name of the saved file, on which there exists a copy of the program, is ALONE; you can now access this program by asking for the file ALONE.

```
1OUT,ALONE
```

```
1
```

2. AN EXAMPLE OF EDITING AND RUNNING AN ALGOL PROGRAM

To compile your program, type ALGOL,I=ALONE,X.

```
#ALGOL,I=ALONE,X
```

This command is typed in the OS-3 Control Mode (#). The command will check your program for errors and send the object program to LUN 56, (Logical Unit Number).

```
OS3 ALGOL  V0.0          AL1          01/19/70    1020
(01)  LINE 0004          PROGRAM BEGINS
(01)  LINE 0010          PROGRAM ENDS
(01)  LINE 0010          SOURCE DECK ENDS
(03)  LINE 0007          DELIMITER
```

Four (4) lines will be typed out for all programs. The first will be general information: Version, Program Name, Date, Time and Page. The second will be the line number where the program begins. The third will be where the program ends. The fourth will be where the source deck ends. Anything appearing after these lines will be error messages.

If the error in line 7 had not been there you could have loaded 56 at this time. However, we will go back to EDIT and correct the program. The error message tells us a delimiter is wrong in line 7. The Search and Replace and List (SARL) command will be used.

```

#EDIT

]FIN,ALONE

]SARL,7,,/=/,/:=/
00007:    SUM:=A+B;

]OUT,ALONE

]

```

The program has been corrected and we are ready to again call the ALGOL Compiler; however, first rewind or release LUN 56.

This time the compiler will be called with another parameter, L. This will cause the program to be listed.

```

#RELEASE,56
#ALGOL,I=ALONE,X,L

```

```

OS3 ALGOL  V0.0      AL1      01/19/70      1022
AL1
THIS WILL BE AN ADDING PROGRAM.

```

```

      'BEGIN'
      'REAL' A,B,SUM;
      INPUT(60, '(' ')', A,B);
      SUM:=A+B;
      OUTPUT(61, '(' * ')' );
      OUTPUT(61, '(' ')', A,B,SUM);
10**  'END'
      'EOP'
(01)  LINE 0004    PROGRAM BEGINS
(01)  LINE 0010    PROGRAM ENDS
(01)  LINE 0010    SOURCE DECK ENDS

```

You are now ready to load the object program. Type LOAD,56, then type RUN followed by a carriage return (CR) and a Line Feed (LF).

```

#LOAD,56
RUN
RUN

```

The computer will now type standard channel information. 60 is the standard unit for TTY input; 61 is the standard unit for TTY output. If the standard channels are all your program requires, type starting in column one CHANNEL,END followed by a (CR) and a (LF).

```
CHANNEL,60=LU60,P80
CHANNEL,61=LU61,P136,PP60
CHANNEL,END (CR) (LF)
CHANNEL,END
```

The computer will now wait for you to enter the values for variables A and B (a result of the statement in line 6 of the program). End each record with a (CR) and (LF).

```
12.0 (CR) (LF)
13.0 (CR) (LF)
```

The computer will now type out the variable values indicated in line 9 of the program.

```
+1.200000000'+001 +1.300000000'+001 +2.500000000'+001
```

```
END OF ALGOL RUN
```

```
#LOGOFF
```

```
TIME 7.226 SECONDS MFBLKS 1 COST $0.83
```

Since this program was saved in the previous example with the command]OUT,ALONE (see page 5) the user can run the program at any time with the command

```
#ALGOL,I=ALONE,X.
```

#####

JANUARY 19, 1970 10:26 AM TERMINAL 042

#ALGOL,I=ALONE,X

OS3 ALGOL	VO.0	ALI	01/19/70	1026
(01)	LINE 0004	PROGRAM BEGINS		
(01)	LINE 0010	PROGRAM ENDS		
(01)	LINE 0010	SOURCE DECK ENDS		

#LOAD,56
RUN
RUN

CHANNEL,60=LU60,P80
CHANNEL,61=LU61,P136,PP60
CHANNEL,END
CHANNEL,END

12.0
13.0

+1.200000000'+001 +1.300000000'+001 +2.500000000'+001

END OF ALGOL RUN

#LOGOFF
TIME 4.900 SECONDS MFBLKS 1 COST \$0.62

3. SAVING AND RUNNING A BINARY VERSION OF A PROGRAM

The program is compiled as before but before loading 56, the compiler (object) program on 56 is saved with the command

```
SAVE,56=BINALONE.
```

```
#####
```

```
JANUARY 19, 1970 10:29 AM TERMINAL 042
```

```
#ALGOL,I=ALONE,X=56
```

```
OS3 ALGOL  V0.0           AL1           01/19/70      1029
(01)  LINE 0004          PROGRAM BEGINS
(01)  LINE 0010          PROGRAM ENDS
(01)  LINE 0010          SOURCE DECK ENDS
```

```
#SAVE,56=BINALONE
```

```
#LOGOFF
```

```
TIME 1.848 SECONDS MFBLKS 1 COST $0.20
```

Since the command #ALGOL,I=NAME,X automatically equips the ALGOL library and since we will not use this command with a binary program we must equip the library, *ALGLIB to LUN 63. The rest of the run is the same as before.

JANUARY 19, 1970 10:32 AM TERMINAL 042

#EQUIP,63=*ALGLIB
#LOAD,BINALONE
RUN
RUN

CHANNEL,60=LU60,P80
CHANNEL,61=LU61,P136,PP60
CHANNEL,END
CHANNEL,END
4325.7896 .97654325

+4.325789600'+003 +9.765432500'-001 +4.326766143'+003

END OF ALGOL RUN

#LOGOFF
TIME 2.140 SECONDS MFBLKS 0 COST \$0.29

4. MAKING A PAPER TAPE OF A PROGRAM ON A SAVED FILE

The command, TTP, will generate several inches automatically for the beginning and the end of the tape. After typing TTP make sure the dial is turned to KT (keyboard and tape).* This command will automatically place a control shift R [TAPE] and a control shift T on the tape.

```
#####
JANUARY 19, 1970 10:37 AM TERMINAL 042

#EDIT

JFIN,ALONE

JTTP
AL1
THIS WILL BE AN ADDING PROGRAM.

'BEGIN'
'REAL' A,B,SUM;
INPUT(60, '(' ')', A,B);
SUM:=A+B;
OUTPUT(61, '(' * ')' );
OUTPUT(61, '(' ')', A,B,SUM);
'END'
'EOP'

J
#LOGOFF
TIME 0.421 SECONDS MFBLKS 0 COST $0.06
```

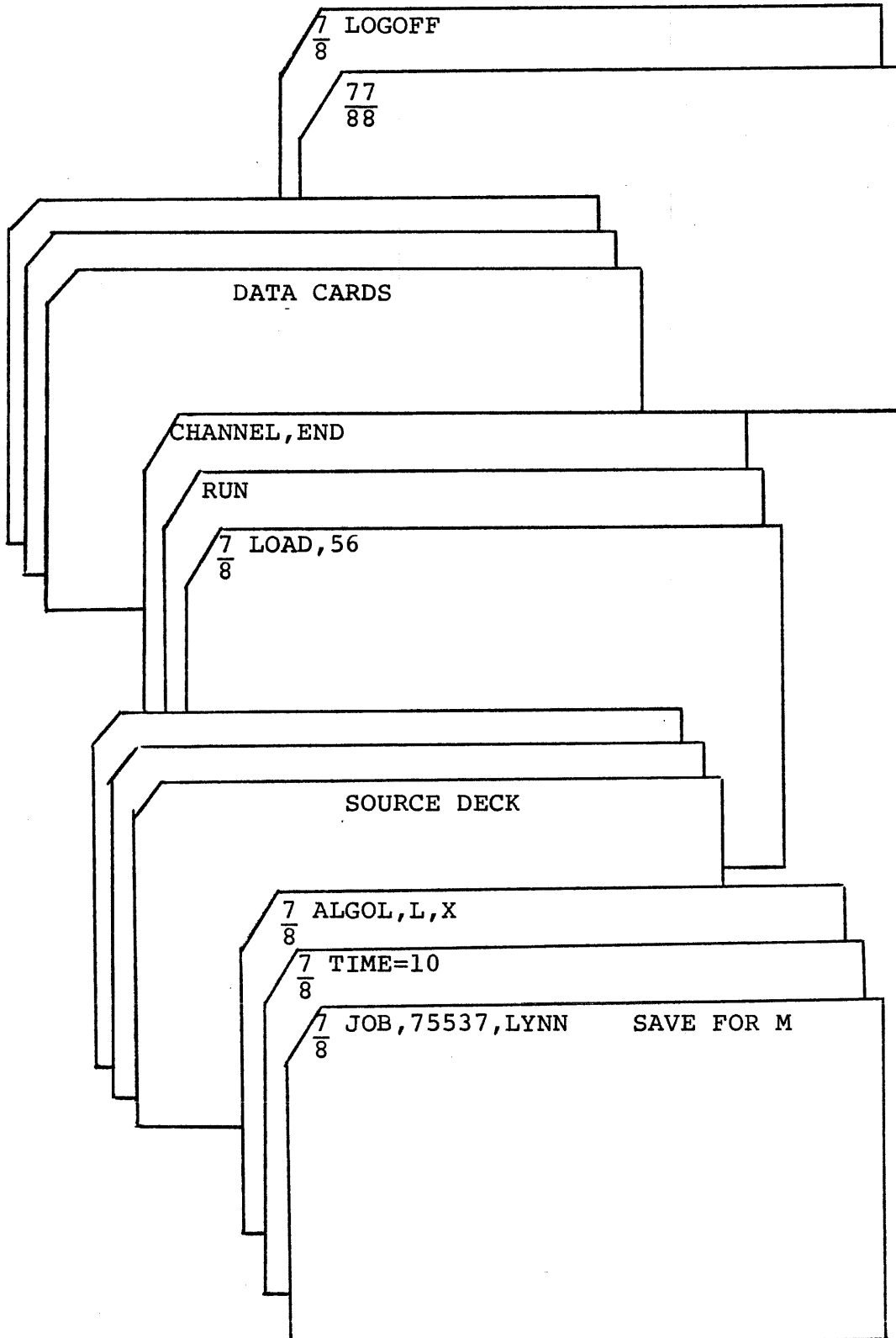
* On a TTY 33 just punch and keyboard have to be turned on.

II. AN INTRODUCTION TO ALGOL FROM BATCH

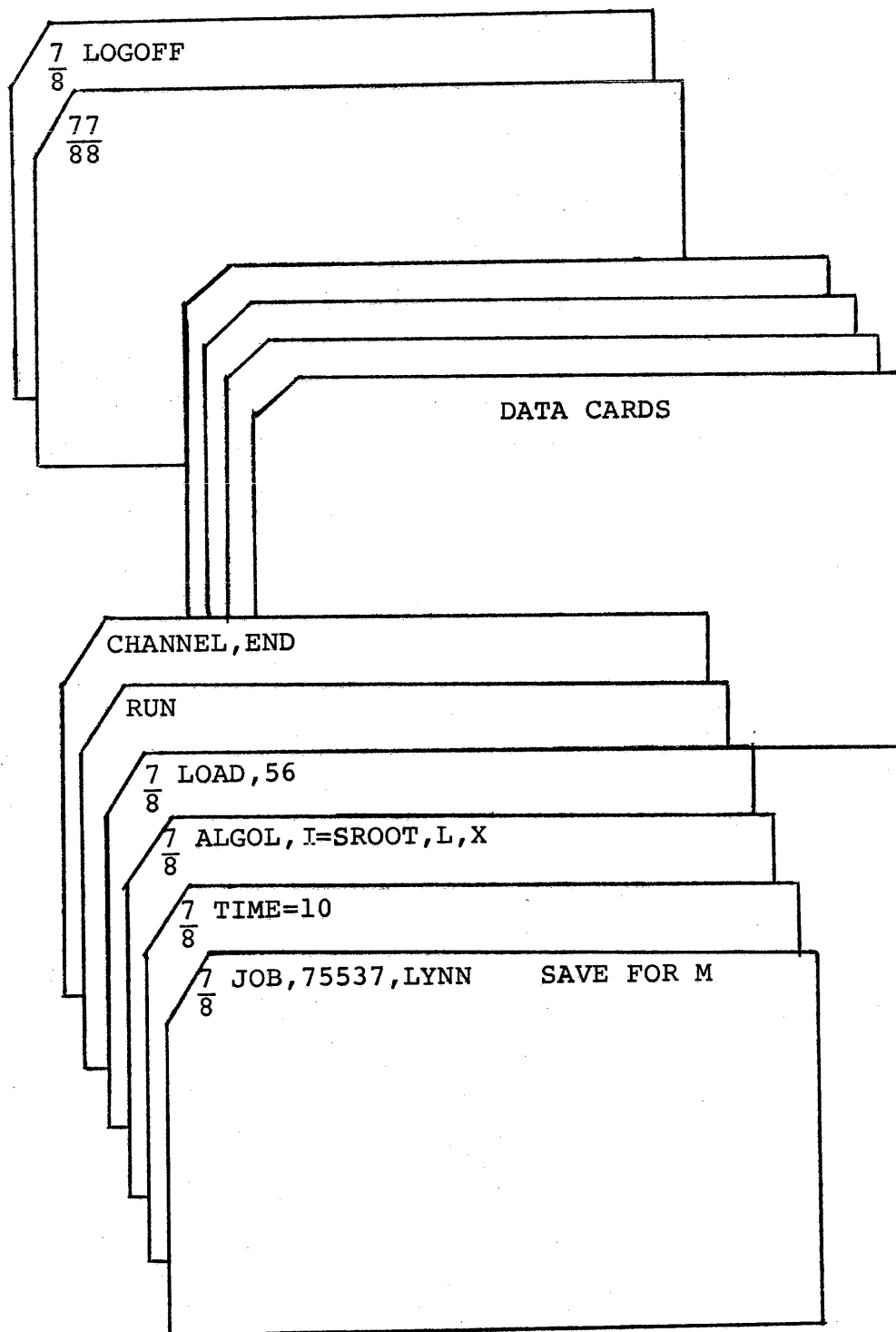
A batch job consists of a deck of cards submitted by the user to the Computer Center. The computer operator places this deck in the card reader, along with other job decks. OS-3 reads in the decks and processes each job in turn. The user may refer to the Computer Center User's Manual, cc-69-10, for additional information.

The following are sample control cards and deck structures for the batch operation of ALGOL programs.

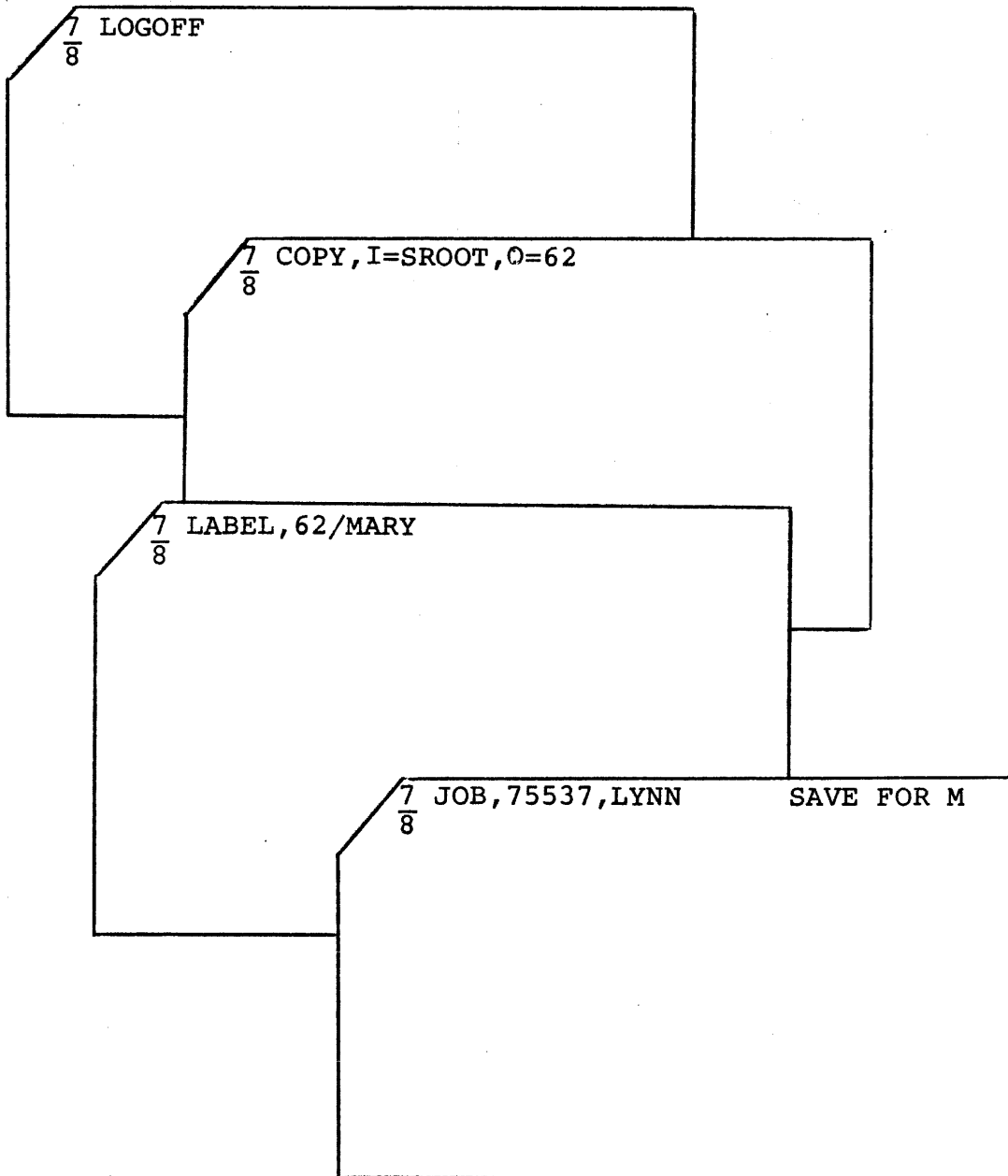
1. TO RUN AN ALGOL SOURCE DECK



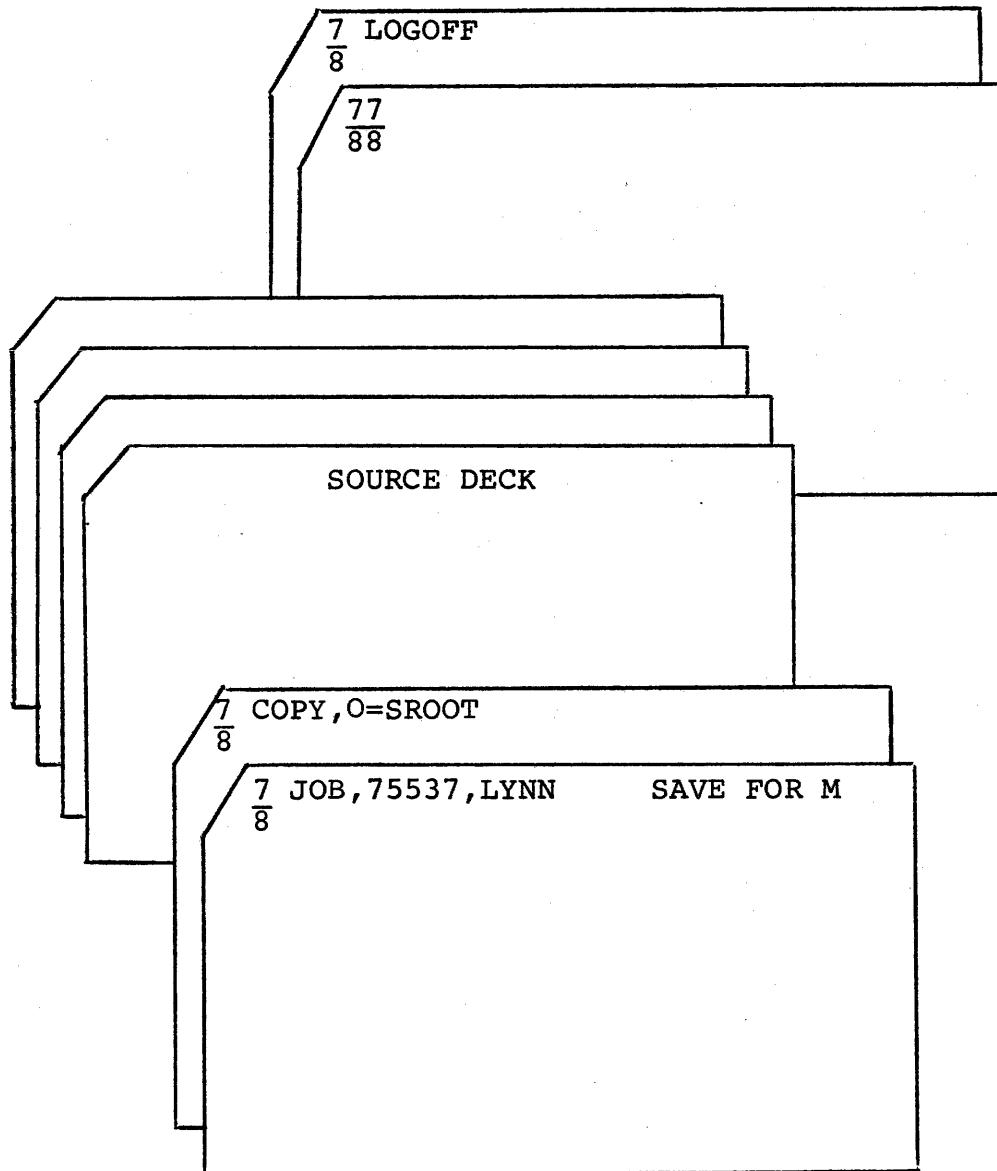
2. TO RUN AN ALGOL PROGRAM THAT IS ON FILE



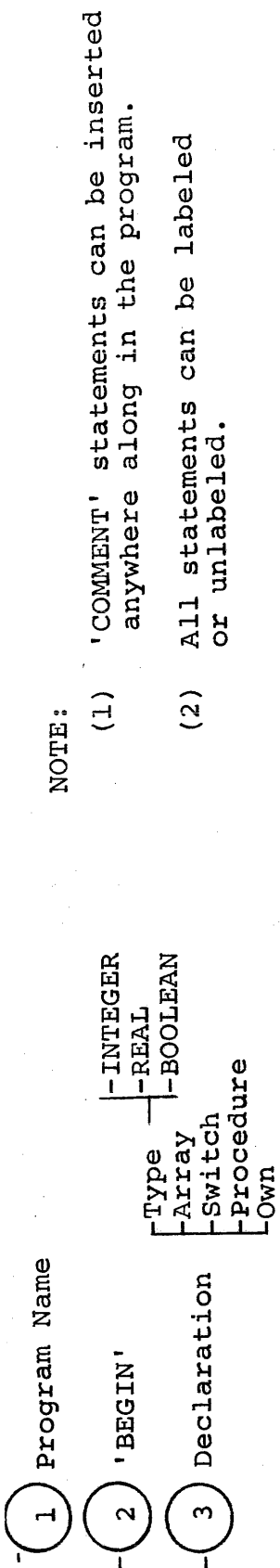
3. TO PUNCH A CARD DECK FROM A PROGRAM ON FILE



4. TO COPY A SOURCE DECK ONTO A SAVED FILE



PART TWO



NOTE:

- (1) 'COMMENT' statements can be inserted anywhere along in the program.
- (2) All statements can be labeled or unlabeled.

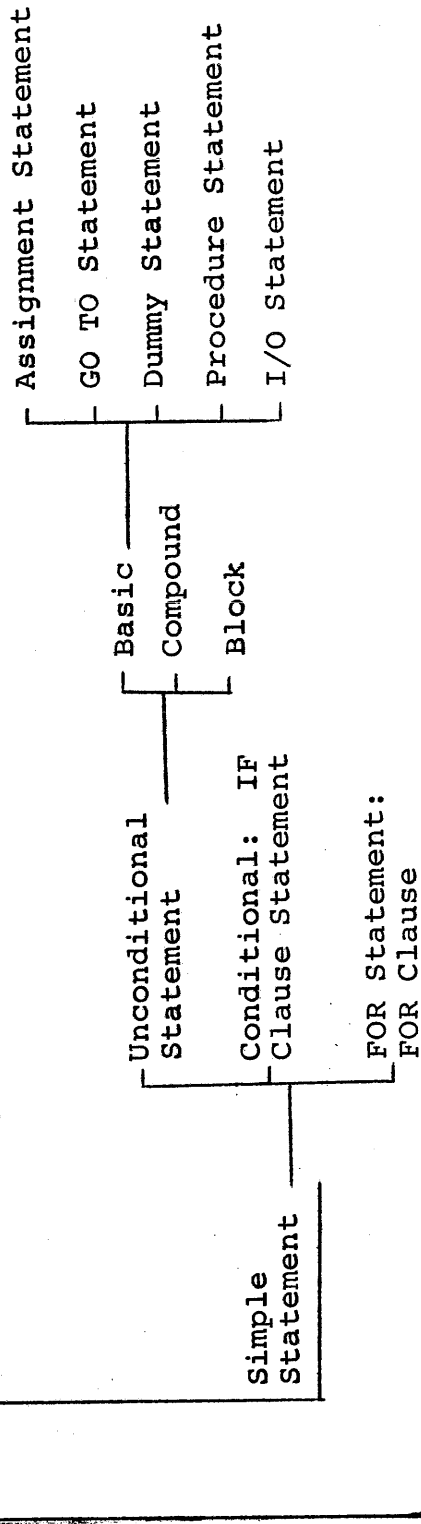


CHART I. PROGRAM SOURCE DECK STRUCTURE

I ELEMENTS OF SIMPLE PROGRAM

1. ILLUSTRATION ON USE OF 'BEGIN', 'COMMENT', 'END', and 'EOP'

```
..... (a)
.....
'BEGIN' (b)
.....
'COMMENT' (c)
.....
'END' (d)
'EOP' (e)
```

'BEGIN', 'END', 'EOP' and 'COMMENT' have the following definitions and/or equivalencies:

- (a) Any words before the first 'BEGIN' are comment. However, the first 8 characters are designated to be the identification of the program.
- (b) 'BEGIN' - The first executable symbol of an ALGOL program, a Block or a Compound Statement. 'BEGIN' 'COMMENT' (LIST) is equivalent to 'BEGIN', where (LIST) is any sequence not containing a ;.
- (c) 'COMMENT' - This word is used to indicate that an explanatory comment follows.
; 'COMMENT' (LIST) is equivalent to ;, where (LIST) is any sequence not containing a ;.
- (d) 'END' - This indicates the physical end to a compound statement. Each 'END' corresponds to each 'BEGIN'.
'END' (LIST) is equivalent to 'END', where (LIST) is any sequence not containing 'END' or ; or 'ELSE'.
- (e) 'EOP' - Indication of end of a program (columns 10-14) following the last 'END'.

2. SAMPLE PROGRAM

This program illustrates the source deck of a simple program.

```
#ALGOL,I=STRU,X,L
```

```
OS3 ALGOL  V0.0      SIMPLE      02/20/70      1412      PAGE  1
SIMPLE  THE NAME OF THIS PROGRAM IS SIMPLE
'BEGIN' 'REAL'A,B,SUM;
'COMMENT' THIS PROGRAM WILL READ 2 REAL NUMBERS,
ADD THEM AND PRINT OUT THE SUM;
INPUT(60,'(' ')',A,B);
'COMMENT' THE ABOVE INPUT STATEMENT WITH FORMAT STRING
'(' ') IS EQUIVALENT TO INREAL(60,A) AND
INREAL(60,B) WHICH WILL FREEFORM INPUT A AND B;
SUM:=A+B;
10** OUTPUT(61,'('*')');
'COMMENT' THIS WILL GIVE A PAGE EJECT;
OUTREAL(61,SUM);
OUTPUT(61,'('/')');
'COMMENT' / MEANS NEW LINE.

'END'
'EOP'
```

(01)	LINE 0002	PROGRAM BEGINS
(01)	LINE 0017	PROGRAM ENDS
(01)	LINE 0017	SOURCE DECK ENDS

```
#LOAD,56
RUN
RUN
```

```
CHANNEL, 60=LU60,P80
CHANNEL, 61=LU61,P136,PP60
CHANNEL,END
CHANNEL,END
2.0 3.456
```

```
+5.456000000
```

```
END OF ALGOL RUN
```

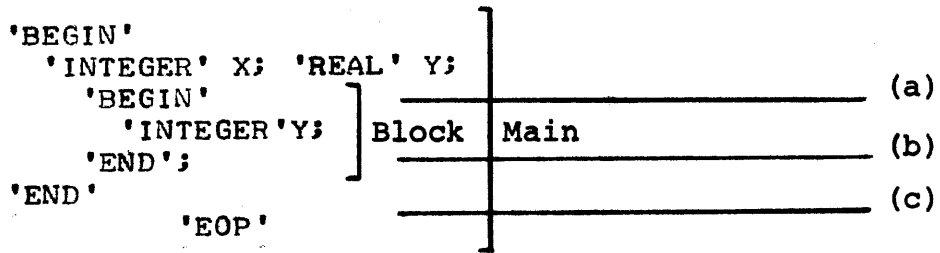
```
#
```

II DECLARATIONS

All identifiers of a program, except standard functions and labels must be declared by one of the following:

'OWN', 'BOOLEAN', 'INTEGER', 'REAL', 'ARRAY', 'SWITCH', 'PROCEDURE'

A declaration of an identifier must appear after the first 'BEGIN' of a block and is only valid for one block. For example, observe the following program:



- At point
- (a) X = integer, Y = real
 - (b) X = integer, Y = integer
 - (c) X = integer, Y = real

1. TYPE DECLARATIONS

'INTEGER'

All integers are of this type. They may be positive or negative integer values including zero.

Example: 'INTEGER' P,Q,S;

'BOOLEAN'

The BOOLEAN declarator. 'BOOLEAN' declared variables may only assume the values 'TRUE' and 'FALSE'. (For BOOLEAN FORMAT, see format description page 56.

Example: 'BOOLEAN' C;

'OWN'

A declaration may be marked with the additional declarator 'OWN'.

The effect is that upon re-entry into the block containing OWN quantities, they will remain the same as their last values at the last exit. However, OWN values are initially defined as zero at their first entry in the block.

AN EXAMPLE ON USE OF 'OWN':

```
OWN EXAM
'BEGIN'
  'INTEGER' A,B,I;
  'FOR' I:=1 'STEP' 1 'UNTIL' 3 'DO'
    'BEGIN' A:=1; B:=2;
    'BEGIN'
      'OWN' 'INTEGER' A;
      A:=A+1; B:=B+1;
      OUTPUT(61, (('('A='), ZDBBB, ('B='), ZD'), A,B);
      OUTPUT(61, ('/'));
    'END';
  'END';
'END'
'EOP'
```

Intermediate Analysis of the Program at Points (a) and (b)

	I=1		I=2		I=3	
	(a)	(b)	(a)	(b)	(a)	(b)
A	**0	1	1	2	2	3
B	2	3	2	3	2	3

** A is an own value, it is initially defined as zero in the block.

2. ARRAY DECLARATIONS

An array declaration does the following:

- * Declares one or several subscripted variables (arrays).
- * Gives dimension of the arrays.
- * Gives the bounds of the subscripts.
- * Declares the type of the variable.

It is in the form of:

$A[L_1:U_1, L_2:U_2, L_3:U_3, \dots]$

where

L_1, L_2, L_3, \dots , are the lower bounds, and

U_1, U_2, U_3, \dots , are the upper bounds.

Both L and U can be any arithmetic expression

Limitation: A is defined if and only if $U_i \geq L_i$ for all i.

Example:

```
'BEGIN'  
'REAL' 'ARRAY' B[2:3,4:5];  
'INTEGER' 'ARRAY' IA['IF' C < 0 'THEN' 2 'ELSE' 1:3];
```

NOTE: C has to be defined before declaration is processed during run time, and it must be declared in higher block.

Analysis of Example

Array IA

```
* Type: Integer  
* Dimension: 1  
*  $L_1 = 2$  if  $C < 0$        $U_1 = 3$   
   $L_1 = 1$  if  $C \geq 0$        $U_1 = 3$ 
```

Array B

```
* Type: Real  
* Dimension: 2  
*  $L_1 = 2, U_1 = 3$   
   $L_2 = 4, U_2 = 5$ 
```

NOTE: If [and] are not presented on the keyboard, use (/ and /) instead.

3. SWITCH DECLARATIONS AND CALLING

A SWITCH declaration defines the set of values of the corresponding switch designators. With each of the designational expressions there is associated a positive integer obtained by counting the items in the list from left to right.

EXAMPLE:

'SWITCH' TEST: = S1, S2, S3, S4; where TEST is the switch identifier; S1, S2, S3, and S4 are designational expressions with integer values 1, 2, 3, and 4 associated with them respectively.

i.e. TEST [1] = S1; TEST [2] = S2;
 TEST [3] = S3; TEST [4] = S4;

the corresponding calling statement should be: GO TO TEST [I] where I is any arithmetic expression of integer value.

SAMPLE PROGRAM ON THE USE OF SWITCH

Program is to find angle and compute cube root of a complex number. Coordinates in the complex plane are input as X and Y.

SWITCH in this program is used to branch on 9 possible conditions

X < 0, Y < 0

X < 0, Y = 0

X < 0, Y > 0

etc.

$$\sqrt[3]{X+iy} = \begin{cases} 0 & \text{if } X=0, Y=0 \\ R^{\frac{1}{3}} (\cos \frac{\theta}{3} + i \sin \frac{\theta}{3}) & \text{otherwise} \end{cases}$$

$$R = \sqrt{X^2 + Y^2}$$

$$\theta = \begin{cases} \pi + \text{TAN}^{-1}(Y/X) & X < 0 \\ \text{TAN}^{-1}(Y/X) & X > 0 \\ -\pi/2 & X = 0, Y < 0 \\ \pi/2 & X = 0, Y > 0 \end{cases}$$

ALGOL, I=SWITCH, X, L

OS3 ALGOL V0.0 SWITCH 02/20/70 1422 PAGE 1

SWITCH TO FIND CUBE ROOT OF A COMPLEX NUMBER

'BEGIN'

'REAL' X, Y, U, V, THETA, PI, R;

'SWITCH' ANGLE:=S1, S1, S1, S2, S3, S4, S5, S5, S5;

EOF(60, END);

PI:=3.14159265;

IN: INPUT(60, '(')', X, Y);

'GOTO' ANGLE[3*SIGN(X)+SIGN(Y)+5];

S1: THETA:=PI+ARCTAN(Y/X);

10**

'GOTO' S6;

S2: THETA:=-PI/2;

'GOTO' S6;

S3: U:=0; V:=0;

'GOTO' OUT;

S4: THETA:=PI/2;

'GOTO' S6;

S5: THETA:=ARCTAN(Y/X);

S6: R:=(X*X+Y*Y)^{1/6};

U:=R*COS(THETA/3);

V:=R*SIN(THETA/3);

20**

OUT: OUTPUT(61, '('('CUBE ROOT OF ')', N, N, '('('*I')')', X, Y);

OUTPUT(61, '('('/')');

OUTPUT(61, '('(' IS ')', N, N,

'('*I')')', U, V);

OUTPUT(61, '('('///')');

'GOTO' IN;

END: 'END';

'EOP'

(01) LINE 0002 PROGRAM BEGINS

(01) LINE 0028 PROGRAM ENDS

(01) LINE 0028 SOURCE DECK ENDS

LOAD, 56

RUN

RUN

CHANNEL, 60=LU60, P80

CHANNEL, 61=LU61, P136, PP60

C HANNEL, END

CHANNEL, END

5 5

CUBE ROOT OF +5.000000000' 000 +5.000000000' 000 *I
IS +1.853981710' 000 +4.967729021'-001 *I

7 0

CUBE ROOT OF +7.000000000' 000 +0.000000000'+000 *I
IS +1.912931183' 000 +0.000000000'+000 *I

4 -8

CUBE ROOT OF +4.000000000' 000 -8.000000000' 000 *I
IS +1.936020528' 000 -7.487949627'-001 *I

-6 -3

CUBE ROOT OF -6.000000000' 000 -3.000000000' 000 *I
IS +6.803253753'-001 +1.758991384' 000 *I

-1 7

CUBE ROOT OF -1.000000000' 000 +7.000000000' 000 *I
IS +1.614999957' 000 +1.037210989' 000 *I

-10 -10

CUBE ROOT OF -1.000000000'+001 -1.000000000'+001 *I
IS +6.258946390'-001 +2.335870582' 000 *I

0 4

CUBE ROOT OF +0.000000000'+000 +4.000000000' 000 *I
IS +1.374729637' 000 +7.937005252'-001 *I

0 0

CUBE ROOT OF +0.000000000'+000 +0.000000000'+000 *I
IS +0.000000000'+000 +0.000000000'+000 *I

0 -2.7

CUBE ROOT OF +0.000000000'+000 -2.700000000' 000 *I
IS +1.205920154' 000 -6.962383244'-001 *I

4. PROCEDURE DECLARATIONS AND CALLING

PROCEDURE DECLARATION.

A procedure declaration serves to define the procedure associated with a procedure identifier. The principal constituent of a procedure declaration is a statement, the procedure body, which, through the use of procedure statements and/or function designators may be activated from other parts of the block where the procedure declaration appears. There are 2 types of procedures, namely, function-type procedure and non-function-type procedure.

EXAMPLE 1: FUNCTION-TYPE PROCEDURE

```
(c)
(a) 'REAL' 'PROCEDURE' AVERAGE (LOWER, UPPER);
      (e) 'VALUE' LOWER, UPPER;
          'REAL' LOWER, UPPER;
          'BEGIN'
              (f) AVERAGE: = (LOWER + UPPER)/2;
          'END';
```

EXAMPLE 2: NON-FUNCTION-TYPE PROCEDURE

```
(d)
(b) 'PROCEDURE' TRANSPOSE (A) ORDER: (N);
      'VALUE' N;
      'ARRAY' A;
      'INTEGER' N;
      'BEGIN' 'REAL' TEMP;
          (g) 'INTEGER' I, J;
              'FOR' I: = 1 'STEP' 1 'UNTIL' N 'DO'
                  'FOR' J: = I+1 'STEP' 1 'UNTIL' N 'DO'
                      'BEGIN'
                          TEMP: = A[I, J];
                          A[I, J]: = A[J, I];
                          A[J, I]: = TEMP;
                      'END';
          'END' TRANSPOSE;
```

EXPLANATION

- (a) For function-type procedure, the procedure identifier must be declared through the appearance of a type declaration as the very first symbol of the procedure declaration. Type of procedure can be 'INTEGER' or 'REAL' or 'BOOLEAN'.
- (b) No type declaration of procedure is needed for non-function-type procedure.
- (c) AVERAGE is the procedure identifier, whereas (LOWER,UPPER) constitutes the formal parameter list. When no parameters are to be passed, the list is empty. Formal parameters are separated by commas, the parameter delimiters.
- (d) TRANSPOSE (A) ORDER:(N) is equivalent to TRANSPOSE(A,N), where TRANSPOSE is the procedure identifier, A and N are formal parameters.)letter string:(is another representation of parameter delimiter.
- (e) Value and specification part for formal parameters can be empty or of the following form:
Value part: 'VALUE' identifier list
Specification part: it has to follow the value part and must be supplied for all formal parameters in CDC ALGOL.
- (f) The procedure body can be a block, a compound statement or a simple statement. For function-type procedure declaration, one or more assignment statements with the procedure identifier in a left part must appear in the procedure body.
- (g) For non-function-type procedure declaration, the procedure identifier is not to appear as a left part of assignment statement but may occur as procedure statement calling itself. (Recursively)

PROCEDURE CALLING

The actual parameter list of the procedure statement must have the same number of entries as the formal parameter list of the procedure declaration heading.

EXAMPLE 3: FUNCTION TYPE PROCEDURE CALLING

'REAL' X,Y,S;

.
.
.
.

(a) S: = S+AVERAGE(X,Y);

EXAMPLE 4: NON-FUNCTION TYPE PROCEDURE CALLING

'REAL' 'ARRAY' B[1:20,1:20];

'INTEGER' M;

.
.
.

(b) TRANSPOSE(B,M);

.
.
.

EXPLANATION

- (a) Function-type procedure can be called in an expression, or by a procedure statement.
- (b) Non-function-type procedure is called by the procedure statement itself. A procedure statement is a procedure identifier followed by the actual parameter list.

5. SAMPLE PROGRAM (on declaration)

Write a procedure to produce the roots of a quadratic equation and a switch to determine the nature of the roots. The main program is used to read the input, call the sub-program, and produce the output.

quadratic equations $ax^2 + bx + c = 0$

$$r_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$$

$$r_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$$

or

$$r_1 = R_1 + R_2$$

$$r_2 = R_1 - R_2$$

where

$$R_1 = \frac{-b}{2a}$$

$$R_2 = \sqrt{\frac{b^2 - 4ac}{2a}}$$

CODE	NATURE OF THE ROOTS	CONDITION
1	2 unequal imaginary roots	$b^2 - 4ac < 0$
2	2 equal real roots	$b^2 - 4ac = 0$
3	2 unequal real roots	$b^2 - 4ac > 0$
4	1 real root = $-\frac{c}{b}$	$a = 0$

#ALGOL,I=DECLAR,X,L

```
OS3 ALGOL V0.0      DECLARAT      02/24/70      1126      PAGE      1
  DECLARATION AND PROCEDURE
  'BEGIN'
    'REAL'D,E,F,P1,P2;
    'INTEGER'ICODE;
    'SWITCH' COD:=AA,BB,CC,DD;
    'PROCEDURE' ROOT(A,B,C,R1,R2,ID);
    'VALUE'A,B,C;
    'REAL'R1,R2,A,B,C;
    'INTEGER'ID;
10**  'BEGIN'
      'REAL'DET;
      'IF' A=0 'THEN' 'GO TO' L1;
      DET:=B2-4*A*C;
      R1:=-B/(2*A);
      R2:=SQRT(ABS(DET))/(2*A);
      'IF' DET>0 'THEN'ID:=3 'ELSE'
      'IF'DET=0 'THEN' ID:=2 'ELSE'ID:=1;
      'GOTO'END;
      L1: ID:=4;
20**  R1:=-C/B;
      END: 'END'ROOT;
  START: P1:=0; P2:=0; ICODE:=0;
        INPUT(60,(' '),D,E,F);
        OUTPUT(61,('//('COEFF OF QUAD ARE')//'));
        OUTPUT(61,(' '),D,E,F);
        OUTPUT(61,('//'));
        EOF(60,END);
        ROOT(D,E,F,P1,P2,ICODE);
        'GOTO' COD[ICODE];
30**  AA: OUTPUT(61,('('2 UNEQUAL IMAG. ROOTS')'));
      OUTPUT(61,('//'));
      OUTPUT(61,('(' ' '),N,(' +I* '),N)'),P1,P2);
      OUTPUT(61,('//'));
      OUTPUT(61,('(' ' AND '),N,(' -I* '),N)'),P1,P2);
      OUTPUT(61,('//'));
      'GOTO'START;
      BB: OUTPUT(61,('('2 EQUAL REAL ROOTS')'));
      OUTPUT(61,('//'));
      OUTPUT(61,('(' ' '),N)'),P1);
```



```

40**      OUTPUT(61, '(//)');
          'GOTO' START;
CC:      OUTPUT(61, '('('2 UNEQUAL REAL ROOTS')')');
          OUTPUT(61, ('/'));
          OUTPUT(61, '('('      ')', N, ('      AND ')',
          N')', P1+P2, P1-P2);
          OUTPUT(61, '(//)');
          'GOTO' START;
DD:      OUTPUT(61, '('('1 REAL ROOT')')');
          OUTPUT(61, ('/'));
50**      OUTPUT(61, '('('      ')', N')', P1);
          OUTPUT(61, '(//)');
          'GOTO' START;
          END: 'END'
          'EOP'
(01)     LINE 0002     PROGRAM BEGINS
(01)     LINE 0053     PROGRAM ENDS
(01)     LINE 0053     SOURCE DECK ENDS

```

```

#LOAD, 56
RUN
RUN

```

```

CHANNEL, 60=LU 60, P80
CHANNEL, 61=LU 61, P136, PP 60
CHANNEL, END
CHANNEL, END
0 2 -6

```

COEFF OF QUAD ARE
+0.000000000'000 +2.000000000' 000 -6.000000000' 000

1 REAL ROOT
+3.000000000' 000

1 2 1

COEFF OF QUAD ARE
+1.000000000' 000 +2.000000000' 000 +1.000000000' 000

2 EQUAL REAL ROOTS
-1.000000000' 000

1 -5 4

COEFF OF QUAD ARE
+1.000000000' 000 -5.000000000' 000 +4.000000000' 000

2 UNEQUAL REAL ROOTS
+4.000000000' 000 AND +1.000000000' 000

4 -3

COEFF OF QUAD ARE
+4.000000000' 000 -3.000000000' 000 +1.000000000' 000

2 UNEQUAL IMAG. ROOTS
+3.750000000'-001 +I*+3.307189139'-001
AND +3.750000000'-001 -I*+3.307189139'-001

III. OPERATORS AND VALUES

1. RELATIONAL OPERATOR

'EQUAL' : =
'GREATER' : >
'LESS' : <
'NOT EQUAL' : ≠
'NOT GREATER' : <_
'NOT LESS' : >_

Relational operators are used to connect simple arithmetic expressions in Boolean expressions.††

2. LOGICAL OPERATOR

'EQUIV' : ≡ (equivalent)
'IMPL' : ⊃
'OR' : ∨ (inclusive or)
'AND' : ∧
'NOT' : ¬

3. ARITHMETIC OPERATOR

+ : ADD
- : SUBTRACT
* : MULTIPLY
/ : DIVIDE

'POWER'
or ↑ : EXPONENTIAL

EXAMPLE: C:= SQRT (A↑2 + B↑2);

†† Arithmetic expressions (IV.2)
Boolean expressions (IV.3)

4. LOGICAL VALUE

'TRUE', 'FALSE'

NOTE: Logical values cannot be used in place of integers, or vice-versa.

5. SAMPLE PROGRAM ON OPERATORS

#ALGOL, I=OPERAT, X, L

OS3 ALGOL VO.0 OPERATOR 02/24/70 1134 PAGE 1

```
OPERATOR
'BEGIN' 'BOOLEAN' Q, R, P, Z, W;
'REAL' X, Y;
X:=3.463;
Y:=3.6;
Q:=X=Y;
'COMMENT' Q IS FALSE;
R:=X<Y;
'COMMENT' R IS TRUE;
10** Z:=Q'AND'R;
'COMMENT' Z IS FALSE;
W:=Q'OR'R;
'COMMENT' W IS TRUE;
P:=Z'AND'W;
'COMMENT' P IS FALSE;
OUTPUT(61, '('('Z IS '), 5F)', Z);
OUTPUT(61, ('//'));
OUTPUT(61, '('('W IS '), 5F)', W);
OUTPUT(61, ('//'));
20** OUTPUT(61, '('('W AND Z IS '), 5F)', P);
OUTPUT(61, ('//'));
'END'
'EOP'
(01) LINE 0002 PROGRAM BEGINS
(01) LINE 0022 PROGRAM ENDS
(01) LINE 0022 SOURCE DECK ENDS
```

#LOAD, 56
RUN
RUN

CHANNEL, 60=LU60, P80
CHANNEL, 61=LU61, P136, PP60
CHANNEL, END
CHANNEL, END

Z IS F

W IS T

W AND Z IS F

END OF ALGOL RUN

#

IV. EXPRESSIONS

1. FUNCTION DESIGNATORS

Function designator is a procedure identifier followed by actual parameter list (actual parameter list can be empty.) It defines single numerical or logical values. There are also standard functions. For example: $\text{SIN}(X)++$.

EXAMPLES:

$\text{COS}(A+B)$

$\text{AVERAGE}(N, X, Y)$ or $\text{AVERAGE}(N)$ HIGH: (X) LOW: (Y)

2. ARITHMETIC EXPRESSIONS (Including IF clause)

EXAMPLES:

- 1) 'IF' A < 0 'THEN' B+C
'ELSE' 'IF' A = 0 'THEN' B/C
'ELSE' D
- 2) $W * U - V + 2$

PRECEDENCE OF OPERATORS

- i) +
- ii) *, /
- iii) +, -

3. BOOLEAN EXPRESSION

EXAMPLES:

- 1) $X = -2$
- 2) $Q \equiv \neg A \wedge B \vee C$

++ See section IV.5 for standard functions list

PRECEDENCE OF OPERATORS

- i) Arithmetic operators
- ii) $<, \leq, =, \geq, >, \neq$
- iii) \neg
- iv) \wedge
- v) \vee
- vi) \supset
- vii) \equiv

4. DESIGNATIONAL EXPRESSIONS

Labels and switch designators

EXAMPLE:

```
COD[ICODE]
ANGLE[3*SIGN(X) + SIGN(Y) + 5]
L1
```

5. STANDARD FUNCTIONS

ABS(E): absolute value of the expression E

SIGN(E): +1 for $E > 0$
0 for $E = 0$
-1 for $E < 0$

SQRT(E): square root of E

SIN(E): sine of E

COS(E): cosine of E

ARCTAN(E): arctangent of E

LN(E): natural logarithm of E

EXP(E): exponential function of E

ENTIER(E): largest integer value not greater than the value
of E

V. STATEMENTS

1. COMPOUND STATEMENTS

A sequence of statements enclosed by 'BEGIN' and 'END'.

SYNTAX:

```
'BEGIN' S; S; ...S; S 'END'
```

(Where S stands for statements, it can be again a complete statement or block.)

EXAMPLE:

```
'BEGIN'  
  SUM: = 0;  
  'FOR' I: = 1 'STEP' 1 'UNTIL' N 'DO'  
    SUM: = SUM + A[I]  
'END'
```

2. BLOCK

A sequence of declarations followed by a sequence of statements and enclosed between 'BEGIN' and 'END'.

NOTE: Every declaration that appears in a block is valid only in that block.

SYNTAX:

```
'BEGIN' D; D; ... D; S; S; ... S 'END'
```

Where S = statements and D = declarations.

Example of Block Structure

```
# ALGOL, I=BLOCK, X, L
```

```
OS3 ALGOL  V0.0          BLOCK          02/24/70      1137  PAGE  1
BLOCK
  'BEGIN' 'INTEGER' I, J;
  INREAL(60, I); INREAL(60, J);
  OUTPUT(61, '('//')');
  OUTPUT(61, '('('I IS '),+ZZD, '(' J IS '),+ZZD')', I, J);
  BLOCK: 'BEGIN'
    'INTEGER' K;
    'COMMENT' THIS IS A BLOCK WHICH
      WILL INTERCHANGE TWO VALUES;
10**      K:=I;
          I:=J;
          J:=K;
    'END' BLOCK;
  OUTPUT(61, '('//')');
  OUTPUT(61, '('('EXCHANGE I AND J')')');
  OUTPUT(61, '('//')');
  OUTPUT(61, '('('I IS '),+ZZD, '(' J IS '),+ZZD')', I, J);
  'END'
    'EOP'
(01)  LINE 0002  PROGRAM BEGINS
(01)  LINE 0018  PROGRAM ENDS
(01)  LINE 0018  SOURCE DECK ENDS
```

```
#LOAD,56
RUN
RUN
```

CHANNEL, 60=LU60, P80
CHANNEL, 61=LU61, P136, PP60
CHANNEL, END
CHANNEL, END
2 -5

I IS +2 J IS -5
EXCHANGE I AND J
I IS -5 J IS +2

END OF ALGOL RUN

#

3. ASSIGNMENT STATEMENTS

SYNTAX:

left part variable: = Arithmetic expression[†]
or Boolean expression^{††}

where left part variable is a variable or a procedure identifier.
Any number of left part variables may appear at the left part of
assignment statement.

EXAMPLES:

S: = N: = S+N;

(If S is 2 and N is 6, then S and N will be of the value 8 after
the assignment statement.)

A: = B+C - B/C ;

W: = U \wedge V;

4. GO TO STATEMENT

SYNTAX:

'GO TO' designational expression;^{†††}

EXAMPLE:

'GO TO' COD[ICODE];

'GO TO' ANGLE[3*SIGN(X) + SIGN(Y) + 5];

'GO TO' L1;

5. DUMMY STATEMENT

A dummy statement executes no operation. It is empty and may
serve to place a label.

[†] Arithmetic Expressions (IV.2)

^{††} Boolean Expressions (IV.3)

^{†††} Designational Expressions (IV.4)

EXAMPLE:

```
START:   'BEGIN'  
        .  
        .  
        .  
        .  
END      'END'
```

6. CONDITIONAL AND UNCONDITIONAL STATEMENTS

CONDITIONAL STATEMENT: LEGAL STRUCTURE

- (a) 'IF' B1 'THEN' S1;
- (b) 'IF' B1 'THEN' S1 'ELSE' S2;
- (c) 'IF' B1 'THEN' S1 'ELSE' 'IF' B2 'THEN' S2
 'ELSE' S3; S4;
- (d) 'IF' 'IF' 'IF' B1
 'THEN' B2 'ELSE' B3
 'THEN' B4 'ELSE' B5
 'THEN' S1 'ELSE' S2;

B1, B2, B3, B4, B5 are Boolean Expressions (IV.3)
S1, S2, S3 are unconditional statements. S4 is the
statement following the complete conditional statement.

UNCONDITIONAL STATEMENTS

- (a) Basic statements:
 - *Assignment statements (V.3)
 - *GO TO statements (V.4)
 - *Dummy statements (V.5)
 - *Procedure statements (V.7 or II.4)
- (b) Compound statements (V.1)
- (c) Block (V.2)

SIMPLE EXAMPLE OF CONDITIONAL STATEMENTS

```
'IF' A < B 'THEN' S: = B ↑ 2  
'ELSE' 'GO TO' L1;
```

'IF': Beginning of a conditional statement.

'THEN': This word ends the IF clause and precedes the true alternative in a conditional statement.

'ELSE': This word follows the true alternative and precedes the false alternative.

'GO TO': Transfer control to the destination.

No GO TO statement can lead from outside into a block.

EXECUTION OF CONDITIONAL STATEMENT

Conditional statement causes certain statements to be executed or skipped depending on the running values of specified Boolean expressions.

```
                                B1 or B2 is true  
                                ↑-----↓  
'IF' B1 'THEN' S1 'ELSE' 'IF' B2 'THEN' S2 'ELSE' S3; S4  
    ↓-----↑                ↓-----↑  
    B1 is false                B2 is false
```

Execution of S1, S2, S3, S4 are as follows:

- (1) True B1: S1; S4;
- (2) False B1, True B2: S2; S4;
- (3) False B1, False B2: S3; S4;

PROGRAM ON ILLUSTRATION OF IF CLAUSE

#ALGOL,I=CONDIT,X,L

OS3 ALGOL V0.0 CONDITIO 02/24/70 1140 PAGE 1

CONDITIONAL STATEMENT - IF CLAUSE
THIS PROGRAM WILL CALCULATE FACTORIAL
F(N)=N! =N*(N-1)*(N-2)*.....*2*1
WHERE N IS LESS THAN 10

```
'BEGIN' 'INTEGER' 'ARRAY' F[1:9];
      'INTEGER' N, I;
START: OUTPUT(61, '('('ENTER N')', '//')');
      INREAL(60, N);
10**   'IF' N < 10 'THEN'
      'BEGIN'
      'FOR' I := 1 'STEP' 1 'UNTIL' N 'DO'
      F[I] := 'IF' I < 2 'THEN' 1
      'ELSE' I * F[I-1];
      OUTPUT(61, ('D, ('! IS '),
      ZZZZD')', N, F[N]);
      OUTPUT(61, ('//')');
      'GOTO' START;
      'END';
20**   OUTPUT(61, ('DD, (' IS GREATER')',
      (' THAN 9, STOP HERE')')', N);
      'END'
```

```
'EOP'
(01)  LINE 0006      PROGRAM BEGINS
(01)  LINE 0022      PROGRAM ENDS
(01)  LINE 0022      SOURCE DECK ENDS
```

#LOAD,56
RUN
RUN

CHANNEL, 60=LU60, P80
CHANNEL, 61=LU61, P136, PP60
CHANNEL, END
CHANNEL, END

ENTER N

1
1! IS 1

ENTER N

4
4! IS 24

ENTER N

9
9! IS 362880

ENTER N

10
10 IS GREATER THAN 9, STOP HERE

END OF ALGOL RUN

#

7. PROCEDURE STATEMENTS

See section II.4

8. FOR STATEMENT

STRUCTURE OF 'FOR' STATEMENT:

- (1) 'FOR' TEST:=A 'STEP' B 'UNTIL' C 'DO' S;
- (2) 'FOR' TEST:=E 'WHILE' F 'DO' S;

A,B,C,E: Arithmetic expressions

S: Statement

'FOR': The beginning of a 'FOR' statement

'STEP': This identifies increment value in a 'FOR' statement

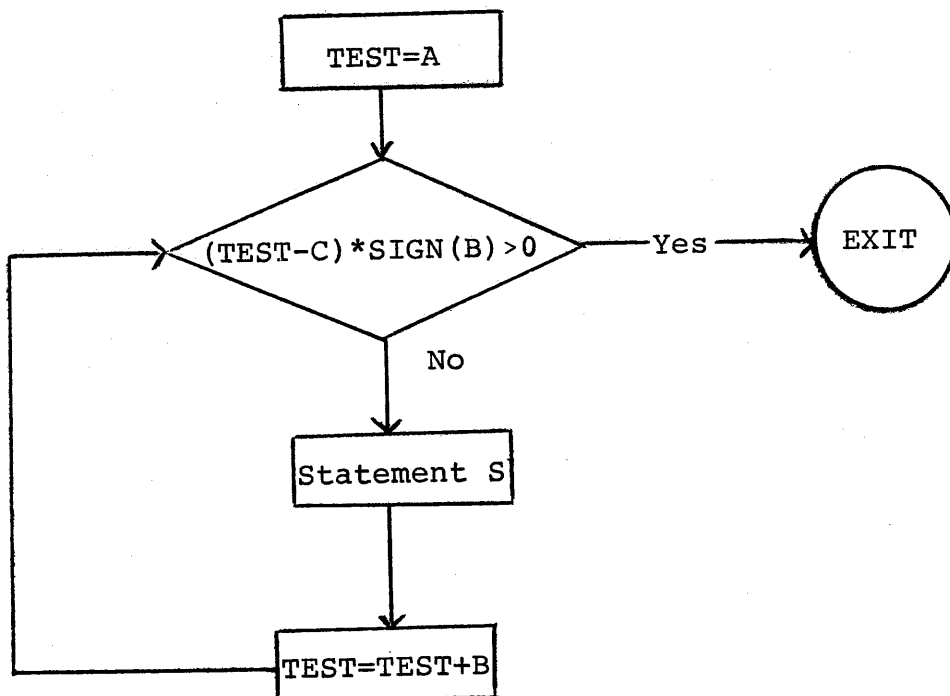
'UNTIL': This word identifies the final value in a 'FOR' statement

'WHILE': Separator in a 'FOR' statement

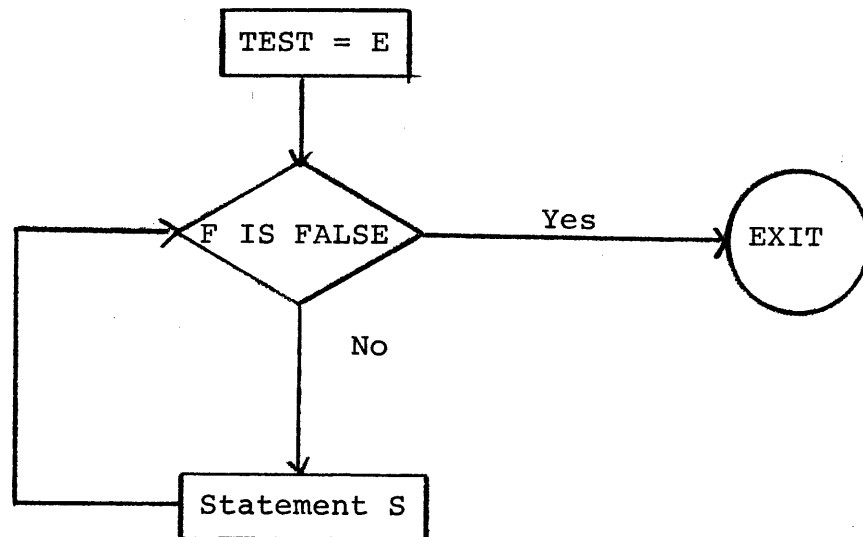
'DO': This word causes the statement that follows it to be executed

TEST: Variable name

FUNCTION OF 'STEP' - 'UNTIL' IN A 'FOR' STATEMENT



FUNCTION OF 'WHILE' IN A 'FOR' STATEMENT



EXAMPLE

'FOR' Statement

```
'FOR' I:=1 'STEP' 1
'UNTIL' N 'DO' A(I):=I;
```

```
Y:=APP:
'FOR' X:=(2.0*Y+A/(Y+2))/3.0
'WHILE' ((Y-X)/X)'GEQ'0.0001
'DO' Y:=X;
```

Equivalent Condition Statement

```
I:=1;
L1:'IF' I-N>0
'THEN' 'GO' 'TO' L2;
A(I):=I;
I:=I+1;
'GO' 'TO' L1;
L2: ...
```

```
Y:=APP;
L1:X:=(2.0*Y+A/(Y+2))/3.0;
'IF' (Y-X)/X<(0.0001)
'THEN' 'GO' 'TO' L1;
Y:=X; 'GO' 'TO' L1;
L2:...
```

PROGRAM ON ILLUSTRATION OF FOR STATEMENT

#ALGOL,I=FOR,X,L

OS3 ALGOL VO.0 FOR STAT 02/24/70 1143 PAGE 1

FOR STATEMENT STRUCTURE ILLUSTRATION
THIS PROGRAM WILL CREATE AND PRINT AN ARRAY
'BEGIN'

'INTEGER' 'ARRAY' 'A[1:5]';

'INTEGER' 'S, I';

S:=0;

'FOR' I:=1 'STEP' 1 'UNTIL' 5 'DO'

'BEGIN' 'A[I]:=3*(I-1);

S:=S+A[I];

10**

'END';

OUTPUT(61, '(' '(' 'SERIES OF ' ')',
' (' 'GEOMETRIC PROGRESSION ' ')',

' (' 'WITH R=3, A=1 ' ') ' ') ' ');

OUTPUT(61, '(' '// ' ') ' ');

'FOR' I:=1 'STEP' 1 'UNTIL' 5 'DO'

OUTPUT(61, '(' '+ZZZD' ') ', A[I]);

OUTPUT(61, '(' '// ' ') ' ');

OUTPUT(61, '(' '(' 'S = ' ') ', ZZZD' ') ', S);

'END'

20**

'EOP'

(01) LINE 0003 PROGRAM BEGINS
(01) LINE 0019 PROGRAM ENDS
(01) LINE 0019 SOURCE DECK ENDS

#LOAD,56

RUN

RUN

CHANNEL, 60=LU60, P80
CHANNEL, 61=LU61, P136, PP60
CHANNEL, END
CHANNEL, END

SERIES OF GEOMETRIC PROGRESSION WITH $R=3, A=1$

+1 +3 +9 +27 +81

S = 121

END OF ALGOL RUN

#

9. INPUT/OUTPUT AND FORMAT STRING

I/O Procedure Call

(a) Character transmission

```
INCHARACTER( CHANNEL,STRING, DESTINATION);  
OUTCHARACTER( CHANNEL,STRING, SOURCE);
```

STRING: Any character strings in the form of

'('('.....')')'

e.g. '('('ABCDE')')'

INCHARACTER: Reads a character from the channel, compares it to the character string until a match is found or upon exhaustion of the characters in the string.

Value of destination = J if a match is found at Jth character
0 if no match is found.

EXAMPLE:

```
INCHARACTER(60,'('('ABC')',I);
```

Input Character	Value of I
A	1
B	2
C	3
other	0

Informative error of type 01- non-format error will appear.
However, this is not an actual error.

OUTCHARACTER: Examines the value of source and writes out

- i) The corresponding character of the string if the value is in the range of 1-J (length of string)
- or ii) the error message "out character error" if the value is not in the range of 1-J.

EXAMPLE:

```
OUTCHARACTER (61, ('TUV'), J);
```

Value of J	Outputted character
2	T
3	U
4	V
other	"OUT CHARACTER ERROR"

Note comment on page 52

(b) Transmission of Arrays

```
INARRAY( CHANNEL, NAME OF ARRAY);
```

```
OUTARRAY( CHANNEL, NAME OF ARRAY);
```

- Reads or writes out a whole array by rows

EXAMPLE:

```
'ARRAY' X[1:2, 1:3];
```

```
INARRAY(60, X);
```

```
OUTARRAY(61, X);
```

X is a 2 x 3 array with the elements inputted or printed in the following order:

X₁₁, X₁₂, X₁₃, X₂₁, X₂₂, X₂₃

(c) Transmission of Type Real

```
INREAL( CHANNEL,VARIABLE NAME);  
OUTREAL( CHANNEL,VARIABLE NAME);
```

- Reads or writes out variables using standard format one at a time.

EXAMPLE:

```
'REAL'A;  
INREAL(60,A);  
OUTREAL(61,A);  
'COMMENT' INREAL(60,A,B) IS INVALID;
```

	INPUT	OUTPUT
A	1.01	+1.010000000'-000

(d) I/O Control Procedure Calls

EOF (channel, label): transfers control to the label when an end of file is encountered on an output device.

EXAMPLE: EOF (60,END);

(e) FORMAT I/O Call

```
INPUT( CHANNEL,FORMAT STRING,LIST OF VARIABLE NAMES);  
OUTPUT( CHANNEL,FORMAT STRING, LIST OF VARIABLE NAMES);
```

- Reads in or writes out one or more values according to specified format string.††

†† Legal format string and example, see section V.9

FORMAT STRING Summary of format codes

B Blank Space
D Digit (without zero suppression)
F Boolean true or false (variable must be declared 'BOOLEAN')
H INTEGER VARIABLE; 8 consecutive BCD characters are to be INPUT/OUTPUT to or from a single integer variable.
N Standard format (+D.9D'+3D)
S String character-used for output of string quantities
T Truncation
V Implied Decimal Point
Z Zero suppression
+ Print the sign
- Print the sign if it is minus
() Delimiters of replicated format string
/ New line
* New page
. Decimal point
'(')'' Inserted character string (can be used for title format)
' Exponent part indicator. E.G. 1.25'+002

EXAMPLE ON INPUT/OUTPUT AND FORMAT STRINGS

(1) NUMBER FORMATS

```
'REAL' A,B;
A:=-1000.999;
B:=21.126;
OUTPUT(61, '('2(+ZZDDD.DD4B)')',A,B);
RESULT: -1001.00 +021.13
OUTPUT(61, '('2(+3Z3D.2D4B)')',A,B);
RESULT: -1001.00 +021.13
OUTPUT(61, '('2(-3D2B3D.2DT4B)')',A,B);
RESULT: -001 000.99 -000 021.12
OUTPUT(61, '('2('INTEGER PART ')',-4ZV, '('FRACTION')',B3D,/))',A,B);
RESULT: INTEGER PART -1000, FRACTION 999
INTEGER PART 21, FRACTION 126
```

(2) STRING FORMAT

FORMAT CODE: S

During compilation, it will give message as:

(01) LINE xxxx NON-FORMAT STRING

EXAMPLES:

```
OUTPUT(61, '('3S')', '('YES')');
```

RESULT: YES

```
OUTPUT(61, '('2S')', '('YES')');
```

RESULT: YE

```
OUTPUT(61, '('5S')', '('YES')');
```

RESULT: YES')

```
OUTPUT(61, '('8S')', '('YES')');
```

RESULT: YES')'00

This is an Algol error

(3) BOOLEAN FORMAT

```
'BOOLEAN' A,B;
```

```
A:='TRUE';
```

```
B:='FALSE';
```

```
OUTPUT(61, '('F')', A);
```

RESULT: T

```
OUTPUT(61, '('FFF')', B);
```

RESULT: F

(4) ALIGNMENT MARKS

```
OUTPUT(61, '('/'')');
```

RESULT: (LINE RETURN)

```
OUTPUT(61, '('*')');
```

RESULT: (PAGE EJECT)

(5) TITLE FORMAT

```
OUTPUT(61, '('('ENTER DATA'))');  
RESULT: ENTER DATA  
OUTPUT(61, '('('STOP'))');  
RESULT: STOP
```

(6) STANDARD FORMAT

```
FORM: +D.9D'+3D
```

There are several ways of inputting and outputting with standard format

- (a) Read or write 1 variable
A:=123.457;

```
OUTPUT(61, '(')',A);
```

```
or OUTPUT(61, '('N)',A);
```

```
or OUTREAL(61,A);
```

Will give the same result as:

```
+1.234570000'+002
```

- (b) For 2 or more variables

```
OUTPUT(61, '(')',A,B); is recommended
```

However, one can still use format code N, for example:

```
OUTPUT(61, '('N,N)',A,B);
```

NOTE: '('N,N)' ≠ '('2N)' this will give
FORMAT STRING ERROR, but 2(N) is okay.

(7) H FORMAT

Read or write out 8 consecutive BCD characters (integer variable)

```

'INTEGER'I;
INPUT(60,('(H)'),I);
OUTPUT(61,('(H)'),I);
DATA= AB12HFZG (8BCD)
RESULT: AB12HFZG
DATA= XYZ1234AB (9BCD)
RESULT: XYZ1234A
DATA= 123556A (7BCD)
RESULT: 123556A

```

(8) FREE FORM INPUT AND STANDARD OUTPUT

FORMAT STRING '()''

It will read in values according to their types and write out values in the form of:

```

+D.9D' + 3D (real)
+15ZD (integer)

```

EXAMPLE

```
#ALGOL,I=FREEFORM,X,L
```

```

OS3 ALGOL  V0.0          FREEFORM          02/24/70          1147          PAGE 1
  FREEFORM
  'BEGIN'
    'INTEGER'A,B;
    'REAL'C,D;
    INPUT(60,('( '),A,B,C,D);
    OUTPUT(61,('( '),A,C);
    OUTPUT(61,('(//')');
    OUTPUT(61,('( ')',D,B);
  'END'
10**          'EOP'

```

(01) LINE 0002 PROGRAM BEGINS
(01) LINE 0009 PROGRAM ENDS
(01) LINE 0009 SOURCE DECK ENDS

#LOAD,56
RUN
RUN

CHANNEL,60=LU60,P80
CHANNEL,61=LU61,P136,PP60
CHANNEL,END
CHANNEL,END

1 -2456 13.44 -0.999

+1 +1.344000000'+001
-9.990000000'-001 -2456

END OF ALGOL RUN

#

Example on I/O and FORMAT String

#ALGOL,I=INOUT,X,L

OS3 ALGOL V0.0 INOUT 02/24/70 1150 PAGE 1

```
INOUT
ILLUSTRATION OF I/O STATEMENTS AND VARIOUS FORMAT STRINGS
'BEGIN'
  'INTEGER' I;
  'REAL' Y;
  'BOOLEAN' B;
  B:='TRUE';
  START: OUTPUT(61,('*'));
        OUTPUT(61,('(ENTER 1 TO READ AND WRITE)'));
10**    OUTPUT(61,('/'));
        OUTPUT(61,('(ENTER 2 TO PRINT LOGICAL VALUE)'));
        OUTPUT(61,('/'));
        OUTPUT(61,('(ENTER 3 OR LARGER TO STOP)'));
        OUTPUT(61,('/'));
        INPUT(60,(' '),I);
        OUTPUT(61,('//'));
        'IF' I=1 'THEN' 'GO TO' IN
        'ELSE' 'IF' I=2 'THEN' 'GOTO' PRINTB
        'ELSE' 'BEGIN'
20**    OUTPUT(61,('(STOP)'));
        'GO TO' FINISH;
        'END';
PRINTB: OUTPUT(61,('F'),B);
        'GO TO' START;
IN:    OUTPUT(61,('(ENTER Y)'));
        OUTPUT(61,('/'));
        INPUT(60,(' '),Y);
        OUTPUT(61,(' '),Y);
        'GO TO' START;
30**    FINISH: 'END'
        'EOP'
(01)   LINE 0003   PROGRAM BEGINS
(01)   LINE 0030   PROGRAM ENDS
(01)   LINE 0030   SOURCE DECK ENDS
```

#LOAD,56
RUN
RUN

CHANNEL, 60=LU60, P80
CHANNEL, 61=LU61, P136, PP60
CHANNEL, END
CHANNEL, END

ENTER 1 TO READ AND WRITE
ENTER 2 TO PRINT LOGICAL VALUE
ENTER 3 OR LARGER TO STOP
1

ENTER Y
1.567
+1.567000000' 000

ENTER 1 TO READ AND WRITE
ENTER 2 TO PRINT LOGICAL VALUE
ENTER 3 OR LARGER TO STOP
2

T

ENTER 1 TO READ AND WRITE
ENTER 2 TO PRINT LOGICAL VALUE
ENTER 3 OR LARGER TO STOP
3

STOP

END OF ALGOL RUN

#

10. SAMPLE RUN ON ALL STATEMENTS

Given a date, this program will find the day of the week on which that date occurred.

INPUT: 12/25/1965 means December 25, 1965
03/15/1970 means March 15, 1970

Compute J by the formula given below.

$$J = K + 2 * M + (3 * (M + 1) / 5) + N + (N / 4) - (N / 100) + (N / 400) + 2.$$

where

M = Month

K = Day

N = Year

Note: Treat January and February as the 13th and 14th months of the preceding year.

Compute L = remainder of J/7.

Then the day is Saturday if L = 0

Sunday if L = 1, etc.

OUTPUT:

THE DATE XX/XX/XXXX FALLS ON XXXXXXXXXX

ALGOL, I=STATE, X, L

OS3 ALGOL V0.0 STATEMEN 02/24/70 1156 PAGE 1

STATEMENTS - ILLUSTRATION ON STATEMENTS.
THIS PROGRAM IS TO FIND THE DAY OF THE WEEK.

```
'BEGIN'
  'INTEGER' M, D, Y, L;
  'SWITCH' DAY:=S1, S2, S3, S4, S5, S6, S7;
  'INTEGER' 'PROCEDURE' J(M, K, N);
    'VALUE' M, K, N;
    'INTEGER' M, K, N;
    'BEGIN'
      10**      J:=K+2*M+ENTIER(3*(M+1)/5)+N+ENTIER(N/4)
                -ENTIER(N/100)+ENTIER(N/400)+2;
    'END';
  'INTEGER' 'PROCEDURE' REM(I, J);
    'VALUE' J; 'INTEGER' I, J;
    'BEGIN'
      'INTEGER' REMM;
      REMM:=I;
      SUB: 'IF' REMM 'NOT LESS' J 'THEN'
          'BEGIN' REMM:=REMM-J;
          20**      'GOTO' SUB;
        'END';
      REM:=REMM;
    'END' REM;
  START: OUTPUT(61, '('('(' 'INPUT THE DATE' ')')')');
        OUTPUT(61, '('('/')')');
        INPUT(60, '('')', M, D, Y);
        EOF(60, END);
        'IF' M<3 'THEN' 'BEGIN' M:=M+12;
          30**      Y:=Y-1;
        'END';
        L:=REM(J(M, D, Y), 7);
        'IF' M>12 'THEN' 'BEGIN' M:=M-12; Y:=Y+1; 'END';
        OUTPUT(61, '('('(' 'THE DATE ' '), ZD, '('('/')', ZD,
          '('('/')', 4D, '(' ' FALLS ON ' ')')', M, D, Y);
        'GOTO' DAY [L+1];
        S1: OUTPUT(61, '('('(' 'SATURDAY' ')')')');
            'GOTO' L1;
        S2: OUTPUT(61, '('('(' 'SUNDAY' ')')')');
            'GOTO' L1;
        40** S3: OUTPUT(61, '('('(' 'MONDAY' ')')')');
            'GOTO' L1;
        S4: OUTPUT(61, '('('(' 'TUESDAY' ')')')');
            'GOTO' L1;
        S5: OUTPUT(61, '('('(' 'WEDNESDAY' ')')')');
            'GOTO' L1;
        S6: OUTPUT(61, '('('(' 'THURSDAY' ')')')');
            'GOTO' L1;
        S7: OUTPUT(61, '('('(' 'FRIDAY' ')')')');
        L1: OUTPUT(61, '('('/')')');
```

50** 'GOTO' START;
END: 'END'
'EOP'
(01) LINE 0003 PROGRAM BEGINS
(01) LINE 0051 PROGRAM ENDS
(01) LINE 0051 SOURCE DECK ENDS

#LOAD,56
RUN
RUN

CHANNEL,60=LU60,P80
CHANNEL,61=LU61,P136,PP60
CHANNEL,END
CHANNEL,END

INPUT THE DATE

01/02/1964
THE DATE 1/ 2/1964 FALLS ON THURSDAY

INPUT THE DATE

11/08/1947
THE DATE 11/ 8/1947 FALLS ON SATURDAY

INPUT THE DATE

12/25/1970
THE DATE 12/25/1970 FALLS ON FRIDAY

INPUT THE DATE

01/01/1970
THE DATE 1/ 1/1970 FALLS ON THURSDAY

INPUT THE DATE

PART THREE

Part Three contains several programs that are meant to serve as examples of ways in which ALGOL can be applied, not only in mathematical areas, but also in the simulation of models.

I. NEWTON'S METHOD

1. NEWTON'S METHOD FOR CUBE ROOTS

This program uses Newton's iteration method for locating the zeros of a function. The equations take the following form for cube roots:

$$f(x) = x^3 - a = 0$$

$$g(x) = x^3 - a + x = x$$

$$g(x) = x - \frac{f(x)}{f'(x)}$$

$$g(x) = x - \frac{x^3 - a}{3x^2} = \frac{2x^3 - a/x^2}{3}$$

The last equation is used in the program in the following manner:

$$y = x$$

$$x = (2y^3 + a/y^2) / 3.$$

The degree of accuracy can be controlled by the IF Statement on line 12.

EXAMPLE

1 { #TIME=10
#ALGOL, I=NEWTON, X, L

2 { OS 3 ALGOL VO.0 NEWTON 10/07/69
NEWTON
CUBE ROOT DETERMINATION
EXAMPLE 1 A

3 { 'BEGIN'
'REAL' A, APP, X, Y;
IN REAL(60,A);
IN REAL(60, APP);
X:=APP;
10** NEWTON:Y:=X;
X:=(2.0*Y+A/(Y²))/3.0 ;
'IF' (Y-X)/X < (.0001) 'THEN'
OUTPUT (61, '(' 4ZD.6D, / ')', X)
'ELSE' 'GOTO' NEWTON;
'END'
'EOP'

4 { (01) LINE 0005 PROGRAM BEGINS
(01) LINE 0015 PROGRAM ENDS
(01) LINE 0015 SOURCE DECK ENDS

5 { #LOAD,56
RUN
RUN

6
7
8

CHANNEL, 60=LU60, P80
CHANNEL, 61=LU61, P136, PP60
CHANNEL, END
CHANNEL, END

14 3.0 5.4

9

5.229322

10

END OF ALGOL RUN

#TIME
TIME 4.947 SECONDS MFBLKS 1 CFBLKS 1
#DATE
OCTOBER 7, 1969 1:45 PM

#

Comments to the example, Newton's Method for Cube Roots

1 The command "ALGOL,I=NEWTON,X,L" is used to call the ALGOL compiler. The "I" is used to specify where the program is coming from. In this case it is the saved file NEWTON. The "X" indicates where the object program should be sent. In this case it is 56, the standard unit that is assumed if none is stated. The "L" is used to specify that a listing of the program as it appears on NEWTON is desired.
NOTE: For other options see reference 7, pages 9, 10.

2 The listing of the program begins. Notice that the date is given and also the version, V0.0, the first eight letters of the program, the time, and the page. Anything that is stated before the first "BEGIN" is treated as a comment, and is not executed.

3 The executable part of the program begins. Under the OS-3 ALGOL certain words must be enclosed in single quotes, " ' ". See Part Two if you are in doubt about a word. Colons can be substituted for two periods: Semicolons can be substituted for a period and a comma.

 In this example, Newton is used as a label for the line "NEWTON:Y:=X;". Notice that later in the

program the command " 'GOTO' NEWTON" is used to return to that line.

For each "BEGIN" there must correspond an "END". Notice the "10**", this indicates line number ten.

4 If the program contains any errors they will be listed here. For a list of error messages and their explanations see reference 3. The beginning, end, and source deck ending is also listed.

5 The command "LOAD,56" was used to load the binary object program. The first "RUN" is typed by the user, and is terminated by a carriage return and a line feed. The second "RUN" is typed by the computer.

6 Two channel specifications are automatically supplied by the ALGOL system. If more channel specifications were desired they would be added at this point. They would have the form:

```
CHANNEL,AA=LUxx,Pbb,PPcc
```

where

AA	integer channel number, up to 14 decimal digits
xx	Operating System LUN (Logical Unit Number), 0-99 60 for input standard unit 61 for output standard unit 62 for card punch output
bb	Maximum physical record size in characters (normally 136 characters per line, or 68 for Teletype)
cc	Number of records per page

For more parameters see reference 8, page 5, Control Data Instant ALGOL, (this booklet can be purchased from the Computer Center main office).

Example: CHANNEL,40=LU40,P80,PP60

where LUN 40 would be equipped earlier by the control mode command

EQUIP,40=FILE

EQUIP,40=<Name of your or data file>,...,etc.

8 The data is typed in by the user. Data may be separated by two or more spaces or by one comma and one space, or data input format may be specified by using the command INPUT (see Part Two, page 7, 8).

9 The cube root of 5.229322 is typed by the computer.

10 The line, END OF ALGOL RUN, which is typed by the computer indicates a normal termination.

2. NEWTON'S METHOD WITH LOOPING

EXAMPLE

```

1 { #####
   #DATE
   OCTOBER 3, 1969  4:48 PM

2 { **SCOOP
   MAXTIME          1022
   SFBLKS           98
   SFBLKLIM        100

3 { #TIME=20
   #ALGOL, I=NEWTON2, X, L
  
```

```

OS3 ALGOL  V0.0          NEWTON2          10/03/69
NEWTON2
CUBE ROOT DETERMINATION, MORE THAN
ONE ROOT CAN BE CALCULATED PER RUN.
EXAMPLE 1 B
  
```

```

4 { 'BEGIN'
   'REAL' A, APP, Y, ENDTEST, X;
   OUTPUT(61, '(' '(' ENTER DATA ')',
   ,/ ')') );
5 { 10** LOOP: INPUT(60, '(' ')', A, APP);
   X..=APP;
   NEWTON..Y..=X;
   X..=(2.0 *Y +A/(Y^2))/3.0;
   'IF' (Y-X)/X < .001 'THEN'
   OUTPUT(61, '(' '(' THE CUBE OF ')',
   , 3ZD.3DBBB, '(' IS ')',
   3ZD.9DBB, // ')', A, X)
   'ELSE' 'GOTO' NEWTON;
6 { 20** INREAL (60, ENDTEST);
   'COMMENT' THIS WILL BE A POSITIVE
   OR NEGATIVE VALUE;
   'IF' ENDTEST < 0 'THEN' 'GOTO'
   LOOP 'ELSE' 'GOTO' FINISH;
   FINISH..'END'
  
```

```

'EOP'
(01) LINE 0006 PROGRAM BEGINS
(01) LINE 0025 PROGRAM ENDS
(01) LINE 0026 SOURCE DECK ENDS
  
```

```

#LOAD,56
RUN
RUN
  
```


CHANNEL, 60=LU60, P80
CHANNEL, 61=LU61, P136, PP60
CHANNEL, END
CHANNEL, END

ENTER DATA
143 5.4 -1 34.00 3.5 1
THE CUBE OF 143.000 IS 5.229321532
THE CUBE OF 34.000 IS 3.239611805

END OF ALGOL RUN

#LOGOFF
TIME 4.700 SECONDS MFBLKS 1 COST \$0.52

Comments to the example, Newton's Method with Looping

- 1 #DATE This command is used to print the date and time on the printed output.
- 2 #*SCOOP This tells the user how much time (seconds) he has left under his number, how many saved file blocks he is using and the maximum number he can use.
- 3 #TIME=20 Time consumption (seconds) is limited to 20.
- 4 OUTPUT(61... This statement is used to print out ENTER DATA. Characters that appear within the format string enclosed in another set of '(' and ')' will be printed with the output.
- 5 LOOP: The label loop is set up for the purpose of reading more input after one calculation has been made. See lines 24 and 25.
- 6 INREAL(60,ENDTEST) If endtest is positive the program will go through the normal termination (i.e., END OF ALGOL RUN). If endtest is negative, another computation will be performed.

II. TWO-DIMENSIONAL ARRAY

This program declares a series of arrays of ever-increasing dimensions.

EXAMPLE

```

#####
#DATE
OCTOBER 7, 1969  1:48 PM

1 { #TIME=10
    #ALGOL, I=ALGOLDIM, X, L

OS 3 ALGOL  V0.0      2-DIMENS      10/07/69      1349
      2-DIMENSIONAL ARRAY
2 { THIS PROGRAM DECLARES A SERIES OF ARRAYS
    OF EVER-INCREASING DIMENSION.  THE ARRAY IS
    THEN FILLED WITH COMPUTED VALUES, ONE OF
    WHICH IS ALTERED.  THE ALTERED VALUE IS
    THEN SEARCHED FOR AND PRINTED.
    SINCE THE PROGRAM HALTS NORMALLY WHEN THE
    DECLARED ARRAY SIZE EXCEEDS THE AVAILABLE
    MEMORY IT IS NECESSARY TO LIMIT THE
    TIME.
3 { 10**
    'BEGIN'
    'INTEGER' I, M, N;
    I:=10;
    L:I:=I+1;
    OUTPUT(61, '(' /, 3D ')', I);

4 { 20**
    'BEGIN'
    'ARRAY' A[-3*I:-1, I:2*I];
    'INTEGER' P, Q;
    'FOR' P:= -3*I 'STEP' 1 'UNTIL' -1
    'DO' 'FOR' Q:= I 'STEP' 1 'UNTIL'
5 { 2*I 'DO' A[P, Q]:= -P+100*Q;
    M:= -2*I;
    N:= I+2;
    A[M, N]:= A[M, N] + 10000;
    'FOR' P:= -3*I 'STEP' 1 'UNTIL' -1
    'DO' 'FOR' Q:= I 'STEP' 1 'UNTIL'
    2*I 'DO' 'IF' A[P, Q] 'NOT EQUAL' 100*Q-P
    'THEN'
30**

    'BEGIN'
    OUTPUT(61, '(' /, 5D ')', A[P, Q] )
    'END';

```

'GOTO' L
'END'

'END';
40** 'EOP'
(01) LINE 0012 PROGRAM BEGINS
(01) LINE 0039 PROGRAM ENDS
(01) LINE 0039 SOURCE DECK ENDS

#LOAD,56
RUN
RUN

CHANNEL,60=LU60,P80
CHANNEL,61=LU61,P136,PP60
CHANNEL,END
CHANNEL,END

011
11322
012
11424
013
11526
014
11628
015
11730
016
11832
017
11934
018
12036
019
12138
020
12240
021
12342

TIME CUT
#TIME
TIME 9.912 SECONDS MFBLKS 4 CFBLKS 2
#TIME=12
#GO
022
12444
023
12546
024
12648

TIME CUT
#LOGOFF
TIME 11.956 SECONDS MFBLKS 4 COST \$1.15

Comments to Example II, Two-Dimensional Array

1 In this program, it is necessary to set a time limit, since the program will run until memory overflow. A time limit of 60 seconds is automatically set. However, it is still good procedure to set a smaller time limit if you are not sure of a program or if you know the program does not have a normal exit.

 When you do get a time cut, reset time=old value+10 etc., and then type GO.

 This program was saved under the file name ALGOLDIM.

2 In this example the comment statements are used to explain the purpose of the program. The name of the program is taken from the first eight characters of the first line. Before loading this program, or a program of this type, it is a good idea to set a time limit. This is done from the control mode, "#", by stating "TIME=(NUMBER OF SEC)".

3 The executable program begins on line 12. Notice the "10**" which indicates line number 10.

4 The "/" also represents the left bracket "[" and they can be interchanged. The same is true for the right bracket.

 The symbols '(', ')' used in the format are

delimiters and must be used to enclose the format string under the OS-3 ALGOL. The symbol / causes a line feed. The code 3D indicates a 3-digit output for integers. The I indicates the variable that is to be printed.

- 5 The values for M and N are set and the value of the M and Nth element of the Array is set.

III. ECONOMICS EQUILIBRIUM PROBLEM

This program uses a model that is made to resemble an economic system that is controlled by $NNP = \text{Consumption} + \text{Investment}$, with an assumed multiplier working on Investment. Government and Taxes are excluded from this simplified model.

The assumption is made that the public will be slow to react to the increased Investment. Therefore, a time lag is introduced in line 19.

EXAMPLE

#####

#DATE
OCTOBER 7, 1969 1:55 PM

#TIME=10

#ALGOL, I=ALECON, X, L

```
05 3 ALGOL  V0.0          ALECON          10/07/69      1355
1 {   ALECON
    ECONOMICS PROBLEM IN EQUILIBRIUM.
    AI STANDS FOR INVESTMENT DEMAND
    C STANDS FOR CONSUMPTION DEMAND
    ANNP STANDS FOR THE VALUE OF NNP
    ANNP L STANDS FOR THE VALUE OF THE LAST
2 {   NNP.
    THIS PROGRAM CALCULATES THE CHANGE IN NNP CAUSED BY AN
    INCREASE OF 10 BILLION IN INVESTMENT, ACCORDING TO THE
    10** MULTIPLIER THEORY.
3 {   'BEGIN'  'REAL' AI, C, ANNP, ANNP L, DF;
4 {     AI:=60.0;
5 {     C:=740.0;
6 {     ANNP:=800.0;
7 {     ECON..OUTPUT(61, ('3(.10D),/'), C, ANNP, AI);
    ANNP L:=ANNP;
    AI:=70.0;
    C:=206.667 + (2.0/3.0)* ANNP L ;
    20** ANNP:= C+ AI;
    DF:= .001 -(ABS(ANNP-ANNP L));
    'IF' DF < 0.0 'THEN' 'GOTO' ECON 'ELSE'
    'GOTO' LAST;
    8 { LAST..OUTPUT(61, ('3(.5D),/'), C, ANNP, AI, ANNP L);
    'END'
    'EOP'
    (01) LINE 0012  PROGRAM BEGINS
    (01) LINE 0025  PROGRAM ENDS
    (01) LINE 0025  SOURCE DECK ENDS
```

#LOAD,56
RUN
RUN

```
CHANNEL, 60=LU60, P80
CHANNEL, 61=LU61, P136, PP60
CHANNEL, END
CHANNEL, END
```

```
9 {
**+7.400000000'+002**+8.000000000'+002**+6.000000000'+001*
**+7.400003334'+002**+8.100003334'+002**+7.000000000'+001*
**+7.466672223'+002**+8.166672223'+002**+7.000000000'+001*
**+7.511118148'+002**+8.200
#
```

```
#EDIT
```

```
IFIN,ALEN\CON
```

```
IRESEQ
```

```
10 {
JSARL,,,/.10D/,/3ZD.10D/
00016: ECON..OUTPUT(61,('(3(3ZD.10D),/)',C,ANNP,AI));
JOUT,ALECON
```

```
#ALGOL, I=ALECON, X
```

```
OS3 ALGOL  V0.0          ALECON          10/07/69
(01)  LINE 0012        PROGRAM BEGINS
(01)  LINE 0025        PROGRAM ENDS
(01)  LINE 0025        SOURCE DECK ENDS
```

```
#LOAD,56
RUN
```

```
TIME CUT
#TIME
TIME 9.910 SECONDS  MFBLKS 1  CFBLKS 1
#TIME=20
#GO
RUN
```

CHANNEL, 60=LU60, P80
CHANNEL, 61=LU61, P136, PP60
CHANNEL, END
CHANNEL, END

740.0000000000	800.0000000000	60.0000000000
740.0003333000	810.0003333000	70.0000000000
746.6672222000	816.6672222000	70.0000000000
751.1118148000	821.1118148000	70.0000000000
754.0748765000	824.0748765000	70.0000000000
756.0502510000	826.0502510000	70.0000000000
757.3671674000	827.3671674000	70.0000000000
758.2451116000	828.2451116000	70.0000000000
758.8304077000	828.8304077000	70.0000000000
759.2206051000	829.2206051000	70.0000000000
759.4807368000	829.4807368000	70.0000000000
759.6541578000	829.6541578000	70.0000000000
759.7697719000	829.7697719000	70.0000000000
759.8468479000	829.8468479000	70.0000000000
759.8982319000	829.8982320000	70.0000000000
759.9324880000	829.9324880000	70.0000000000
759.9553253000	829.9553253000	70.0000000000
759.9705502000	829.9705502000	70.0000000000
759.9807001000	829.9807001000	70.0000000000
759.9874668000	829.9874668000	70.0000000000
759.9919778000	829.9919778000	70.0000000000
759.9949852000	829.9949852000	70.0000000000
759.9969901000	829.9969902000	70.0000000000
759.9983268000	829.9983268000	70.0000000000
759.99922	829.99922	70.00000
+8.299983268'+002		

END OF ALGOL RUN

#LOGOFF

TIME 16.700 SECONDS MFBLKS 1 COST \$1.79

Comments to Example III, Economics Equilibrium Problem

1 The first six characters of the beginning
of an ALGOL program are used for the name, in this
case ALECON.

2 The characters that occur between the name and
the first "BEGIN" are ignored by the computer. They
are usually used for comments.

3 The first "BEGIN" appears and immediately fol-
lowing it is a declaration, in this case "REAL".
This gives the variables AI,...,DF real storage loca-
tions (decimal).

4 The values of the variables are set.

5 The label ECON is set up. This will be used for
reference later in the program. See line 00022,
'GOTO' ECON.

6 The last value of NNP is saved before the next
is calculated by giving it the name ANNPL.

7 The Investment (AI) is increased by +10 billion.
This causes disequilibrium of the economic model.
Notice that in this simplified model, Government and
Taxes are absent.

8 DF is a test value that will have a positive
value when the economic model is in equilibrium.

9 Output statement allows format specifications.
See Part Two. The format that was used was not
readily readable, so we will go back to EDIT and
change the format.

10 The format is changed, and the changed program
saved.

IV. SIMPSON'S RULE FOR INTEGRATION

This program gives an approximation to the integral of a specified function. The program will calculate only a positive integral.

The interval of integration is subdivided into an even number of intervals.

The number of intervals is n .

The step width is $J=(B-A)/n$, where B is the upper limit and A is the lower limit of integration

$$X_k = X_0 + KJ \quad (K=1, 2, \dots, n)$$

$$\int_A^B f(x) dx = \frac{J}{3} [f(x_0) + 4f(x_1) + 2f(x_2) + \dots + 4f(x_{n-1}) + f(x_n)] + \text{Rem.}$$

Example IV

#TIME=20
#ALGOL,I=ALSIMIN,X,L

OS3 ALGOL V0.0 ALSIMIN 10/07/69 1409

ALSIMIN

THIS PROGRAM GIVES AN APPROXIMATION FOR THE
INTEGRAL OF F(I) , WHICH IN THIS EXAMPLE IS:

1

$F(I)=SQRT(2.0/3.1416)*1.0/2.7183*((-X^2.0)/2.0).$

THIS EXAMPLE IS SET UP TO HANDLE ONLY POSITIVE
UPPER AND LOWER LIMITS.

10**

A STANDS FOR THE LOWER LIMIT OF THE INTEGRAL.
B STANDS FOR THE UPPER LIMIT OF THE INTEGRAL.
N STANDS FOR THE NUMBER OF INTERVALS.

THIS EXAMPLE USES SIMPSON'S RULE FOR INTEGRATION.

2

'BEGIN' 'REAL' A,B,N,X,J,R,SUM, INTEGRAL;

3

'INTEGER' K,I;

'ARRAY' F[1:100];

4

20**

INPUT(60, '(' 3(2D.3DBB),/ ')',A,B,N);

'IF' N < 0.0 'THEN' 'GOTO' TWO 'ELSE' 'GOTO'
START;

START..X:=A;

R:=SQRT(2.0/3.1416);

J:=(B-A)/N;

'FOR' I:=1 'STEP' 1 'UNTIL' 100 'DO'

5

'BEGIN' F[I]:=R*2.7183*((-X^2)/2.0);

'IF' X < B 'THEN' 'GOTO' NEXT

'ELSE' 'GOTO' TWO;

NEXT..X:=X + J;

'END';

30**

TWO..K:=1;

SUM:=F[K];

ONE..K:=K+ 1 ;

SUM:=SUM+ 4.0 * F[K] ;

K:=K + 1 ;

SUM:= SUM+ 2.0 * F[K] ;

'IF' ((K+1)-I) < 0.0 'THEN' 'GOTO'

ONE 'ELSE'

K:=K + 1;

6

40**

SUM:=SUM + F(K);
INTEGRAL := (J/3.0)*SUM;

OUTPUT(61, '(' //, '(' THE STEPWIDTH ')',
2D.4D,/, '(' THE UPPER AND LOWER L ')',
,2(3D.4DBBBBB),/, '(' THE INTEGRAL VALUE ')',
BBBB3D.4D,/ ')', J, B, A, INTEGRAL);

'END'

'EOP'

(01) LINE 0014 PROGRAM BEGINS
(01) LINE 0047 PROGRAM ENDS
(01) LINE 0047 SOURCE DECK ENDS

#LOAD,56
RUN
RUN

CHANNEL,60=LU60,P80
CHANNEL,61=LU61,P136,PP60

CHANNEL,END
CHANNEL,END

7 {
 ••
 #GO
 01.000 02.000 10.000

THE STEPWIDTH 00.1000
THE UPPER AND LOWER L 002.0000 001.0000
THE INTEGRAL VALUE 000.2754

END OF ALGOL RUN

#LOGOFF
TIME 8.400 SECONDS MFBLKS 4 COST \$1.50

Comments to Example IV, Integration Approximation

- 1 A blank line is ignored.

- 2 ALGOL uses free-spacing (i.e., there are no specified columns in which instructions must fall with the one exception of "EOP" 10-14).

- 3 The variable F is declared to be an array having 100 locations, F[1], F[2], F[3], etc.

- 4 An "IF" statement is used to test the value of N. If N is negative the program jumps to line 31, where the label TWO is found. If N is positive the program goes to the line where the label START is found, line 21.

- 5 A "FOR" statement is used to set up a loop around a "BEGIN", "END". The "IF" statement tests to see if the upper limit of the integral has been reached. If it hasn't been reached the program goes to line 29 where the label NEXT is found.

- 6 These lines are a statement of Simpson's Rule for Integration: $I = J/3 [F(K) + 4F(L) + 2F(M) + \dots F(P)]$.

- 7 The usage of the break key was made here when the typing error, ".", occurred. The break key was pressed before the carriage return so the "." was not sent to the computer. The OS-3 control mode command GO causes the return back to the program.

REFERENCES

1. An Introduction to ALGOL 60, by C. Anderson, Mass., Addison-Wesley Co., Inc., 1964.
2. Computer System ALGOL, reference manual, Control Data Corporation, 1966.
3. ALGOL Generic Reference Manual, CDC, 3000,6000; 1968.
4. OS-3: A User's Manual with Examples, OSU Computer Center, Baughman, Berryman, Pielstick, cc-69-24, September, 1969.
5. Oregon State University Computer Center, OS-3 User's Manual, G. Bachelor, cc-68-3, March, 1968.
6. OS-3 Editor Manual, Dayton, ccm-70-7, January, 1970.
7. OS-3 Reference Manual, Skinner, ccm-70-8, January, 1970.
8. Control Data Instant ALGOL, 3100/3200/3300/3500, Control Data Corporation, 1966.

INDEX

- *ALGLIB;10,11
- ALGOL;2
- Arithmetic Expression
 - Operators;35
 - Precedence of Operators;39
- 'ARRAY' Declarations;24
- Assignment Statements;43

- Batch;2,13-15
- 'BEGIN';19
- Binary Program;10
- Blanks-Format;55
- Block
 - Example;41,42
 - Structure;21
 - Syntax;40
- Boolean
 - Declarations;21,22
 - Expressions;38
 - Format;56
- Bounds-Array
 - Upper & Lower;24

- Card Usage;13-17
- Carriage Return Key on
 - Remote Terminal;4
- CDC 3500;preface
- CHANNEL END;8
- Channels;8,69
- 'COMMENT';19
- Compound Statements;40
- Conditional Statements;44
- CONTROL A Key;4
- Control Statements on
 - Remote Terminal;4,6,7

- Decimal Point-Implied;55
- Declarations;21
- Delimiters;6
- Designational Expressions;39
- Diagnostic-Example;6
- Dummy Statement;43,44

- EDIT Mode;3
- Elements of Single Program;19
- 'END';19
- 'EOP';19
- Error Message-Example;6

- Expressions
 - Arithmetic;38
 - Designational;38,39

- Format
 - Alignment;56
 - Boolean;56
 - Codes;55
 - Free Form;58
 - Number;55
 - Standard;57,58
 - String;52,54,55
 - Title;57
- FOR Statement;48
- Function Designators;38
- Functions (Standard);39

- GO TO Statements;43

- 'IF';44-46
- INARRAY;24,53
- INCHARACTER;52
- INPUT-Editor;4,5
- INPUT-OUTPUT;52
- INREAL;
- 'INTEGER' Declaration;21

- LABELS
- LOGICAL Values;

- Operators
 - Arithmetic;35
 - Boolean;35
 - Precedence;39
 - Relational;35
- OS-3;preface
- OUTARRAY;53
- OUTCHARACTER;53
- OUT-Editor;5
- OUTPUT;24
- OUTREAL;
- 'OWN' Declarations;21,22

- Paper Tape;12
- Procedure
 - Calling;29,30
 - Declaration;21,28
- Program;1

Program Diagram of
Source Deck Structure;18
Programs
Simple Programs;6-12,20
Boolean;36,37
Cube Roots of Complex
Numbers;26
Day of Week;62-64
Economics Equilibrium;79-84
Examples with I/O;60-61
Geometric Progression;50,51
Newton's Method for
Cube Roots;65-73
Roots of Quadratic;31-34
Simpson's Rule for
Integration;85-88
Two-Dimensional Array;74-78

'REAL' Declarations;21
Remote Terminal-Teletype;2,3

SARL-Editor;7
Statements
Assignment;43
Conditional;44
Dummy;43,44
FOR;48
GO TO;43
Procedure;21
'STEP';48
'SWITCH'
Declaration;21
Sample Program;25,26

Teletype;2,3
TIMECUT;76,77
TTP-Editor;12
TYPE Declarations;21

'WHILE';48

Zero Suppression;55