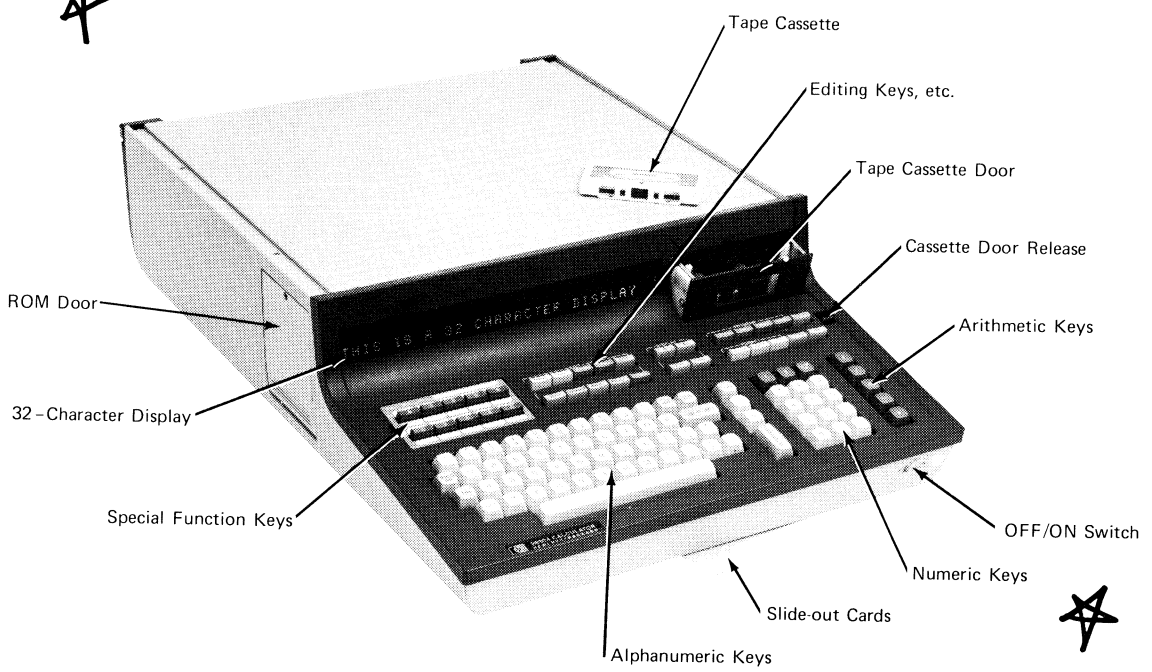
Model 30


30 MATRIX OPERATIONS
30 STRING VARIABLES
30 PLOTTER CONTROL
9830A CALCULATOR

Model 9830A System



* Powerful computer-like language 'Basic' - easy to use.
* Operator Friendly - typewriter keyboard - Alpha Numeric Display.
* Expandability - from 2K to 8K
* Special Functions - user definable (standard)
* Additional Capabilities - Strings. Matrices etc.
* Terminal Capability - Terminal I and Datacom

Features Galore
And Many More!

MEMORY

* **BASIC CALCULATOR** - IS FITTED WITH 4K BYTES (2K WORD) OF MEMORY.

- OPTIONS :-

  275 FACTORY INSTALLED 8K BYTES (4K WORDS)

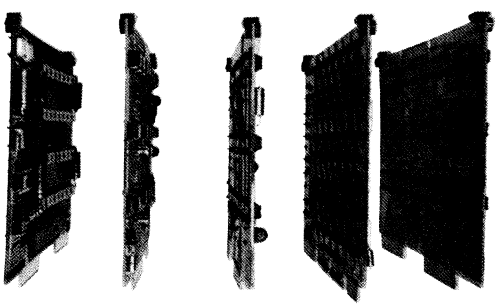
  276 - " - - " - 16K BYTES (8K WORDS)

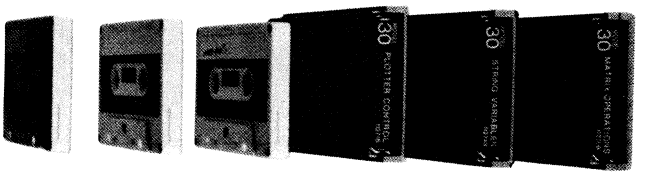
  11281A FIELD INSTALLED 8K BYTES

* **READ ONLY MEMORIES.**

- 11270B MATRIX ROM (11270F) *
- 11271B PLOTTER ROM (11271F) *
- 11272B EXTEND I/O ROM (11272F) *
- 11274B STRING VARIABLES ROM (11274F) *
- 11277B TERMINAL I ROM (11277F) *
- 11278B BATCH BASIC ROM
- 11279B ADVANCED PROGRAMMING ROM

* DENOTES FIELD INSTALLED.
THESE ROMS CAN ALSO BE INTERNAL FACTORY INSTALLED (MAX. 3)





GENERAL INFORMATION

* KEYBOARD

- ALPHANUMERIC KEYS SIMILAR TO A TYPEWRITER - LETTERS ALWAYS APPEAR IN UPPER CASE.
- NUMERIC KEYS - STANDARD CALC. TYPE BLOCK.
- ARITHMETIC KEYS - STANDARD KEYS
  + − / * ↑
- SPECIAL FUNCTION KEYS f0-f9 PROVIDE 20 USER DEFINABLE ACTIONS
- OTHERS - EDITING, DISPLAY, FORMAT, TAPE CASSETTE ETC.

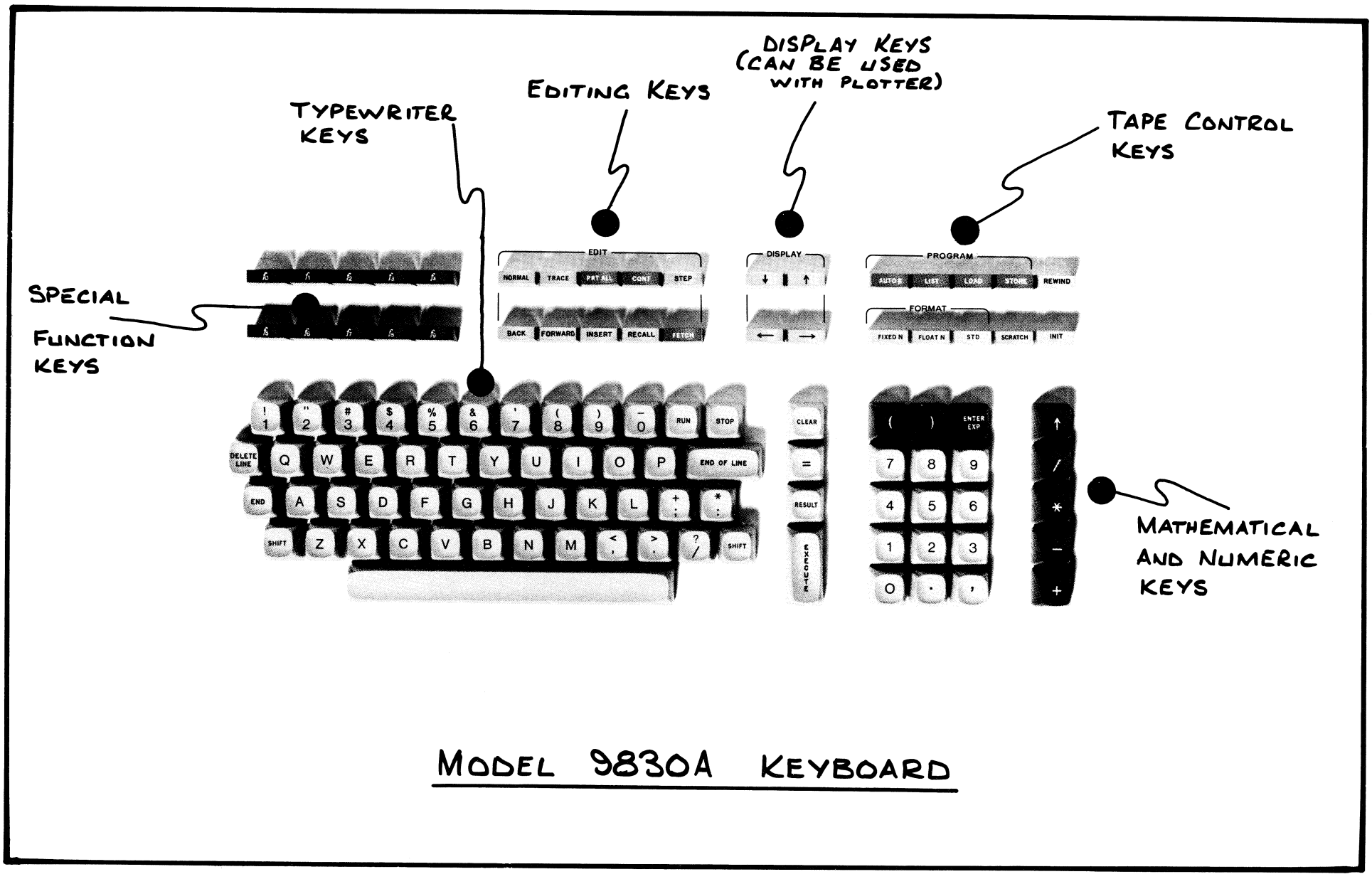
* REPETITIONNNNNNNNN - MOST KEYS IF HELD DOWN PRODUCE REPEATED ACTION.

* FREE FORMAT - IN ALL CASES EXCEPT PRINT, DISP, TYPE OF SITUATION SPACES ARE IGNORED e.g.

  4+1 = 4 + 1

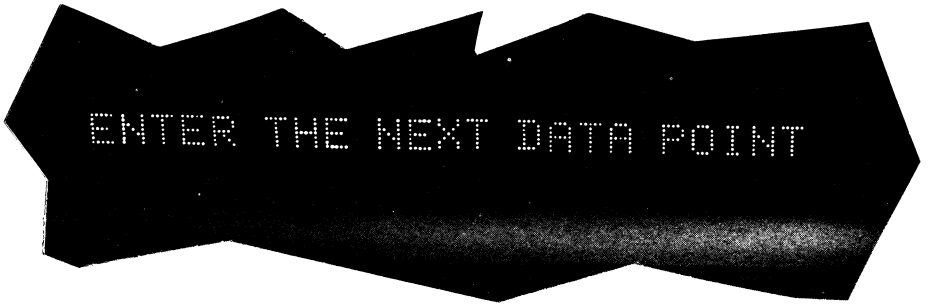
* CONTACTLESS KEYBOARDS

  ALL 9800 SERIES CALCULATORS ARE FITTED WITH CONTACTLESS KEYBOARDS USING PRINTED CIRCUIT TRANSFORMERS.



MODEL 9830A KEYBOARD

DISPLAY

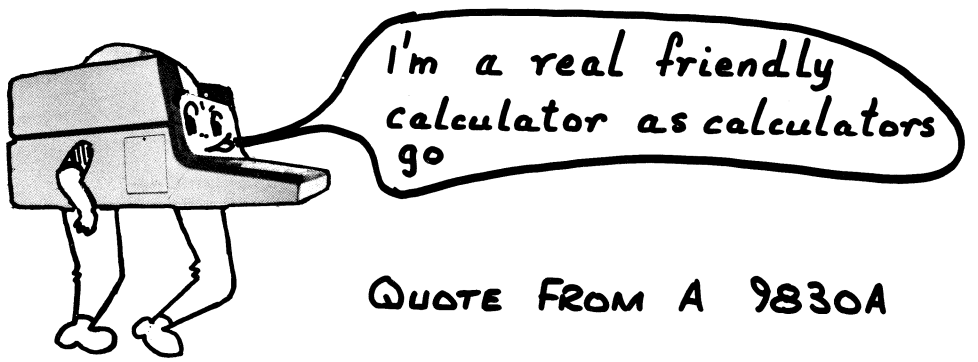


* FULLY ALPHA-NUMERIC

- 32 CHARACTER ALPHA/NUMERIC LED DISPLAY
- FULL AUTOMATIC SCROLLING OF AN 80 CHARACTER LINE WITH THE AP1 ROM.

* COMMUNICATION BETWEEN CALCULATOR-OPERATOR-CALCULATOR, POWERFUL FEATURE. USE IT IN YOUR DEMOS.

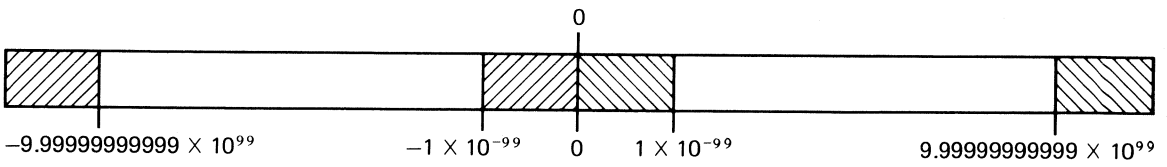
- USED IN THE QUESTION-ANSWER SITUATION OPERATORS DO NOT NEED EXTENSIVE MACHINE LANGUAGE.



QUOTE FROM A 9830A

* **LINE LENGTH** – MAX. 80 CHARACTERS
  - 32 CHARACTERS DISPLAYED
  - AN AUDIBLE TONE AFTER 72 CHAR.

* **RANGE** – • $-9.99999999999 \times 10^{99}$ TO $9.99999999999 \times 10^{99}$ e.g.



* **MEMORY** – THE USERS MEMORY IS EXPRESSED IN WORDS:

1 WORD = 16 BITS = 2 BYTES

* **ERROR MESSAGES** – INDICATES A FAULT BY A NUMBER REFERRING TO ERROR LIST UNDER CALCULATOR.

THREE IMPORTANT KEYS

CLEAR : ERASES DISPLAY

EXECUTE: PERFORMS THE OPERATION IN THE DISPLAY.

END OF LINE: PUTS PROGRAM LINES IN MEMORY

LET'S USE THE KEYBOARD!

KEYBOARD OPERATION

* FIXED, FLOATING, STANDARD

- FIXED N – FIXES NUMBER OF SIGNIFICANT DIGITS
- FLOAT N – FIXES NUMBER OF SIGNIFICANT DIGITS AND DISPLAYS IN FLOATING POINT WITH EXPONENT.
- STD – WHEN CALCULATOR IS TURNED ON RESULTS ARE IN STD. FORM AND APPEAR AS WOULD BE EXPECTED.

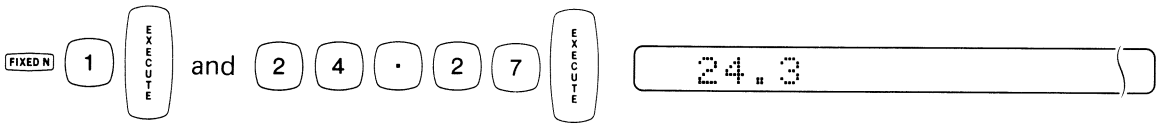
EXAMPLES:

		Standard	Fixed '2'	Float '4'
a)	23	23	23.00	2.3000E+01
b)	−173.0	−173	−173.00	−1.7300E+02
c)	.068	0.068	0.07	6.8000E−02
d)	12345.6	12345.6	12345.60	1.2346E+04
e)	.003	3.00000E−03	0.00	3.0000E−03
f)	7.314	7.314	7.31	7.3140E+00

- THESE INSTRUCTIONS ARE PROGRAMMABLE.
- THE CALCULATOR ALWAYS WORKS WITH 12 POINT ACCURACY - THESE INSTRUCTIONS AFFECT ONLY THE OUTPUT.

* ROUNDING

e.g.

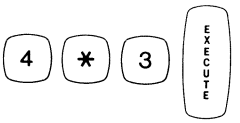


ARITHMETIC

* CALCULATING

- BASIC CALCULATIONS CARRIED OUT BY KEYING REQUIRED FUNCTIONS INTO DISPLAY AND PRESSING EXECUTE.

e.g.



The EXECUTE key must be pressed to evaluate any keyed-in expression.

* ARITHMETIC HIERACHY

- HIERACHY IS THE ORDER OF EXECUTION OF THE MATHEMATICAL OPERATIONS.
- WHEN TWO OPERATORS HAVING THE SAME LEVEL OCCUR EXECUTION IS LEFT TO RIGHT.
- THE USE OF PARENTHESIS WILL CHANGE THE ORDER OF EXECUTION.

TOTAL MATHEMATICAL HIERARCHY

Functions	highest precedence
↑	
NOT	
* /	
+ –	
Relational Operators	
AND	
OR	lowest precedence

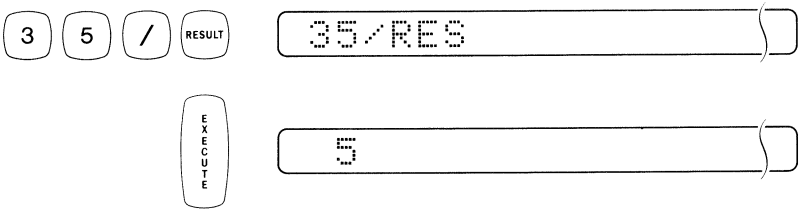
NOTE! ALL OPERATIONS IN PARENTHESIS ARE EXECUTED FIRST!

* RESULT

RESULT

- WHEN PRESSED, WILL DISPLAY THE RESULT OF THE LAST ARITHMETIC STATEMENT EXECUTED
- CAN BE USED IN A FOLLOWING EXPRESSION - SO INCLUDING THE PREVIOUS RESULT.

e.g. IF THE PREVIOUS RESULT = 7



* VARIABLES

- SIMPLE VARIABLES – ANY LETTER FROM A TO Z - ANY LETTER FOLLOWED BY A DIGIT 0 TO 9
- ARRAY VARIABLES – ONE DIMENSION, LETTER A TO Z : A(3) = A(1), A(2), A(3) (VECTOR ARRAY) - TWO DIMENSION LETTER A TO Z : A(2,2) = A(1,2) A(1,1), A(2,1) A(2,2) (MATRIX ARRAY)
- STRING VARIABLE (WITH STRING ROM) A COLLECTION OF ALPHANUMERICS WITHIN QUOTATION MARKS :

  A$ = "HEWLETT-PACKARD"

ERROR 40 ! REMEMBER A VARIABLE MUST BE DEFINED BEFORE BEING USED.

FUNCTIONS

* MATHEMATICAL FUNCTIONS

- ABS : DETERMINES THE ABSOLUTE VALUE OF AN EXPRESSION.
- EXP : RAISES THE CONSTANT 'e' TO THE POWER OF AN EXPRESSION.
- INT : GIVES THE INTEGER VALUE OF THE EXPRESSION.
- LGT : DETERMINES THE LOG TO BASE 10 OF A POSITIVE VALUE, EXPRESSION.
- LOG : AS LGT BUT TO BASE 'e'
- RND : GIVES A RANDOM NUMBER BETWEEN 0 AND 1
- SGN : RETURNS : 1 IF EXPRESSION > 0
  : 0 " " " = 0
  : -1 " " " < 0
- SQR : COMPUTES THE SQUARE ROOT OF A POSITIVE EXPRESSION.

EXAMPLES :

A B S (2 − 7)

```
5
```

which gives a positive value to the expression.

L G T 2

```
0.301029996
```

L O G 2

```
0.693147181
```

S G N (6 * 7)

```
1
```

* Trigonometrical Functions

- SIN : DETRMINES THE SINE OF EXPRESSION
- COS : DETERMINES THE COSINE OF EXPRESSION
- TAN : DETERMINES THE TANGENT OF EXPRESSION
- ATN : DETERMINES THE ARCTANGENT OF EXPRESS.

* Angles assumed to be Radians unless otherwise stated :-

- DEG : WHICH CALCULATES ANGLES IN DEGREES
- GRAD : CALCULATES ANGLES IN GRADS.
- RAD : CALCULATES ANGLES IN RADIANS

* π

THE VALUE FOR π IS OBTAINED BY KEYING IN [P] [I] AND WILL GIVE A VALUE OF 3.141592653G0

Example

[D] [E] [G]	⊢
[C] [O] [S] [(] [−] [1] [2] [0] [)]	−0.500000000
[A] [T] [N] [1]	45
[A] [T] [N] [S] [I] [N] [9] [0]	45

* LOGICAL EVALUATION

- EXPRESSIONS CAN BE COMPARED AND A '1' IS RETURNED FOR 'TRUE' AND A '0' FOR FALSE COMPARISON

* RELATIONAL OPERATORS

TO DETERMINE THE LOGICAL RELATIONSHIP BETWEEN TWO EXPRESSIONS.

equality	=
inequality	# or <>
greater than	>
less than	<
greater than or equal to	>=
less than or equal to	<=

EXAMPLE

IF WE DEFINE A = 1 . B=2 . C=3 . D=3

THEN THE FOLLOWING IS TRUE :

A<B	which is true, so 1 is displayed.
B<A	which is false, so Ø is displayed.
B#C	which is true, so 1 is displayed.
C#D	which is false, so Ø is displayed.
3=C	which is true, so 1 is displayed.
4=A	which is false, so Ø is displayed.
A=4	which is an assignment statement and not a logical statement; so A is assigned the value of 4.

RELATIONAL OPERATORS ARE USED EXTENSIVELY IN THE IF--- THEN STATEMENTS.

* LOGICAL OPERATORS

- THREE LOGICAL OPERATORS CAN BE USED AND ALSO COMBINED WITH RELATIONAL OPERATORS.

AND : COMPARES TWO EXPRESSIONS BOTH MUST BE TRUE FOR A TRUE RESULT

OR : USING 'OR' THEN IF EITHER OF THE EXPRESSIONS ARE TRUE THE RESULT IS TRUE. IF BOTH ARE FALSE THE RESULT IS FALSE.

NOT : CHANGES THE LOGICAL RESULT FROM TRUE TO FALSE IF ORIGINALLY TRUE
FALSE TO TRUE " —— " —— FALSE

EXAMPLE

IF A=0. B=2. C=4. D=4

THEN :-

A OR B

The arithmetic value of A is zero (false) while the arithmetic value of B is two (true). Since at least one of the expressions is true, a 1 is displayed.

A OR C–D

Both arithmetic expressions have a value of Ø (false). So a Ø is displayed.

NOT A

Since A is false, NOT A is true and a 1 is displayed.

NOT B OR NOT C

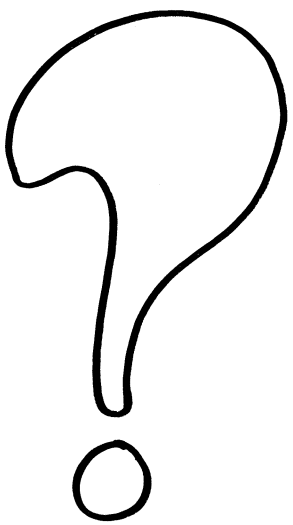
Since the arithmetic values of B and C are both non-zero, they are both true. Therefore, NOT B and NOT C are both false; so a Ø is displayed.

```
310 PRINT "DEPRECIATION METHODS:"
320 PRINT "                1. STRAIGHT LINE"
330 PRINT "                2. DOUBLE DECLINING TO STRAIGHT LINE"
340 PRINT "                3. SUM-OF-THE-YEARS DIGITS"
350 PRINT "ENTER NUMBER OF DEPRECIATION METHOD";
360 INPUT A6
370 PRINT
380 PRINT "IF RATE OF RETURN IS DESIRED, USE A DISCOUNT RATE OF ZERO"
390 PRINT "ENTER DISCOUNT RATE AND TAX RATE";
400 INPUT A7,A8
410 PRINT
420 FORMAT F3.0,F13.2,F14.2,F12.2,2F13.2
430 PRINT
440 PRINT
450 LET B1=0
460 LET B2=0
470 LET B3=0
480 LET B4=0
490 LET B5=0
500 IF A6>1 THEN 540
510 FOR X=1 TO A4
520 LET B[X]=(A3-A5)
530 NEXT X
540 IF A6#2 THEN 630
550 FOR X=1 TO A4
560 LET T1=B9/((A4+1)-X)
570 LET B[X]=A3*2/A4*(1-2/
580 IF T1 >= B[X] THEN 600
590 GOTO 610
600 LET B[X]=B9/((A4+1)-X)
610 LET B9=B9-B[X]
620 NEXT X
630 IF A6<3 THEN 670
640 FOR X=1 TO A4
650 LET B[X]=(A3-A5)*(A4-X+1)
660 NEXT X
670 IF A4 >= A2 THEN 710
680 FOR X=(A4+1) TO A2
690 LET B[X]=0
700 NEXT X
710 FOR X=1 TO A2
720 LET C[X]=A[X]-B[X]
730 LET D[X]=C[X]*A8
740 LET E[X]=C[X]-D[X]
750 IF E[X] >= 0 THEN 770
760 LET E[X]=E[X]*(-1)
770 LET F[X]=E[X]+B[X]
780 LET B1=B1+A[X]
790 LET B2=B2+B[X]
800 LET B3=B3+D[X]
810 LET B4=B4+F[X]
820 NEXT X
830 IF A7=0 THEN 890
840 FOR X=1 TO A2
850 LET G[X]=F[X]/((1+A7)↑X)
860 LET B5=B5+G[X]
870 NEXT X
880 GOTO 1110
890 FOR R=0 TO 0.99 STEP 0.01
900 LET B5=0
910 FOR X=1 TO A2
920 LET G[X]=F[X]/((1+R)↑X)
930 LET B5=B5+G[X]
940 NEXT X
950 R1=R
```

An Introduction To Hewlett Packard BASIC Programming

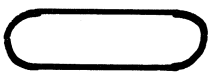
Define The Problem!

- What information do you have
- What kind of answer do you want
- What do you have to do to get your answers

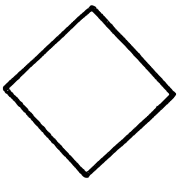


FLOWCHARTS

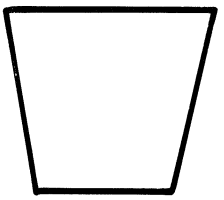
* ARE USED TO HELP DEFINE AND SOLVE THE PROBLEM. THESE SYMBOLS WILL HELP TO SET UP THE FLOWCHART :-



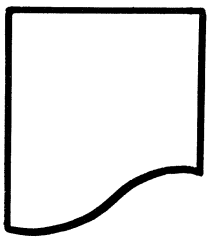
START



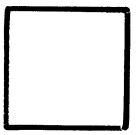
MAKE A DECISION



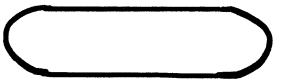
GET INFORMATION



WRITE OUT THE ANSWER



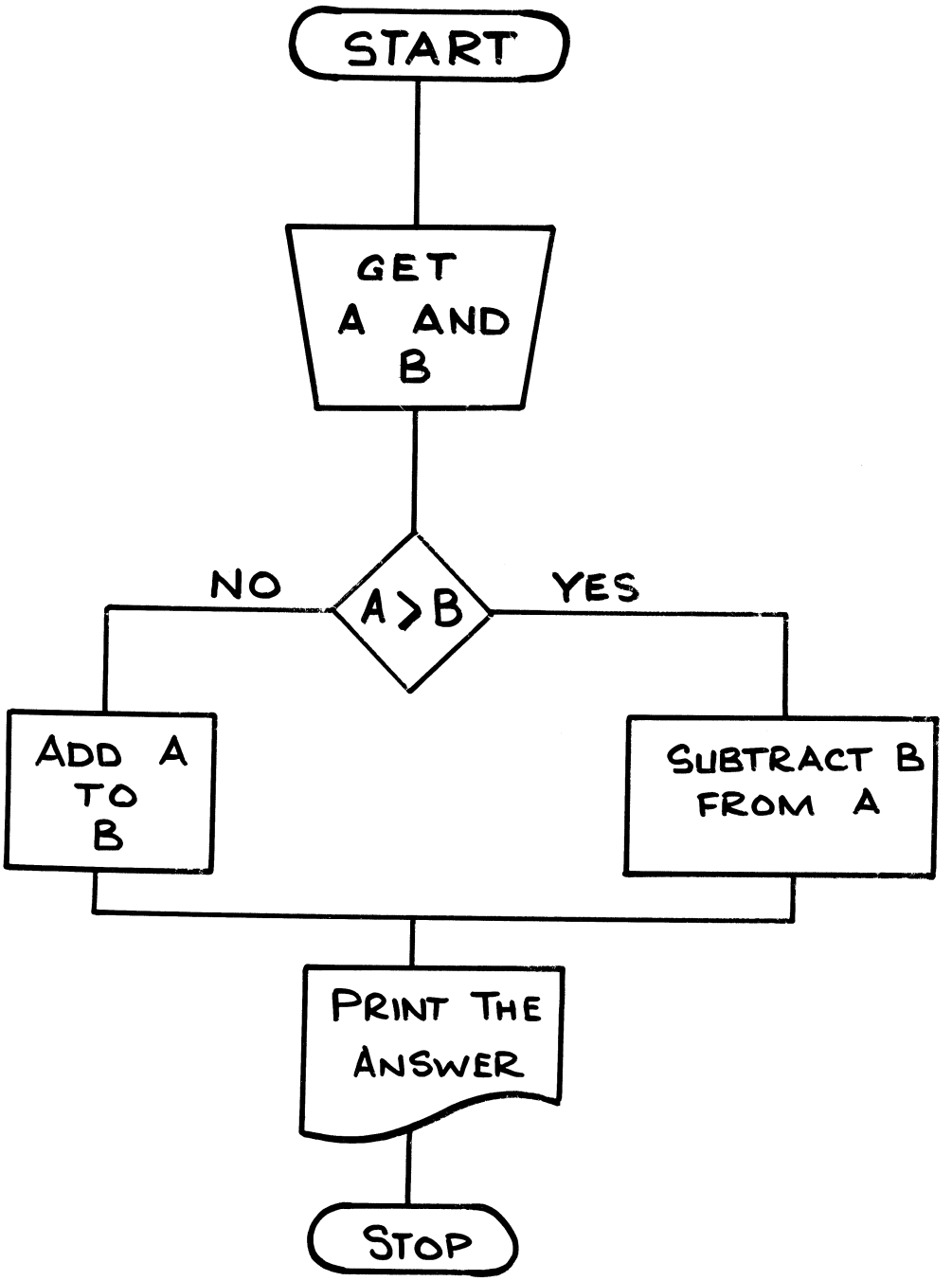
CALCULATE ANSWER



STOP

USE FLOWCHARTS

● SAMPLE FLOWCHART

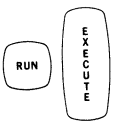


* PROGRAM WRITING

- EACH LINE MUST HAVE A LINE Nº.
- TO ENTER A LINE INTO CALCULATOR MEMORY PRESS 'END OF LINE' KEY.
- BEFORE STARTING TO PROGRAM USE A FLOW CHART.

* PROGRAM EXECUTION

- TO EXECUTE OR RUN A PROGRAM IN THE CALCULATOR PRESS :-



- TO HALT THE PROGRAM OTHER THAN AT THE END PRESS 'STOP'

NOW TO : BEGINNERS

ALL PURPOSE

SYMBOLIC

INSTRUCTION

CODE

IF... THEN

LET

RUN

PRINT

FORMAT

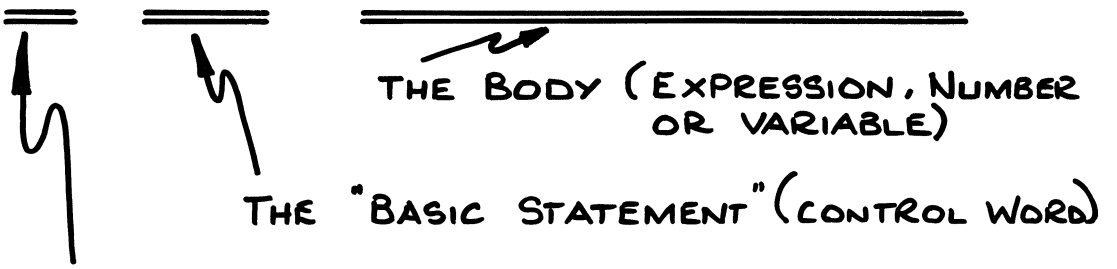
INPUT

DISP

FOR... NEXT

WRITE

* BASIC STATEMENTS



This is a BASIC *statement*:

10 INPUT A,B,C,D,E

- STATEMENTS OR LINES HAVE A UNIQUE NUMBER. THE CALCULATOR USES THIS NUMB. TO KEEP LINES IN ORDER.

- LINES CAN BE ENTERED IN ANY ORDER AND USUALLY WITH GAPS BETWEEN FOR FUTURE INSERTIONS.

REMEMBER! LINES ARE LIKE SENTENCES THEY MUST BE GRAMMATICALLY CORRECT.

* CONSTANTS

- GENERAL FORM :

  A DECIMAL NUMBER BETWEEN MINIMUM OF 10^{-99} THRO' 0 TO A MAXIMUM OF 10^{99}.

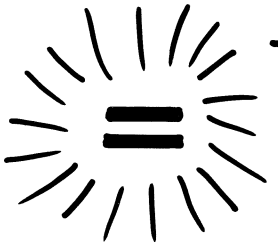
- EXAMPLE :

  25 , 15.75 , -87 , 1234.5678901

* CONSTANTS ARE USED TO REPRESENT FIXED NUMERIC VALUES. THEY CANNOT BE CHANGED BY THE PROGRAM.

* ARITHMETIC OPERATORS

BASIC NOTATION	STANDARD NOTATION	MEANING
+	+	ADDITION
-	-	SUBTRACTION
*	· OR X	MULTIPLICATION
/	÷ OR /	DIVISION
<	<	LESS THAN
>	>	MORE THAN
<=	≤	LESS THAN OR EQ.
>=	≥	MORE THAN OR EQ.
( ) OR [ ]	( ) OR [ ]	BRACKETS
↑	A^2	EXPONENTIATION
# OR <>	≠	NOT EQUAL TO



THIS OPERATOR DOES NOT ALWAYS MEAN EQUALITY

* Special use of Equal Signs

The {=} sign means :-

"takes the value of"

Examples :-

X = A+B : X takes the value of A plus B

X = X+1 : X takes the value of the old X plus the constant 1

X = X*B : X takes the value of the old X Times the value of variable B

X = A*B : X takes the value of A multiplied by B

= Means Takes The Value Of !

REM

* General Form :

```
10 REM {Programmer Remarks}
```

* Example :

```
10 REM  The Following Program
20 Rem  calculates N factorial
30 .....
40 .....
```

* The Rem (Remarks) statements allows a programmer to insert notes in the program. The calculator ignores all lines with Rem. The Rem line will be printed in the listing of the program.

! • Use Rem For Easy Program Reading !

ASSIGNMENT

⇒ **LET** IS AN OPTIONAL STATEMENT!

* GENERAL FORM:

10 LET {ARITHMETIC FORMULA}

* EXAMPLE:

20 LET X = A + B + C

- NORMALLY ON 9830A THE FORM WOULD BE:

20 X = A + B + C

- LITERAL MEANING LET X TAKE THE VALUE OF THE SUM OF A, B AND C

NOTE! THERE MUST ALWAYS BE A VARIABLE TO THE LEFT OF THE '=' OPERATOR

PRINT

* General Form:

25 PRINT {information to be printed}

- Any text must be enclosed within quotation marks " "
- A stand alone PRINT will give a line feed.
- Spaces are not ignored when within a quote field.
- When printing variables a comma will give 5 columns 15 characters a semicolon will pack the results into 12 columns with six character field width.

* Examples :-

```
13 X=3
15 Y=4
17 PRINT "X IS EQUAL TO";X
27 PRINT
37 PRINT X"SQUARED ="X↑2;
47 PRINT Y"SQUARED ="Y↑2
57 PRINT 1,2,X,Y,5
67 PRINT 1;2;X;Y;5
77 PRINT "THE SQUARE ROOT OF"X"SQUARED
        PLUS"Y"SQUARED EQUALS"SQR(X↑2+Y↑2)
```

* RESULTS OF THE EXAMPLES

Lines 13 & 15: , Variables must have assigned values before being used in a PRINT statement.

Line 17: The quote field is printed exactly as seen. Spaces are not ignored in quote fields as they are in other places. The semicolon between the quote field and the variable, X, is not needed since a semicolon would have been assumed anyway. The variable, X, must have an assigned value or an error occurs when the program is run. With X=3 when the program is run, the printout is:

```
X IS EQUAL TO 3
```

Line 27: This statement tells the printer to skip a line before doing any more printouts. Additional printouts can then begin on the following line.

Lines 37 & 47: A variable, a quote field, and an expression are all designated in each of these PRINT statements. Ending the first of two PRINT statements with a semicolon (as in line 37) causes both printouts to appear on the same line. Here is the printout:

```
 3    SQUARED = 9     4    SQUARED = 16
```

A comma could be used instead of the semicolon. But then the printout would be:

```
 3    SQUARED = 9                4    SQUARED = 16
```

Without any punctuation at the end of line 37, the printout would be:

```
 3    SQUARED = 9
 4    SQUARED = 16
```

Lines 57 & 67: Both constants and variables are printed. The only difference between the two statements is that in line 57 the different items are separated by commas, whereas in line 67 they are separated by semicolons. Here are the two printouts:

```
 1             2             3             4             5
 1  2  3  4  5
```

When only commas are used (as in line 57), successive printout fields are 15 character spaces apart and as many as five values can always be printed on the same line.

However, when semicolons are used, as in line 67, the values are 'packed' together in the printout. As many as 12 values can be output on the same line (with a field width of 6 character spaces per value) if semicolons are used. But if any value requires more than 4 character spaces (remember, one space is always allocated to the sign), the number of available fields per line is reduced.

Line 77: A long line of text is combined with variables and an expression. The printout is:

```
THE SQUARE ROOT OF 3    SQUARED PLUS 4    SQUARED EQUALS 5
```

DISP

* General Form:

30 DISP {information to be displayed}

- In general the 'DISP' statement functions in the same way as the PRINT. Only 32 characters can be displayed at one time.

* The 'DISP' function is extremely useful in setting up a conversation between operator and machine e.g. in 'INPUT' situations

* Examples:

```
10 B=5
20 DISP "THE VALUE OF B IS";B
30 DISP B+2-3
40 DISP -1111,2222;3333
```

Lines 20, 30, & 40: These statements merely show that the same operations that are allowed in a PRINT statement are also allowed in a DISP statement. The spacing between fields would be the same with either PRINT or DISP. Here are the three displays:

```
THE VALUE OF B IS 5

 22

-1111          2222          3333
```

STOP, END

* EITHER 'STOP' OR 'END' CAN BE USED TO TERMINATE PROGRAM EXECUTION.

- WHEN STOP IS ENCOUNTERED THE LINE COUNTER CURRENT POSITION IS RETAINED.
- WHEN END IS ENCOUNTERED THE LINE COUNTER IS RESET TO THE LOWEST NUMBERED LINE IN MEMORY

NOTE! THE KEYS 'STOP' AND 'END' CANNOT BE USED AS A PROGRAM STATEMENT.

EXAMPLE:

```
10 P=12
20 DISP P
30 STOP
```

(press: CONT EXECUTE)

```
40 DISP P↑2
50 END
```

ERROR 50 THE LAST STATEMENT EXECUTED BY THE CALCULATOR WAS NOT **END!**

INPUT

* General Form:

40 INPUT {variable list}

- The 'INPUT' statement allows values to be given to variables from the keyboard.
- All types of variables can be input: Simple. Strings. Matrix Elements.
- Program will halt at an 'INPUT' statement with a '?' in the display. Enter value and execute.

* Examples:

```
12 INPUT D,D1,D2
22 INPUT D[1],D[2]
32 DISP "R EQUALS";
42 INPUT R
```

Line 12: The simple variables, D, D1, and D2, are to be assigned values. When more than one variable is designated in an INPUT statement, the variables must be separated by commas.

Line 22: The array variables, D(1) and D(2), are to be assigned values. Since the array variables are less than D(11), they need not be dimensioned (see dimensioning rules on page 3-36).

Lines 32 & 42: By ending line 32 with a semicolon, the text, R EQUALS, will be displayed together with the next display in the program, which is the '?' from line 42. So the display will be:

```
R EQUALS?
```

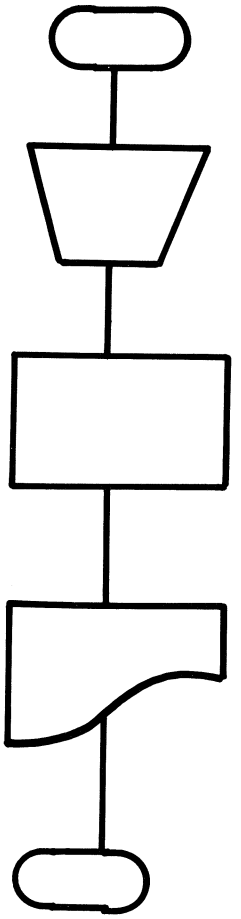
Then the variable, R, in line 42 can be assigned a value.

LET'S HAVE A SAMPLE PROBLEM!

1. GIVE THE CALCULATOR FOUR VALUES.
2. ADD THE VALUES TOGETHER.
3. PRINT OUT THE ANSWER.
4. STOP THE CALCULATOR.

TRY IT! REMEMBER THE FLOWCHART.

A Solution



BASIC Program

```
10 REM THIS PROGRAM ADDS FOUR NUMBERS
20 INPUT A,B,C,D
30 LET X = A+B+C+D
40 PRINT X
50 END
```

LET — optional

IF

* General Form:

80 IF {comparison} THEN {line no.}

- The line no. is the next line to be executed if the test is successful

- The logical evaluation or comparison is carried out and if the result is
- True: Program Execution continues at the specified line number
- False: Program Execution continues at the line following the 'IF'.

```
1 INPUT X,Y

 22 IF X=3 THEN 142
 32 IF INT(X)=3 THEN 222
 42 STOP
  .
  .
  .
313 IF Y<X↑2 THEN 333
323 IF Y THEN 363
333 STOP
  .
  .
  .

404 IF X AND Y THEN 424
414 STOP
424 PRINT X,Y
434 IF X+Y <= 10 THEN 464
444 DISP Y-X
454 STOP
464 DISP X+Y
474 STOP
```

Line 1: For all of these examples, let's assume that X=3.01 and Y=23.

Lines 22 through 42: The IF statement in line 22 is evaluated as 'false' since 3.01 is not equal to 3. So the following statement, line 32, is accessed. This IF statement is then evaluated. Since the integer value of 3.01 equals 3, the statement is evaluated as 'true'. So the program continues execution at line 222.

Lines 313 through 333: Since Y is greater than X^2, the IF statement in line 313 is evaluated as 'false'. So instead of accessing line 333, the program continues execution in its normal sequence with line 323. Line 323 is logically evaluated as 'true' since Y has a non-zero value. So the program continues execution at line 363.

GOTO

* General Form:

90 GOTO {line number of the statement to be exec. next.}

NOTE: variables may not be used.

The 'GOTO' statement is used to alter the normal sequence of program execution.

* Example:

#1

```
 11 A=1
 21 IF A↑2<1000 THEN 51
 31 DISP A;A↑2
 41 STOP
 51 A=A+1
 61 GOTO 21
```

#2

```
 105 Z=0
 110 X=1
 115 Z=X+Z
 120 DISP X;Z/X
 125 X=X+1
 130 GOTO 115
```

Example No. 1: This program 'loops' (repeats part of itself) several times until the IF statement, line 21, is no longer true; that is, until A↑2≥1000. If A↑2<1000, the variable, A, is incremented by 1 in line 51. Then the GOTO statement, line 61, is executed causing the program to loop back to line 21.

Example No. 2: The GOTO statement in this program causes the program to loop between lines 115 and 130. This is a 'closed loop'; that is, there is no program statement in the loop that can cause the loop to be exited. So to halt this program, press the STOP key.

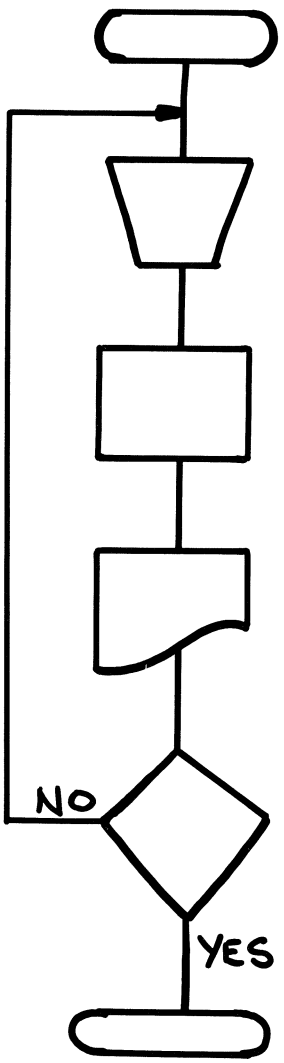
THE NEXT PAGE AND TRY THE PROBLEM

PROBLEM

1. GIVE THE CALCULATOR FIVE VALUES.
2. ADD TOGETHER THE FIRST FOUR.
3. PRINT OUT THE ANSWER.
4. CHECK THE FIFTH VALUE FOR ZERO.
   - IF ZERO STOP THE PROGRAM.
   - IF NON-ZERO GET MORE INPUT.

DON'T FORGET THE FLOWCHART!

A Solution



START

GIVE THE CALCULATOR THE VALUES:
A. B. C. D. E.

SUM = A+B+C+D

PRINT RESULT

IS 'E' ZERO

STOP

```
10  REM THIS PROGRAM ADDS FOUR
20  REM NUMBERS. AND TESTS A FIFTH
30  REM FOR ZERO.
40  INPUT A,B,C,D,E
50  X = A+B+C+D
60  PRINT "THE ANSWER IS", X
70  IF E = 0 THEN 90
80  GO TO 40
90  END
```

FOR . . . NEXT

* General Form:

```
10  FOR {variable} = {starting value} TO {end value}
 :
50  NEXT {variable}
```

- The 'FOR....NEXT' statements allow for controlled repetition of a group of statements within a program.
- The advantages of using FOR...NEXT instead of 'IF' is shown in the following example:

Using IF

```
10 N=1
20 IF N>100 THEN 60
30 DISP N
40 N=N+1
50 GOTO 20
60 END
```

Using FOR . . . NEXT

```
10 FOR N=1 TO 100
20 DISP N
30 NEXT N
40 END
```

In the example the program is easier to key in, takes up considerably less calculator memory, and executes much faster if the FOR . . . NEXT loop is used.

NOTE! VARIABLES CAN BE USED FOR THE START AND STOP VALUES

FOR ... NEXT
(continued)

* EXAMPLES :

#1
```
11 Z=0
21 FOR P=90 TO 100
31 Z=Z+P
41 NEXT P
51 PRINT "THE TOTAL IS"Z
61 END
```

#2
```
1 Z=0
11 INPUT M,N
21 FOR P=M TO N
31 Z=Z+P
41 NEXT P
51 PRINT "THE TOTAL IS"Z
61 END
```

#3
```
19 FOR A=1 TO 12
29 PRINT A;
39 NEXT A
49 PRINT A
59 END
```

#4
```
9 Z=0
19 INPUT V
29 FOR A=1 TO V
39 Z=Z+A
49 NEXT A
59 PRINT "THE AVERAGE IS"Z/(A-1)
```
or
```
59 PRINT "THE AVERAGE IS"Z/V
69 END
```

* FOR...NEXT LOOPS CAN BE NESTED. BUT MAKE SURE LOOPS DO NOT OVERLAP e.g.

Correct

```
10 FOR E=1 TO 15
20 FOR Z=3 TO 12
 .
 .
 .
 .
80 NEXT Z
90 NEXT E
```

Incorrect

```
10 FOR E=1 TO 15
20 FOR Z=3 TO 12
 .
 .
 .
 .
80 NEXT E
90 NEXT Z
```

* THE FOR..NEXT LOOPS SHOWN UP TO NOW ALL INCREMENT TO THE STOP POINT BY '1' - TO INCREMENT BY OTHER THAN 1 A STEP VALUE MUST BE ADDED

FOR . . . NEXT with STEP

* STEP IS USED TO INCREMENT A LOOP BY SOME VALUE OTHER THAN 1. AND MAY ALSO BE NEGATIVE.

EXAMPLES:

#1

```
12 FOR X=1 TO 4
or
12 FOR X=1 TO 4 STEP 1
22 PRINT X↑2-X;
32 NEXT X
42 END
```

Example No. 1: Either version of line 12 could be used with identical results. In either case, when the program loops, the value of X is incremented by 1. Therefore, if you want the variable to increment by 1, it is unnecessary to use the STEP feature.

#2

```
15 FOR C=1 TO 6 STEP 2
25 DISP C;
35 NEXT C
45 END
```

Example No. 2: This loop is executed three times: when C=1, 3, and 5. The final display is:

```
1  3  5
```

#3

```
17 FOR T=4 TO -8 STEP -4
27 PRINT T;T↑2
37 NEXT T
47 END
```

Example No. 3: This program shows that FOR . . . NEXT loops can be incremented by negative values, in this case, –4. This loop is executed four times: when T=4, Ø, –4, and –8. The printout is:

```
 4    16
 0    0
-4    16
-8    64
```

NOTE! DO NOT USE 'GOTO' OR 'IF' TO ENTER A LOOP, ONLY A 'FOR'

WAIT

* THE WAIT STATEMENT INTRODUCES A TIMED DELAY BETWEEN TWO PROGRAM STATEMENTS.

* GENERAL FORM :

```
30 WAIT 2000
```

- THE VALUE IN THE WAIT STATEMENT IS THE SPECIFIED DELAY IN MILLISECS. MAX. APPROX 33 SECONDS.

- THE MILLISEC. VALUE MAY BE A CONSTANT - VARIABLE - EXPRESSION.

PROBLEM :

WRITE A SHORT PROGRAM TO PRINT OUT THE ANGLES AND THEIR CORRESPONDING SINE VALUE. THE START - END AND FREQUENCY VALUES OF THE ANGLES TO BE INPUT FROM KEYBOAD.

* THE READ.....DATA STATEMENTS COMBINE TO ASSIGN VALUES TO SPECIFIC VARIABLES.

- READ STATEMENT DETERMINES THE VARIABLE.
- DATA STATEMENT DETERMINES VALUE TO BE ASSIGNED.

- WHEN READING DATA, A POINTER IS USED IN THE MACHINE TO KEEP TRACK OF THE ELEMENTS BEING READ. DATA IS READ FROM RIGHT TO LEFT.

  - POSITION OF THE DATA STATEMENT IN THE PROGRAM IS IMMATERIAL.

EXAMPLES:

#1

```
10 READ A,B,C
 .
 .
 .
40 READ A1,B1
 .
 .
 .
70 READ P
 .
 .
 .
200 DATA 4,5,6,7.31,2.69,0
```

#2

```
12 FOR I=1 TO 5
22 READ X
32 PRINT X"SQUARED ="X↑2
42 NEXT I
52 DATA 24,8.3,17,19,3.2
62 END
```

#3

```
7 DATA 3,4,8,15,7,24,47,1104
17 FOR J=1 TO 4
27 READ X,Y
37 PRINT X;Y;SQR(X↑2+Y↑2)
47 NEXT J
57 END
```

#4

```
18 READ N
28 FOR P=1 TO N
38 READ D,D1
48 PRINT D↑2-D1
58 NEXT P
68 DATA 3
78 DATA 9,1,8,4,7,9
88 END
```

READ ... DATA with RESTORE

* RESTORE IS USED TO RE-POSITION THE POINTER SO ALLOWING DATA TO BE RE-READ.

GENERAL FORM:

80 RESTORE OR 100 RESTORE 150

RESETS POINTER TO BEGINNING OF THE LOWEST NO. DATA STATEMENT.

RESETS POINTER TO THE BEGINNING OF THE DATA STATEMENT LN. 150.

EXAMPLES:

#1

```
10 FOR I=1 TO 5
20 READ A
30 PRINT A"SQUARED ="A↑2
40 NEXT I
50 RESTORE
60 PRINT
70 FOR J=1 TO 3
80 READ B
90 PRINT B"CUBED ="B↑3
100 NEXT J
110 DATA 4,9,12,8,27
120 END
```

#2

```
10 READ N
20 FOR I=1 TO N
30 READ A
40 PRINT A"SQUARED ="A↑2
50 NEXT I
60 RESTORE 130
70 PRINT
80 FOR J=1 TO 3
90 READ B
100 PRINT B"CUBED ="B↑3
110 NEXT J
120 DATA 5
130 DATA 4,9,12,8,27
140 END
```

TRY THESE TWO EXAMPLES ON THE CALCULATOR

FORMAT

* BASICALLY THE FORMAT STATEMENT ALLOWS OUTPUT SPECIFICATIONS e.g. SPACES FORMATTING etc. TO BE DETAILED FOR THE WRITE STATEMENTS.

  - SYMBOLS NOT ABLE TO BE OUTPUT BY A PRINT STATEMENT CAN ALSO BE OUTPUT.

  - THE WRITE STATEMENT REFERENCES THE FORMAT STATEMENT BY A LINE NUMBER e.g.

    10 WRITE (15, 200) A

    FORMAT STATEMENT Ln

FORMAT & WRITE STATEMENTS CAN OUTPUT THE FOLLOWING :

- TEXT (IN QUOTE FIELD)
- VALUES FOR VARIABLES
- VALUES FOR EXPRESSIONS
- CONSTANTS

FORMAT CAN APPEAR ANYWHERE IN THE PROGRAM!

FORMAT
(continued)

* FOR NUMERICS, FORMATTING OF OUTPUT IS EASILY ACCOMPLISHED:

F w.d	F : FIXED FORMAT w : FIELD WIDTH d : SIGNIFICANT DIGITS
E w.d	E : EXPONENTIAL FORMAT OR SCIENTIFIC.

* OTHER SPECIFICATIONS:

X	: INDICATES A SPACE
/	: INDICATES CRL (LINE FEED ON PRINTER)
"TEXT"	: QUOTE FIELD
B	: ALLOWS SYMBOLS NOT NORMALLY ABLE TO BE OUTPUT.

NOTE! ANY COMBINATION CAN APPEAR IN A FORMAT STATEMENT, ITEMS SEPARATED BY COMMAS.

* SPECIFICATIONS CAN BE DUPLICATED BY A REPEAT FACTOR.

10 FORMAT 2F6.2,4X,F6.2

WRITE

* THE WRITE STATEMENT CAN BE USED LIKE THE PRINT STATEMENT ALTHOUGH IT IS NORMALLY USED WITH FORMAT.

* USING THE WRITE STATEMENT ALLOWS YOU TO SPECIFY THE OUTPUT DEVICE.

GENERAL FORM:

```
10 WRITE (15,80) 1,2,3
```

SELECT CODE

LINE NUMBER OF FORMAT

* IF THE LINE NO. IS REPLACED BY * THEN THE CALCULATOR KNOWS THERE IS NO FORMAT STATEMENT.

Examples

#1

```
10 X=4
20 WRITE (8,*)X"SQUARED ="X↑2
30 END
```

#2

```
12 INPUT S,Z
22 IF S#8 AND S#15 THEN 12
32 WRITE (S,*)Z"SQUARED ="Z↑2
42 END
```

NOTE! THE SELECT CODE CAN BE A VARIABLE.

WRITE (15,*)..... IS IDENTICAL TO

PRINT.....

PRINT with TAB

* THE TAB COMMAND ALLOWS OUTPUT TO BE LOCATED AT A SPECIFIC CHARACTER POSITION.

* TAB CAN BE USED WITH 'DISP' AND 'WRITE' (WHEN WRITE IS USED LIKE PRINT)

NOTE! CHARACTER POSITIONS Ø THRO' 71 CAN BE USED - THEY ARE ABSOLUTE POSITIONS AND <u>NOT</u> SPACING.

TAB COMMAND CAN ALSO USE A VARIABLE POSITION.

Examples

#1

```
10 INPUT X,Y,Z
20 PRINT "AVERAGE"TAB20"MEAN"TAB40"MEDIAN"
30 PRINT X,TAB20,Y,TAB40,Z
40 END
```

#2

```
10 DEG
20 X=0
30 PRINT TAB(35+25*SIN(X),X;"DEG"
40 X=X+30
50 GOTO 30
```

GOSUB ... RETURN

* GOSUB.... RETURN ALLOWS A ROUTINE TO BE EXECUTED MANY TIMES IN A PROGRAM, WITHOUT NEEDING TO WRITE THE ROUTINE MORE THAN ONCE.

* THE GOSUB TRANSFERS PROGRAM CONTROL TO A SPECIFIED LINE No. AND RETURN BRINGS CONTROL BACK TO THE MAIN PROGRAM, TO THE NEXT LN AFTER THE GOSUB.

Examples

#1

```
10 INPUT A
 .
 .
 .
60 GOSUB 1000
70 PRINT A
 .
 .
 .
200 GOSUB 1000
210 PRINT A,N
 .
 .
 .
990 STOP
1000 N=0
 .
 .
 .
1200 RETURN
1210 END
```

#2

```
10 INPUT A,N
 .
 .
 .
60 GOSUB 1000
70 PRINT A,N
 .
 .
 .
900 STOP
 .
 .
1000 IF A*N<150 THEN 1020
1010 GOSUB 2000
1020 A=A*5
 .
 .
1200 RETURN
 .
 .
1990 STOP
2000 A=(A*N)/5
 .
 .
2200 RETURN
2210 END
```

QUESTION?

WHAT WILL BE THE ORDER OF EXECUTION OF EXAMPLES 1 & 2?

GOTO & GOSUB with OF

* THE BASIC STATEMENTS ARE AS PREVIOUSLY DISCUSSED – HOWEVER THE 'OF' PART ALLOWS A COMPUTED GOTO OR GOSUB.

GENERAL FORM:-

50 GOTO X OF 130, 150, 210

* BY COMPUTING X TO BE EITHER 1, 2, OR 3. WILL TAKE PROGRAM CONTROL TO LINE NO. 130, 150 OR 210.

Examples

#1

```
10 INPUT X
20 GOTO X OF 300,330,360,390
30
 .
 .
 .
500 END
```

#2

```
10 INPUT X,Y
20 GOSUB ABS(X-Y) OF 200,300,400
30
 .
 .
 .
110 GOTO 10
 .
 .
 .
400 PRINT X↑2+Y↑2
 .
 .
 .
500 RETURN
```

DIM

* In many instances the calculator must be told which variables are to be used and the sizes for example of an array.

  • The statement used can be a DIM or short for DIMension.

General Form :-

```
10  DIM A[20,20], B$[20], CI[25]
```

A S [20,20] ..... The 'S' provides Split precision

A I [20,20] The 'I' provides integer precision

* With neither 'S' or 'I' included then full precision assumed.

Note ! DIM can be anywhere in a program, and there can be more than one DIM line.

* Using DIM only one array can be stored on a tape file !

* THE COM STATEMENT IS SIMILAR TO DIM IN THAT IT RESERVES SPACE IN MEMORY FOR VARIABLES.

BUT!

* COM MUST BE LOWEST LINE N°.!
* COM MUST BE ENTERED FIRST !
* COM CANNOT BE CHANGED ONCE ENTERED!
* COM ALLOWS STORAGE ON A FILE OF MORE THAN ONE VARIABLE AND SIMPLE VARIABLES!

REMEMBER! TAKE CARE WHEN USING. COM.

* STUDY OPERATING *

MANUAL

*