PROGRAMMABLE CALCULATOR

CASIO FX-702P

INSTRUCTION MANUAL

First of all, we would like to thank you very much for purchasing this product. This instrument is a high performance, programmable calculator which incorporates microcircuitry to provide for repetitious or complex calculation. The most important feature of this instrument is that it uses BASIC program language. This provides a conversation type language for problem solving. Operation is easy, even for a beginner. Additionally, programming is simple using one key commands which permit highly efficient keying.

This instrument's calculation management functions are generally separated as follows.

1. Manual Calculation
2. Program Calculation

It not only performs high level program calculations like a computer but is also designed for easy operation as a scientific calculator.

CONTENTS

Prior to Use

This calculator is brought to you as a result of our highly developed electronic technology. Strict quality control procedures and a rigid inspection process were employed in its production. Please follow the precautions below to ensure long equipment life.

■ Use Precautions

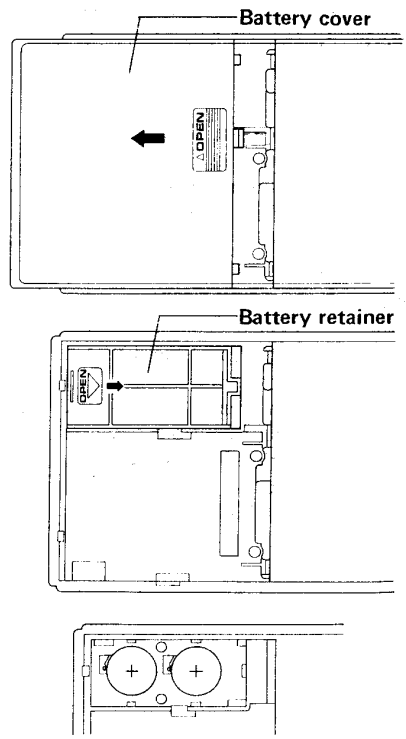
- This calculator is composed of precision electronic parts. Never try to take it apart. Avoid dropping or throwing. Do not permit it to undergo extreme temperature variations. Do not store or leave in any location where there is dampness, high temperature or dust. During periods of low temperature, the display response may be slower or there may be no display. Normal display will resume when the temperature returns to normal.
- Do not use any other equipments than the optional ones.
- While the calculator is performing calculations, a "—" will be displayed. Key operation during this time will not be effective except for one section so please pay attention to the display at all times and press the keys carefully.
- Concerning the batteries, even when the calculator is not used to any great extent, please change the batteries every 2 years. If worn out batteries are used, they may leak and cause damage to the instrument. Never leave worn out batteries in the instrument.
- To clean the calculator, use a soft, dry cloth or a damp cloth and mild detergent to wipe it off. Never use paint thinner or benzine.
- In case of malfunction, contact the store where it was purchased or a nearby dealer.
- Prior to seeking maintenance, please read the instruction manual again and also check the power supply as well as program or operational error.

■ Power Supply and Battery Replacement

This calculator uses 2 lithium batteries (CR2032).
When the display is still weak even though the contrast control is turned to full strength, this indicates that the batteries are run down. In this case, replace the batteries as shown in the figure below. If the calculator is used with weak batteries, this may cause errors.
Even if the calculator is functioning normally, be sure to replace the batteries every 2 years.

- **How to replace the batteries.**

(1) After turning the power switch OFF, slide the battery cover and remove from the rear of the instrument.
(2) Push the battery retainer in the direction of the arrow and remove.
(3) Remove the two run down batteries (this will be easier if the instrument is tapped with the battery compartment facing downward).
(4) If the surface of the new batteries contains white powder or dust it may cause a bad contact. Wipe them off with a dry cloth. Install the batteries with the positive side ⊕ facing up.
(5) Press batteries in with the battery retainer and slide the battery cover closed.



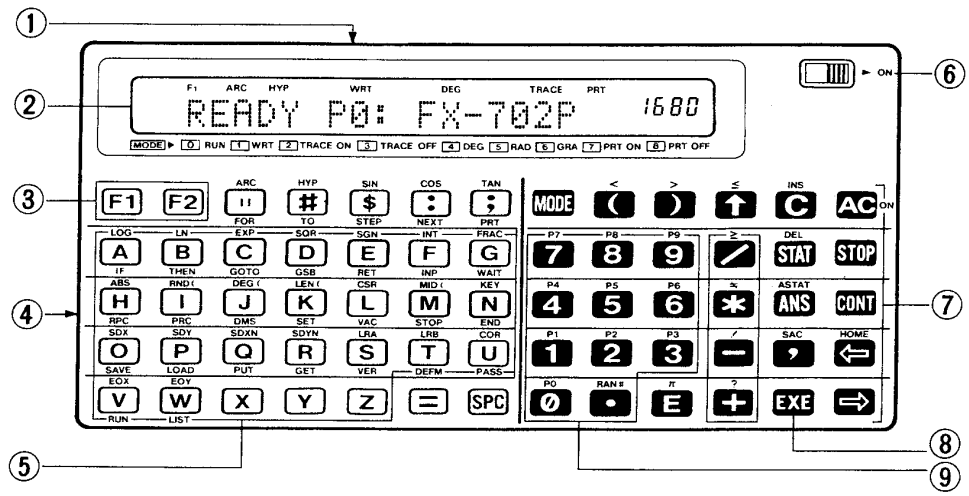
Caution

Be sure to perform the following operations after battery replacement.
Turn the power switch to ON.

MODE 1 **CLR ALL** EXE
MODE 0 F1 SAC F2 DEFM 0 EXE **VAC** EXE

Chapter 1

Each Section's Nomenclature and Operation



① Display Contrast Dial
② Display Window
③ Function Keys
④ Adaptor Connector
⑤ Alphabet Keys
⑥ Power Switch
⑦ Keyboard
⑧ Execute Key
⑨ Numerical Keys and Decimal Key

1-1 Each Section's Nomenclature

Each key has 3 separate operations.
Press the keys directly for the function printed on the key. Press [F1] and then the key for the function printed above the key. Press [F2] and then the key for the function printed below the key.

Example

LOG ..... 1st function mode
[A] ..... direct mode
IF ..... 2nd function mode

[F1] 1st Function Key

If this key is pressed the 1st function mode is selected ("F1" is displayed) and the 1st function printed above each key can be used.

[F2] 2nd Function Key

If this key is pressed the 2nd function mode is selected ("F2" is displayed) and the 2nd function printed below each key can be used.

[MODE] Mode Key

Calculator situation and unit of angular measure are designated by pressing the [MODE] key and numerical keys [0] through [8].

[MODE] [0] ..... "RUN" is displayed. Manual and program calculation can be performed.

[MODE] [1] ..... "WRT" is displayed. Program write and checking and editing can be performed.

[MODE] [2] ..... "TRACE" is displayed. Execution trace can be performed. (See page 38 for details.)

[MODE] [3] .... When "TRACE" is displayed, it will be erased. Execution trace function will be cancelled.

MODE 4 .... "DEG" is displayed. Unit of angular measure designation will be "degree".

MODE 5 .... "RAD" is displayed. Unit of angular measure designation will be "radian".

MODE 6 .... "GRA" is displayed. Unit of angular measure designation will be "gradient".

MODE 7 .... "PRT" is displayed. If a printer is connected, print output can be accomplished. (See page 78 for details.)

MODE 8 .... When "PRT" is displayed, it will be erased and print output will be cancelled.

AC All Clear Key

- When keying in data, the display will be completely cleared and all characters input up to that time will be completely cancelled.
- If pressed during program run, program run will stop and return to input ready.
- When an error message is displayed, press to cancel error.
- When AUTO POWER OFF (automatic power saving function, refer to page 13) is in operation and the display goes OFF, press to turn power ON again.

INS C Clear Key (input character deletion key)/Insert Key

- When keying in data, the character just prior to the cursor will be cancelled. The characters to the right of the cursor will move one position to the left.
- In the 1st function mode, this key becomes the insert key. When it is pressed, the character above the cursor and those to the right of the cursor will be moved one space to the right and a new character can be inserted.

STOP Stop Key

If pressed during program run, "STOP" will be displayed. Program run will stop at the end of one line. If pressed during execution trace, program number, line number and program content will be displayed.

CONT Continue Key

If pressed when program has been stopped by the STOP key, or stop sentence, program run will begin again from the next sentence. During execution trace, press to advance to the next sentence.

DEL STAT Stat Key/Delete Key

- Press this key to input data for performing statistical calculations.
  Example: Standard deviation . . . . x STAT . Regression calculation . . . . . x, y STAT .
- In the 1st function mode, press this key to delete data when performing statistical calculations.

ASTAT ANS Answer Key/Answer Stat Key

- For manual or program calculation, press this key to recall the previous calculation result.
- In the 1st function mode, press this key to display the result of a statistical calculation (number of data, sum of x, sum of y, square sum of x, square sum of y and product sum of $x \cdot y$).

HOME ← Cursor Key/Home Position Key

- Press this key once to move the cursor one position to the left.
- In the 1st function mode, press to move the cursor to the home position (beneath the first input character).

→ Cursor Key

Press this key once to move the cursor one position to the right.

EXE Execute Key

- Press this key instead of "=" to get the result of a manual calculation.
- When in the "WRT" mode, press this key for program write on a line-by-line basis. Write cannot be accomplished if this key is not pressed.
- When in the "RUN" mode, press this key to input during program run.

SAC , Comma Key/Statistical Use Memory Clear Key

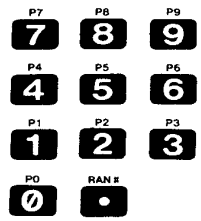
- Press to write a comma.
- In the 1st function mode, press to clear statistical use memory.

Parenthesis Keys/Comparative Keys

- Press these keys at the parantheses positions when performing parenthesis calculations.
- In the 1st function mode, press these keys for comparative relationships.

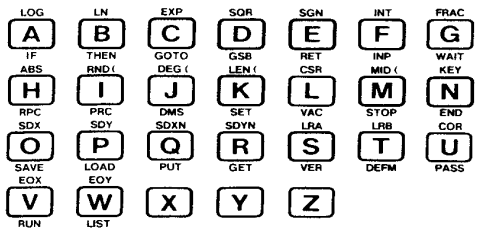
Calculation Command Keys/Comparative Keys/Character Keys

- + − × ÷ Symbols for addition, subtraction, multiplication and division. Press for desired function. Press ↑ to obtain powers of numbers.
- In the 1st function mode, press ≤ ≥ ≠ for comparative relationships. Press ! to get factorial results. Press ? to write a question mark.



Numerical Keys/Program Number Keys

- Press to enter numbers for calculation. Press . to insert a decimal point.
- In the 1st function mode, these keys are program number designators. When writing, program can be started.



Alphabet Keys/One Key Command Keys

- When writing programs or when writing commands or function commands, if these keys are pressed, alphabetical letters will be displayed.
- In the 1st function mode, one key function commands will be displayed.
- In the 2nd function mode, one key program commands will be displayed.

= Equal Key

Press this key to write substitution symbol or equal symbol.

SPC Space Key

Press to insert a space between characters during input.

" Strings Key/One Key Command Key

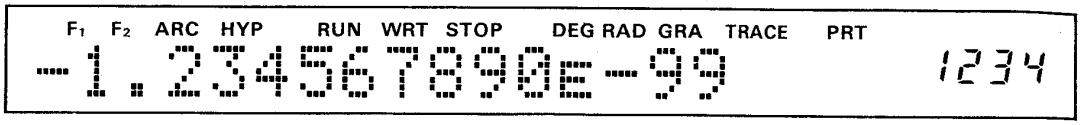
- Press this key at the beginning and end of character strings when inputting or displaying character constants.
- In the 1st function mode, one key function commands will be displayed.
- In the 2nd function mode, one key program commands will be displayed.

Character Keys/One Key Command Keys

- Press to use the symbols written on the keys.
- In the 1st function mode, one key function commands will be displayed.
- In the 2nd function mode, one key program commands will be displayed.

1-2 How to Read the Display

Dot Matrix Display



Displays calculation values and results. Each character is composed of dots of 5 wide by 7 high. A maximum of 20 numbers and characters can be displayed. (Zero is displayed as Ø .) If a formula or sentence contains more than 20 characters, the numbers or letters will move to the left. A maximum of 62 characters can be input. The 4 positions at the right side of the display show the remaining program steps. During calculation, a minus sign will be displayed to the right of the 4 digits on the far right side of the display. Also, units of angular measure, such as "DEG", "RAD" and "GRA" or "F1", "F2" (when [F1] or [F2] key is pressed). Also, "ARC", "HYP" (when [F1] [ARC] or [F1] [HYP] keys are pressed), "RUN" (RUN mode), "WRT" (WRT mode), "TRACE" (TRACE mode), "PRT" (PRT mode), etc. These symbols will be displayed for each situation. Also, sexagesimal or alphabetical letters and symbols are displayed as follows.

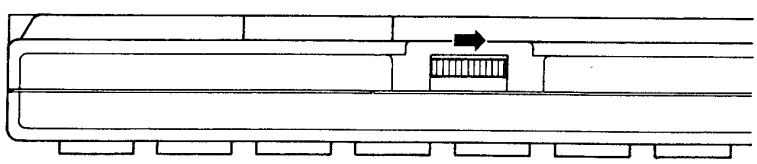
- **Sexagesimal display example**

- **Alphabetical display example**

- **Symbol display example**

1-3 Contrast Adjustment

Display contrast adjustment is accomplished using the control dial on top of the calculator.



Turn in the direction of the arrow to strengthen the display and in the opposite direction to weaken the display. This controls the contrast to compensate for battery strength and also according to personal viewing preference.

Chapter 2

Prior to Beginning Calculation

Manual calculation and program calculation are performed in the "RUN" mode. (RUN display = MODE 0)
Units of angular measure display, such as "DEG", "RAD" and "GRA" may be disregarded if the calculation is not related to angles.

2-1 Calculation Priority Sequence (True Algebraic Logic)

- This calculator determines the calculation priority automatically and calculates using that sequence. Calculation priority sequence is determined as follows.

① Functions (sin, cos, tan, etc.)
② Powers, Factorials
③ x, ÷ (*, /)
④ +, −

When the priority sequence is the same, it will calculate from the beginning (from the left). If parentheses are used, the calculations contained in parentheses take priority.

2-2 Input/Output Number of Units and Calculation Number of Digits

This calculator's input/output units are composed of 10 mantissa positions and 2 exponent positions. The range is 1.0^{-99} to $\pm 9.999999999 \times 10^{+99}$.
Number of input/output positions is 10 mantissa positions and 2 exponent positions. However, internal calculation number of positions and number of stored positions in the memory is 12 mantissa positions and 2 exponent positions. When the input or calculation value is greater than 10 positions, up to 12 positions will be written. Anything greater will be ignored. Output is 10 mantissa positions.

Example: 1.23456789123×100= 123.4567891

When the calculation result is greater than 10^{10} or less than 10^{-3}, it will be automatically displayed exponentially.

Example: 1234567890×10= 1.23456789E 10

(The exponential display is displayed following the mantissa display with an exponential sign "E".)

Example: 1.234÷10000= 1.234E-04

2-3 Basic Calculation

2-3-1 Calculation Symbols and Function Commands

Using BASIC calculation symbols, + and − symbols are the same as standard symbols for addition and subtraction but * and / are used for multiplication and division symbols.
For instance:

```
2+3-4*5/6
```

is expressed as

```
2+3-4*5/6
```

Also, this instrument incorporates the following functions.

Function Name		Form	
Trigonometric Function	$\sin x$	SINx	[F1] [SIN]
	$\cos x$	COSx	[F1] [COS]
	$\tan x$	TANx	[F1] [TAN]
Inverse Trigonometric Function	$\sin^{-1} x$	ASNx	[F1] [ARC] [SIN]
	$\cos^{-1} x$	ACSx	[F1] [ARC] [COS]
	$\tan^{-1} x$	ATNx	[F1] [ARC] [TAN]
Hyperbolic Function	$\sinh x$	HSNx	[F1] [HYP] [SIN]
	$\cosh x$	HCSx	[F1] [HYP] [COS]
	$\tanh x$	HTNx	[F1] [HYP] [TAN]
Inverse Hyperbolic Function	$\sinh^{-1} x$	AHSx	[F1] [ARC] [HYP] [SIN]
	$\cosh^{-1} x$	AHCx	[F1] [ARC] [HYP] [COS]
	$\tanh^{-1} x$	AHTx	[F1] [ARC] [HYP] [TAN]
Square Root	$\sqrt{x}$	SQRx	[F1] [SQR]
Exponential Function	e^x	EXPx	[F1] [EXP]
Natural Logarithm	$\ln x$	LNx	[F1] [LN]
Common Logarithm	$\log x$	LOGx	[F1] [LOG]
Factorial	$x!$	$x!$	[F1] [=]
Change to Integer	INTx	INTx	[F1] [INT]
Cancel Integer Portion	FRACx	FRACx	[F1] [FRAC]
Change to Absolute Value	$\lvert x \rvert$	ABSx	[F1] [ABS]
Change to Symbol	Positive Number → 1 0 → 0 Negative Number → −1	SGNx	[F1] [SGN]
Degrees Minutes Seconds	(Sexagesimal → Decimal)	DEG (x, y, z)	[F1] [DEG(]
Degrees Minutes Seconds	(Decimal → Sexagesimal)	DMSx	[F2] [J DMS]
Coordinate Conversion	(Rectangular → Polar)	RPCx, y	[F2] [H RPC]
Coordinate Conversion	(Polar → Rectangular)	PRCx, y	[F2] [I PRC]

* x, y coordinates and r, θ can be substituted for using X, Y variables.

Rounding Off	(10^y of x is rounded off)	RND (x, y)	[F1] [RND(]
Random Numbers		RAN #	[F1] [RAN#]
Statistical Calculation			
Number of Data n		CNT	—
Standard deviation of x $x\sigma_{n-1}$		SDX	[F1] [SDX]
Standard deviation of y $y\sigma_{n-1}$		SDY	[F1] [SDY]
Standard deviation of x $x\sigma_n$		SDXN	[F1] [SDXN]
Standard deviation of y $y\sigma_n$		SDYN	[F1] [SDYN]
Average of x $\overline{x}$		MX	—
Average of y $\overline{y}$		MY	—
Sum of x Σx		SX	—

Sum of y	Σy	SY	–
Sum of x squared	Σx^2	SX2	–
Sum of y squared	Σy^2	SY2	–
Sum of data product	Σxy	SXY	–
Constant term	A	LRA	[F1] [LRA (S)]
Regression coefficient	B	LRB	[F1] [LRB (T)]
Correlation coefficient	r	COR	[F1] [COR (U)]
Estimated value of x	$\hat{x}$	EOXy	[F1] [EOX (V)]
Estimated value of y	$\hat{y}$	EOYx	[F1] [EOY (W)]

2-3-2 Previous Calculation Result Callout

The result of a manual or program calculation is stored until the next calculation has been executed. This result can be displayed by pressing the [ANS] key.

Example: 741+852=1593
2431−1593=838

Operation:

[7][4][1][+][8][5][2]	741+852
[EXE]	1593
[2][4][3][1][−][ANS]	2431- 1593
[EXE]	838

Also, the number value displayed following the calculation can be used as is in the next calculation.

Example: (Subsequently to the abolve)
838×2=1676

Operation:

[×][2]	838*2
[EXE]	1676

2-3-3 Error Message

If the formula or substitute sentence is not written correctly according to BASIC language or if the calculation range is exceeded, an error will occur at the time of run and an error message will be displayed. The following error messages will be displayed during manual calculation.

ERR-2	(sentence structure error)
ERR-3	(mathematical error)

The following error messages will be displayed during program calculation.

ERR-2 IN PØ-1Ø
program area line number

(A sentence structure error has occured at line 10 in program area PØ)

ERR-3 IN P2-2Ø

(A mathematical error has occurred at line 20 in program area P2)

Furthermore, for the meaning of the error messages, see Page 79 for the error message list.

* When the calculation range is exceeded (±9.999999999E+99), an overflow condition occurs and an error message is displayed. Also, below 1.0E−99, an underflow condition occurs and the calculation result will be "0".

2-4 Key Operation

First, turn the power ON.
"READY P0" is displayed and the calculator is ready for input.

1. Key Input

- **Alphabetical Input**

Example: Input ABC

Operation: [A][B][C]

```
ABC
```

Example: Input SIN

Operation: [S][I][N] (or [F1] [SIN/S])

```
SIN
```

* Either one key command or alphabetical command may be used.

- **Numerical Input**

Example: Input 123

Operation: [1][2][3]

```
123
```

Example: Input 96.3

Operation: [9][6][.][3]

```
96.3
```

- **Symbol Input**

Example: Input $#?

Operation: [$][#][F1] [?]

```
$ # ?
```

- **Input of numbers with exponents**

Example: Input 7.896×10^{15}

Operation: [7][.][8][9][6] [π/E] [1][5]

```
7.896E15
```

Exponential symbol

Example: Input -2.369×10^{-45}

Operation: [−][2][.][3][6][9] [π/E] [−][4][5]

```
-2.369E-45
```

2. Changing Input Contents (Correction, Deletion, Insertion)

- **Correction**

Move the cursor to the position where the correction is to be made (use [←][→]). Press the key for the desired letter, number or symbol.

Example: To change A$ to B$

```
A$_
```

Operation: Move the cursor 2 positions to the left.
[←][←]

```
A$
```

Press the [B] key.

```
B$
```

Example: To change "LIST" to "RUN"

```
LIST_
```

Operation: Move the cursor 4 positions to the left.
[←][←][←][←]

```
LIST
```

Press [R][U][N] or [F2] [RUN/V] .

```
RUN _
```

* In the above example, to move the cursor beneath the first letter, press [F1] [HOME/←] .

- **Deletion**

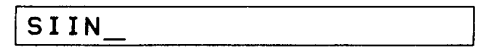
Position the cursor one space to the right of the character to be deleted. Press the [C] key. Each time it is pressed, the character to the left of the cursor will be deleted.

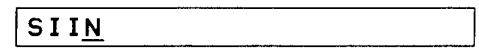
Example: To delete one "I" from "SIIN"

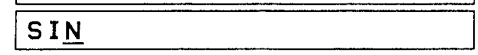
Operation: Move the cursor one space to the left.

(Use [←])

Press the [C] key.





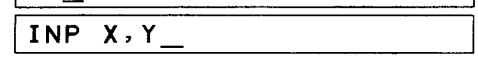


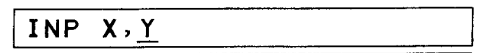
Example: To delete "X," from "INP X, Y"

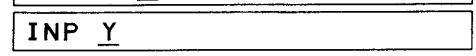
Operation: Move the cursor one space to the left.

(Use [←])

Press [C][C] . The two characters will be deleted.

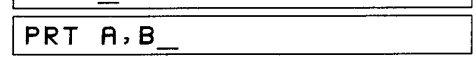


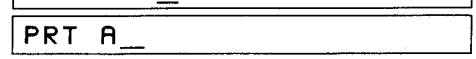




Example: To delete ",B" from "PRT A, B"

Operation: Press [C][C] . The two characters will be deleted.





- **Insertion**

Move the cursor to the right of the position where the new character is to be inserted. Press [F1][INS] . Each time they are pressed, a space will appear. Press the desired key to insert into the space.

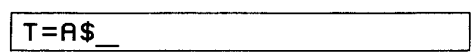
Example: To insert "$" into "T = A$" to make it "T$ = A$"

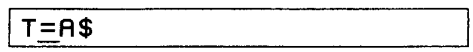
Operation: Move the cursor 3 positions to the left.

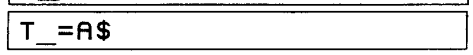
(Use [←][←][←])

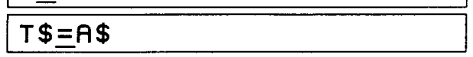
Press [F1][INS] . One space will appear above the cursor.

Press the [$] key.







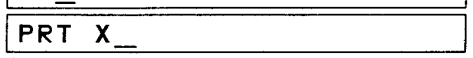


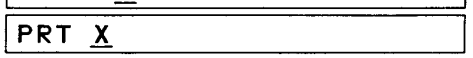
Example: To change "PRT X" to "PRT SIN X"

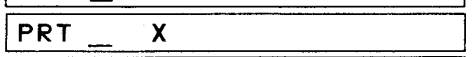
Operation: Move the cursor 1 position to the left. (Use [←])

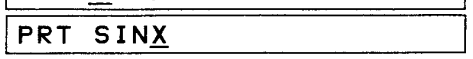
Make 3 spaces. (Use [F1][INS][F1][INS][F1][INS])

Press [S][I][N] .









2-5 Memory Expansion

There are normally 26 memories (variables). At this time there are 1680 steps. The memory can be expanded up to a maximum of 226 memories. This expansion is accomplished by converting to 10 units per memory with each having 8 steps.

Number of memories	Number of program steps
26 (A ~ Z)	1680 steps
36 (A ~ Z, A0 ~ A9)	1600 steps
46 (A ~ Z, A0 ~ A9, B0 ~ B9)	1520 steps
56 (A ~ Z, A0 ~ A9, B0 ~ B9, C0 ~ C9)	1440 steps
⋮	⋮
226 (A ~ Z, A0 ~ A9, ... , T0 ~ T9)	80 steps

DEFM command is used to expand the memory. Also, expansion is designated in units of 10 each, with 10 units forming 1 group. To expand to 10, use "1". To expand to 50, use "5".

Example: To expand the memory to 56 by increasing 30.

Operation: Performed using RUN mode (MODE 0) or WRT mode (MODE 1).

DEFM 3 EXE → `VAR: 56 PRG: 1440`

number of memories — number of steps

* DEFM can be input by pressing D E F M or F2 (DEFM) .

Example: To expand the memory to 166 by increasing 140.

Operation: **DEFM 14** EXE → `VAR: 166 PRG: 560`

* If a large number of programs are already in use, to protect those programs already written, an error message will appear if the remaining area is insufficient.

`ERR-1` ..... indicates that number of steps is insufficient.

* Since the exclusive character variable ($) is a special memory, it is not counted at the memory capacity designation.

2-6 Auto Power OFF

In order to save power when the power switch is left ON, the power will shut off automatically after approximately 8 minutes (except during program calculation). In this case, turn the power switch ON again or press the AC key and the power will come ON again.

* The memory and program contents will not be erased when the power is turned OFF. Angle mode designation or mode designation ("WRT", "TRACE", "PRT", etc) will all be cancelled.

2-7 Excess Numbers

When displaying the results of manual or program calculation, the 11th position can be rounded off or deleted. This command is performed using "SET" command.

SET { 0 / 5 } ("0" for deletion / "5" for rounding off)

Example: 11÷3=3.6666.....

Operation: To delete

S E T 0 EXE

1 1 ÷ 3 EXE → `3.666666666`

* This deletion is only for the display. The actual internal calculations are performed by rounding off.

(Continuing) ANS → `3.666666667`

- **Rounding Off**

S E T 5 EXE

1 1 ÷ 3 EXE → `3.666666667`

This SET command automatically becomes rounded off by auto power OFF or power switch OFF. Calculations in this manual will be performed using "SET 5" (rounded off).

Chapter 3

Manual Calculation

3-1 Explanation of Manual Calculation

For manual calculation, calculations are not automatically performed using previously stored formulas as in program calculation. Operations for moving numbers from right to left, calculating by hand and calling out contents of variables are called manual calculations.

3-2 Manual Calculation Operation

- Addition, subtraction, multiplication and division are by true algebraic logic. [+], [−], [×] (X), [÷] (÷) and [EXE] (=) are used. The [EXE] key is used to obtain calculation results and functions as "=".

Example: 12+36−9×5÷4=36.75

Operation: [1][2][+][3][6][−][9][×][5][÷][4]

```
12+36-9*5/4
```

[EXE]

```
36.75
```

- Function calculation includes addition, subtraction, multiplication and division and is the same as true algebraic logic. Data is written after function commands.

Example: log 1.23=0.08990511144

Operation: LOG 1[.]23

```
LOG1.23
```

[EXE]

```
0.08990511144
```

* Letters and numbers will be written in this manual without frames.

Example: [S][I][N][(][1][5][+][8][)][EXE] → SIN(15+8)[EXE]

* Function and program commands can be written using either one key commands or alphabetical commands but will be written in this manual using alphabetical commands.

- Calculations such as memory calculations used to store number values or calculation results and to obtain totals will use variables. These variables are alphabetical letters (A–Z), or letters and numbers (0–9) in combination (when the memories are expanded to more than 26). To put a number value or calculation result into a variable, use substitution.

Example: Store 1234 as variable A

Operation: A[=]1234

```
A=1234
```

[EXE]

Example: Add the result of 23 x 56 to variable K

Operation: K⊟K+23×56 | K=K+23*56

EXE |

This method is the manual way to perform the same operation as sentence substitution in programs.

- To make corrections prior to pressing the EXE key, move the cursor to the position to be corrected and press the correction key. (See Chapter 2)
- Press the AC key to cancel the entire display.

3-3 Manual Calculation Examples

3-3-1 Fundamental Calculations

■ **Addition, subtraction, multiplication and division**

Example: 23+4.5−53=−25.5

Operation: 23+4·5−53EXE | −25.5

Example: 56×(−12)÷(−2.5)=268.8

Operation: 56×(−12)/(−2·5)EXE | 268.8

(may be omitted)

Example: 12369×7532×74103=6.903680613×10^12 (=69036806 13000)

Operation: 12369×7532×74103EXE | 6.903680613E 12

Example: 1.23÷90÷45.6=2.997076023×10^−4 (=0.0002997076023)

Operation: 1·23/90/45·6EXE | 2.997076023E-04

* When the result is greater than 10^9 or less than 10^{-3}, it will be displayed exponentially.

Example: 7×8+4×5=76

Operation: 7×8+4×5EXE | 76

Example: 12+(2.4×10^5)÷42.6−78×36.9=2767.602817

Operation:

12+2·4π5/42·6−78×36·9EXE | 2767.602817

■ **Memory Calculation**

Example: 12×45=540
12×31=372
75÷12=6.25

Operation:

A⊟12EXE	
A✱45EXE	540
A✱31EXE	372
75／AEXE	6.25

Example:

23+9=32
53−6=47
−) 45×2=90
99÷3=33
Total 22

Operation:

M⊟23＋9EXE	
M⊟M＋53−6EXE	
M⊟M−45✱2EXE	
M⊟M＋99／3EXE	
MEXE	22

* Using this operation method, individual calculation results cannot be identified. To see the individual results, perform in the following manner.

23＋9EXE	32
M⊟ANSEXE	
53−6EXE	47
M⊟M＋ANSEXE	
45✱2EXE	90
M⊟M−ANSEXE	
99／3EXE	33
M⊟M＋ANSEXE	
MEXE	22

3-3-2 Function Calculations

■ **Trigonometric Functions (sin, cos, tan) and Inverse Trigonometric Functions ($\sin^{-1}$, $\cos^{-1}$, $\tan^{-1}$)**

- When using trigonometric and inverse trigonometric functions, be sure that the angular units are designated.
  (If angular units are not changed, this does not have to be accomplished again.)

Example: $14°25'36'' = 14.42666667°$

Operation: DEG (14 ,25 ,36) EXE

14.42666667

* "DEG(" can be input using either one key or alphabetical command. (same for following)

Example: $14.2536° = 14°15'12.96''$

Operation: DMS 14 .2536 EXE

14° 15' 12.96"

Example: $\sin 12.3456° = 0.2138079201$

Operation: MODE 4 → "DEG"

SIN 12 .3456 EXE

DEG 0.2138079201

(or F1 SIN 12 . 3456, same for following)

Example: $2 \cdot \sin 45° \times \cos 65°6' = 0.5954345575$

Operation:

2 × SIN 45 × COS DEG (65 ,6) EXE

DEG 0.5954345575

Example: $\sin^{-1}0.5 = 30°$ (To obtain x when $\sin x° = 0.5$)

Operation: ASN 0 .5 EXE

DEG 30

(or F1 ARC SIN 0 . 5, same for following)

Example: $2.5 \times (\sin^{-1}0.8 - \cos^{-1}0.9) = 68.22042398$

Operation:

2 .5 × (ASN 0 .8 − ACS 0 .9) EXE

DEG 68.22042398

Example: $\cos(\frac{\pi}{3}\text{rad}) = 0.5$

Operation: MODE 5 → "RAD"

COS (F1 π ÷ 3) EXE

RAD 0.5

Example: $\cos^{-1}\frac{\sqrt{2}}{2} = 0.7853981634$

Operation: ACS (SQR 2 ÷ 2) EXE

RAD 0.7853981634

Example: $\tan(-35\text{gra}) = -0.6128007881$

Operation: MODE 6 → "GRA"

TAN − 35 EXE

GRA -0.6128007881

■ Logarithmic Functions (log, ln) and Exponential Functions (e^x, x^y)

Example: $\log 1.23\ (=\log_{10} 1.23) = 0.08990511144$

Operation: LOG 1 . 23 EXE

0.08990511144

Example: $\ln 90\ (=\log_e 90) = 4.49980967$

Operation: LN 90 EXE

4.49980967

Example: $\log 456 \div \ln 456 = 0.4342944819$

Operation: LOG 456 ◢ LN 456 EXE

0.4342944819

Example: $e^{4.5} = 90.0171313$

(To get the antilogarithm of the natural logarithm 4.5)

Operation: EXP 4 . 5 EXE

90.0171313

Example: $10^{1.23} = 16.98243652$

(To get the antilogarithm of common logarithm 1.23)

Operation: 10 ↑ 1 . 23 EXE

16.98243652

Example: $5.6^{2.3} = 52.58143837$

Operation: 5 . 6 ↑ 2 . 3 EXE

52.58143837

Example: $123^{\frac{1}{7}}\ (=\sqrt[7]{123}) = 1.988647795$

Operation: 123 ↑ (1 ◢ 7) EXE

1.988647795

Example: $(78-23)^{-12} = 1.305111829 \times 10^{-21}$

Operation: (78 − 23) ↑ − 12 EXE

1.305111829E-21

Example: $2^2+3^3+4^4 = 287$

Operation: 2 ↑ 2 + 3 ↑ 3 + 4 ↑ 4 EXE

287

Example: $\log \sin 40° + \log \cos 35° = -0.2785679838$

The antilogarithm is 0.5265407845 (logarithmic calculation of sin 40° x cos 35°)

Operation: MODE 4 (DEG designation)

LOG SIN 40 + LOG COS 35 EXE

-0.2785679838

10 ↑ ANS EXE

0.5265407845

■ **Hyperbolic Functions (sinh, cosh, tanh) and Inverse Hyperbolic Functions ($\sinh^{-1}$, $\cosh^{-1}$, $\tanh^{-1}$)**

Example: $\sinh 3.6 = 18.28545536$

Operation: HSN 3 . 6 EXE → 18.28545536

(or F1 HYP (#) SIN ($) 3 . 6, same for following)

Example: $\tanh 2.5 = 0.9866142981$

Operation: HTN 2 . 5 EXE → 0.9866142981

Example: $\cosh 1.5 - \sinh 1.5 = 0.2231301602$

Operation: HCS 1 . 5 − HSN 1 . 5 EXE → 0.2231301602

Example: $\sinh^{-1} 30 = 4.094622224$

Operation: AHS 30 EXE → 4.094622224

(or F1 ARC (11) HYP (#) SIN ($) 30, same for following)

Example: What is x when $\tanh 4x = 0.88$? $x = \dfrac{\tanh^{-1} 0.88}{4} = 0.3439419141$

Operation: AHT 0 . 88 ÷ 4 EXE → 0.3439419141

■ **Other Functions ($\sqrt{\ }$, $x!$, SGN, RAN#, RND, ABS, INT, FRAC)**

Example: $\sqrt{2} + \sqrt{5} = 3.65028154$

Operation: SQR 2 + SQR 5 EXE → 3.65028154

Example: $8!\ (= 1 \times 2 \times 3 \times \cdots\cdots \times 7 \times 8) = 40320$

Operation: 8 F1 (x!) EXE → 40320

Example: Give "1" to a positive number, "−1" to a negative number, and "0" to a zero.

Operation:

SGN 6 EXE → 1

SGN 0 EXE → 0

SGN − 2 EXE → −1

Example: Random number generation (pseudo random number of $0 < \text{RAN\#} < 1$)

Operation: RAN (#) EXE → 0.904186914

Example: The result of 12.3 x 4.56 is rounded to 10^{-2} $12.3 \times 4.56 = 56.088$

Operation: RND (12 . 3 + 4 . 56 ▷ − 2) EXE → 56.1

Example: $|-78.9 \div 5.6| = 14.08928571$

Operation: ABS (− 78 . 9 ÷ 5 . 6) EXE → 14.08928571

Example: The integer portion of $\dfrac{7800}{96}$ is 81

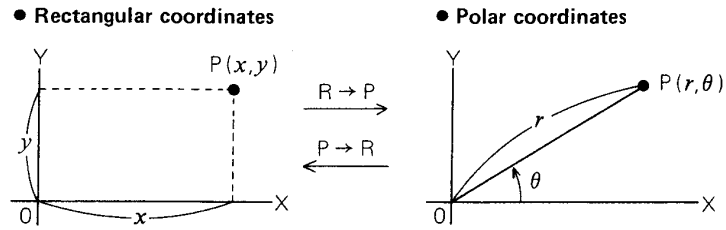
Operation: INT (7800 ÷ 96) EXE → 81

* This command will not exceed the original number value.

Example: The decimal portion of $\dfrac{7800}{96}$ is 0.25

Operation: FRAC (7800 ÷ 96) EXE → 0.25

■ Coordinate Conversion (R → P, P → R)



The θ of R → P can be obtained in the range of $-180° < \theta \leqq 180°$.
("RAD" and "GRA" are in the same range.)

Example: What are r and θ when $x=14$ and $y=20.7$?

Operation: MODE 4 (DEG designation)

RPC 14 ▷ 20 ▷ 7 EXE

(r)	X EXE	24.98979792
(θ)	DMS Y EXE	55°55'42.20"

Example: What are r and θ when $x=7.5$ and $y=-10$?

Operation: MODE 5 (RAD designation)

RPC 7 ▷ 5 ▷ (−) 10 EXE

(r)	X EXE	12.5
(θ)	Y EXE	-0.927295218

Example: What are x and y when $r=25$ and $\theta=56°$?

Operation: MODE 4

PRC 25 ▷ 56 EXE

(x)	X EXE	13.97982259
(y)	Y EXE	20.72593931

Example: What are x and y when $r=4.5$ and $\theta=2/3\,\pi$ rad?

Operation: MODE 5

PRC 4 ▷ 5 ▷ 2 ◢ 3 × F1 π EXE

(x)	X EXE	-2.25
(y)	Y EXE	3.897114317

■ **Effective Number of Units Designation and Decimal Designation**

Effective number of units designation and decimal designation are performed using "SET" command.

Effective number of units designation . . . . . SET E n (n=0 to 9)
Decimal designation . . . . . SET F n
Designation cancellation . . . . . SET N

* For effective number of units designation, "SET E 0" is a 10 unit designation.
* If designation is performed, excess number management (See page 13 will cause deletion and rounding off on the display one unit below the designated unit. The original number value will remain in the internal calculation section or in the memory.

Example: 100÷6=16.66666666.....
Operation: SET [E] 4 [EXE] (effective number of units designated as 4 units)

100 ÷ 6 [EXE] → 1.667E 01

Example: 123÷7=17.57142857.....
Operation: SET [F] 2 [EXE] (decimal designated as 2 units)

123 ÷ 7 [EXE] → 17.57

Example: 1÷3=0.3333333333.....
Operation: SET [N] [EXE] (designation cancellation)

1 ÷ 3 [EXE] → 0.3333333333

3-3-3 Statistical Calculations

- For statistical calculation, be sure to press [F1] [SAC] to clear the statistical memory, then start.

■ **Standard Deviation Calculation**

- Use the [STAT] key and press one time for each data.
  Data [STAT]
- When there is duplication of data, press the [STAT] key the desired number of times or
  data [:] quantity [STAT]
- Standard Deviation

$$\sigma_{n-1}=\sqrt{\frac{\sum_{i=1}^{n}(x_i-\bar{x})^2}{n-1}}=\sqrt{\frac{\Sigma x^2-(\Sigma x)^2/n}{n-1}}$$

Use sample data in a group and designate that group's standard deviation.

- Average

$$\bar{x}=\frac{\sum_{i=1}^{n}x_i}{n}=\frac{\Sigma x}{n}$$

Example: When data is 55, 54, 51, 55, 53, 53, 54, 52
Operation: [F1] [SAC]

55 [STAT] 54 [STAT] 51 [STAT] 55 [STAT]
53 [STAT] [STAT] 54 [STAT] 52 [STAT]

		STAT 52
(Standard deviation σ_n)	SDXN [EXE]	1.316956719
(Standard deviation σ_{n-1})	SDX [EXE]	1.407885953
(Average $\bar{x}$)	MX [EXE]	53.375
(Data n)	CNT [EXE]	8
(Sum of x Σx)	SX [EXE]	427
(Sum of x squared Σx^2)	SX2 [EXE]	22805

Example: In the following figure, what are σ_n and σ_{n-1}?

Grade Number	Grade Value	Degree
1	110	10
2	130	31
3	150	24
4	170	2
5	190	3

Operation:

F1 SAC

110 , 10 STAT

130 , 31 STAT

150 , 24 STAT

170 STAT STAT

190 STAT STAT STAT

F1 ASTAT ANS

STAT 190
CNT= 70
SX=9640
SY=0
SX2=1351000
SY2=0
SXY=0

SDXN EXE

18.2968716

SDX EXE

18.42898069

* Deletion and Correction of Erroneous Data (Correct operation: 51 STAT)
(1) 50 STAT → continue F1 DEL STAT, then operate correctly.
(2) 49 STAT (several items before) → 49 F1 DEL STAT, then operate correctly.
Deletion and correction may be accomplished in a similar manner for a number of items.
49 , 12 STAT (several items before) → 49 , 12 F1 DEL STAT, then operate correctly.

■ Regression Calculation

- Press x data , y data STAT for data.
- When there are a number of similar data pairs, press STAT key for the number of data desired or x data , y data ; quantity STAT.

■ Linear Regression Calculation

- Regression equation is $y = A + Bx$
  Coefficients A and B are calculated using the following formulas.

Regression coefficient of regression equation

$$B = \frac{n \cdot \Sigma xy - \Sigma x \cdot \Sigma y}{n \cdot \Sigma x^2 - (\Sigma x)^2}$$

Constant term of regression equation

$$A = \frac{\Sigma y - B \cdot \Sigma x}{n}$$

- Correlation coefficient r for the input data pairs is calculated using the following formula.

$$r = \frac{n \cdot \Sigma xy - \Sigma x \cdot \Sigma y}{\sqrt{\{n \cdot \Sigma x^2 - (\Sigma x)^2\}\{n \cdot \Sigma y^2 - (\Sigma y)^2\}}}$$

Example: ● Bar steel temperature and length

Temperature	Length
10°C	1003mm
15	1005
20	1010
25	1008
30	1014

Obtain the regression equation and correlation coefficient from this measured result. Calculate the length at 18°C and the temperature at 1000mm.

Operation: [F1] [SAC]

10 [▸] 1003 [STAT]	STAT 10, 1003
15 [▸] 1005 [STAT]	STAT 15, 1005
20 [▸] 1010 [STAT]	STAT 20, 1010
25 [▸] 1008 [STAT]	STAT 25, 1008
30 [▸] 1014 [STAT]	STAT 30, 1014
(Constant term A) **LRA** [EXE]	998
(Regression coefficient B) **LRB** [EXE]	0.5
(Correlation coefficient r) **COR** [EXE]	0.9190182776
(Length at 18°C) **EOY 18** [EXE]	1007
(Temperature at 1000mm) **EOX 1000** [EXE]	4

* To cancel or correct erroneous input data (correct operation: 10 [▸] 1003 [STAT])
(1) 11 [▸] 1003 [STAT] → continue [F1] [DEL STAT], then operate correctly.
(2) 10 [▸] 1030 [STAT] → continue [F1] [DEL STAT], then operate correctly.
(3) 11 [▸] 1003 [STAT] (Several items before) → 11 [▸] 1003 [F1] [DEL STAT], then operate correctly.
Quantity can also be deleted or corrected in a similar manner.
11 [▸] 1003 [:] 10 [STAT] (Several items before) → 11 [▸] 1003 [:] 10 [F1] [DEL STAT], then operate correctly.

■ **Logarithmic Regression Calculations**

● Regression equation is $y = A + B \cdot \ln x$. Data x is input as logarithm (ln) of x. Data y is input the same as linear regression.
● To obtain the regression coefficient or to correct, use the same operation as linear regression, but $\Sigma \ln x$ is required for Σx, $\Sigma(\ln x)^2$ is required for Σx^2 and $\Sigma \ln x\, y$ is required for Σxy.

Example:

x_i	y_i
29	1.6
50	23.5
74	38.0
103	46.4
118	48.9

Make logarithmic regression of this data and obtain the constant term, coefficient and correlation coefficient of the regression equation, then calculate the determining coefficient (r^2).

Operation: [F1] [SAC]

LN 29 [▸] 1 [▪] 6 [STAT]	TAT 3.36729583, 1.6
LN 50 [▸] 23 [▪] 5 [STAT]	T 3.912023005, 23.5
LN 74 [▸] 38 [▪] 0 [STAT]	TAT 4.304065093, 38
LN 103 [▸] 46 [▪] 4 [STAT]	T 4.634728988, 46.4
LN 118 [▸] 48 [▪] 9 [STAT]	T 4.770684624, 48.9
(Constant term A of the regression equation) **LRA** [EXE]	-111.1283976
(Regression coefficient B) **LRB** [EXE]	34.02014749
(Correlation coefficient r) **COR** [EXE]	0.9940139464
(Determining coefficient r^2) [↑] 2 [EXE]	0.9880637256

■ Exponential Regression Calculation

- The regression equation is $y = A \cdot e^{B \cdot x}$ ($\ln y = \ln A + B \cdot x$).
  Data y is input as logarithm (ln) of y. Data x is input the same as linear regression.
- Correction method is the same operation as linear regression but lnA is required for constant term A, $\Sigma \ln y$ is required for SUM Y, $\Sigma(\ln y)^2$ is required for SUM Y2 and $\Sigma x \cdot \ln y$ is required for SUM XY.

Example:

x_i	y_i
6.9	21.4
12.9	15.7
19.8	12.1
26.7	8.5
35.1	5.2

Make exponential regression of this data and obtain the regression equation and correlation coefficient.

Operation:

[F1] [SAC]

6 ⊡ 9 ⊡ LN 21 ⊡ 4 [STAT]	AT 6.9, 3.063390922
12 ⊡ 9 ⊡ LN 15 ⊡ 7 [STAT]	T 12.9, 2.753660712
19 ⊡ 8 ⊡ LN 12 ⊡ 1 [STAT]	T 19.8, 2.493205453
26 ⊡ 7 ⊡ LN 8 ⊡ 5 [STAT]	T 26.7, 2.140066163
35 ⊡ 1 ⊡ LN 5 ⊡ 2 [STAT]	T 35.1, 1.648658626
(Constant term A) **EXP LRA** [EXE]	30.49758742
(Coefficient B) **LRB** [EXE]	-0.04920370831
(Correlation coefficient r) **COR** [EXE]	-0.9972473519

■ Power Regression Calculation

- The regression equation is $y = A \cdot x^B$ ($\ln y = \ln A + B \ln x$)
  Both x and y data are input as logarithm (ln).
- Correction method is the same operation as linear regression but lnA is required for constant term A, $\Sigma \ln x$ is required for Σx, $\Sigma(\ln x)^2$ is required for Σx^2, $\Sigma \ln y$ is required for Σy, $\Sigma(\ln y)^2$ is required for Σy^2 and $\Sigma(\ln x, \ln y)$ is required for Σxy.

Example:

x_i	y_i
28	2410
30	3033
33	3895
35	4491
38	5717

Make power regression of this data and obtain the regression equation and correlation coefficient.

Operation:

[F1] [SAC]

LN 28 ⊡ LN 2410 [STAT]	3220451, 7.787382026
LN 30 ⊡ LN 3033 [STAT]	1197382, 8.017307508
LN 33 ⊡ LN 3895 [STAT]	6507561, 8.267448958
LN 35 ⊡ LN 4491 [STAT]	5348061, 8.409830673
LN 38 ⊡ LN 5717 [STAT]	3758616, 8.651199471
(Constant term A) **EXP LRA** [EXE]	0.2388010829
(Coefficient B) **LRB** [EXE]	2.771866148
(Correlation coefficient r) **COR** [EXE]	0.9989062562

Chapter 4

Program Calculation

This instrument uses BASIC as its program language. BASIC is an abbreviation for Beginner's All-purpose Symbolic Instruction Code. Also, it is said that it is a fundamental language system which is easy for beginners to use.

BASIC Language Features

1. It is a problem solving language and programming efficiency is improved.
2. Not only is it a program language which is not limited to a particular field, but is a general use language which is applicable to many fields, such as natural science, social science and business fields.
3. It is a conversation type language and the computer user can compose programs while conversing with the computer and use it for making entries.
4. It has many of the capabilities and features of FORTRAN but is free from the rules encountered when using FORTRAN.
5. Many fundamental built-in functions have been prepared and it has an expanded calculation range.

4-1 Program Outline

Program calculation **1)** programs calculation contents to be executed, **2)** calculator stores programs, and **3)** using these programs. Simply input the data and the results can be obtained automatically.

■ Programming Fundamentals

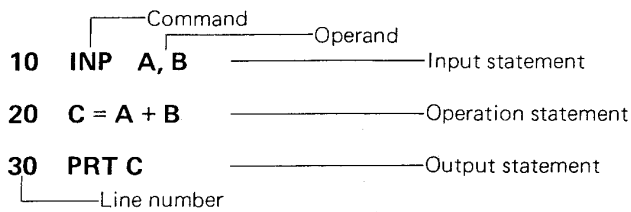
Let's take a look at the programming necessary to use the computer for a given problem, and at the concept and programming procedures.

- **Programs and Programming**

When the operator uses the computer to manage a problem, instructions must be used which are in words that the computer can understand. These instructions are called programs and the composing of these instructions is called programming.

- **What is a program?**

In order to make a program, there are many grammatical rules, but these details will be explained later. First, to discuss what a program is and its form, let's take a look at an example of a fundamental program.



The above program is a fundamental program and consists of an input statement, an operation statement, an output statement and line numbers. That is to say, the input statement inputs data. According to that data, the operation statement performs various operations, and the output statement outputs the execution results. Also, each line has a line number preceding it. These operation statements are not limited to one but are performed several times and, by adding decision statements, the program becomes long and complicated. Nevertheless, they are fundamentally the same.

Also, on one line, following the line number, a COMMAND is written which tells the calculator what to perform next. It is composed of alphabetical characters. It is followed by OPERANDS which indicate the necessary information for the command.

The above is called a PROGRAM and is in the fundamental form.

● **Number of program steps**

Program steps are counted as follows.

1) Program command . . . . . 1 step/1 command
2) Functional command . . . . . 1 step/1 command
3) Line number . . . . . 2 steps/1 line number
4) Character . . . . . 1 step/1 character
5) Pressing the [EXE] key after each line number's key input to store in the calculator. . 1 step

Example:

1 (2) INP (1) A (1) [EXE] . . . . . 5 steps

10 (2) B (1) = (1) SIN (1) A (1) [EXE] . . . . . 7 steps

100 (2) PRT (1) " (1) B (1) = (1) " (1) ; (1) B (1) [EXE] . . . . . 10 steps

Total 22 steps

Note: Assigning a password requires 6 steps. (See page 74.)

■ Programming Sequence

Programming progresses in the following sequence.

1) Problem analysis
2) Flow chart preparation
3) Writing of the program on a coding sheet
4) Debugging execution

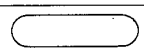
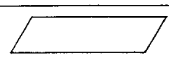
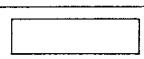
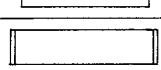
Explanation

Step 1: Analyze the given problem to determine the required steps for solving.
Step 2: Write a flow chart to represent the flow of logic for the solution of the problem. The flow chart is composed of symbols which show processing and decision making elements.
Step 3: Based on the flow chart, write a program on coding sheets or similar forms using BASIC language.
Step 4: Check the program for mistakes.

The above is the procedure for composing a program.

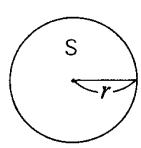
■ Flow Chart

The most widely used flow chart symbols are written below.

Symbol	Nomenclature	Meaning
	Terminal	Initiate, terminate, etc.
	Input/Output	Input/output functions.
	Process	Numerous processing functions.
	Predefined process	A group of commands defined elsewhere, such as a subroutine.
	Decision	Decision on the choice of a specific route from among several possible routes.

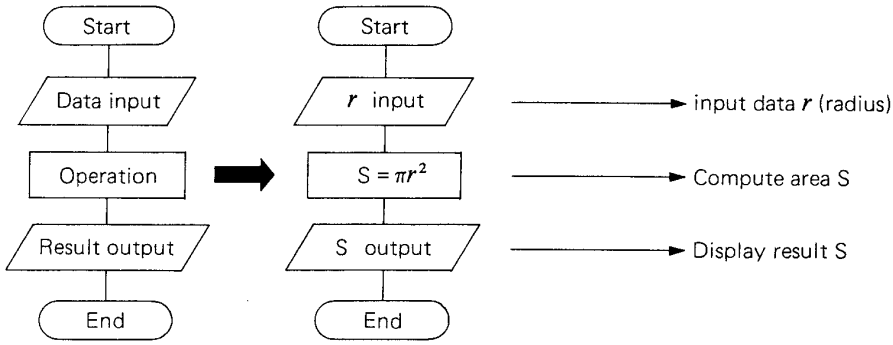
● **Flow Chart Example**

Here, let's take a look at a simple example of a program for determining the area of a circle.



$S = \pi r^2$

First, following programming fundamentals, consider the data input, operation and result output separately as shown below.



The flow chart is prepared in this manner following the program flow and is composed viewing the program in its entirety.
When programming, make it a habit to compose a flow chart. If more complex programs are created, confusion can be avoided.

■ Coding

Coding is the process of writing programs on coding sheets or similar forms after program assembly.
When assembling programs, certain operating symbols are required. The simplest fundamental operating symbols are written below.
The four fundamental arithmetic operation symbols
+ and − are expressed as " + " and " − "
x and ÷ are expressed as " * " and " / "
Powers, such as x^2 and x^3, are expressed as "x↑2" and "x↑3".
The " = " used in assignment statements will be explained in greater detail later but is explained briefly here. For example, the " = " in $S = \pi r^2$ means to assign the calculation result of πr^2 to S as opposed to having the meaning of equal as in mathematics.
Let's write a program to determine the area of the previously specified circle.

1) Write an input statement to input data *r*.

There are many input commands but usually the "INP" command is used to input data from the keyboard during program execution. The input statement used to input data *r* becomes INP R. If a line number is added to the input statement to make it complete, it becomes 10 INP R.
These line numbers can be in the range 1 to 9999 but identical line numbers cannot be used in the same program. Otherwise, the later line number will take priority. Usually, it is more convenient to assign line numbers in increments of ten for easy addition, correction and deletion. If the program is started, the statements are executed in smaller to larger line number sequence. So, assign the line numbers in program execution sequence.

2) Next, make an assignment statement to assign the result of the calculation from the input data to S.

Since πr^2 means $\pi \times r^2$
then **S = π*R↑2** (X is represented by * and R^2 is represented by R↑2)
Therefore, with the line number attached as usual, it becomes
20 S = π*R↑2

3) Write an output statement to output (display) the result of the operation.

Since this is just for calculation and does not cause a display, use "PRT" command to display the result.
The output statement to output (display) result S becomes
PRT S
And, adding a line number to this, it becomes
30 PRT S
So, the complete program for determining the area of a circle is:

```
10 INP R
20 S=π*R↑2
30 PRT S
```

For program coding, it is not necessary to use the exclusive coding paper but, if it is used, it will be more convenient for problem analysis or writing flow charts in a standard format after composing.

4-2 Program Fundamentals

■ Constants and Variables

Characters used for BASIC are upper case letters of the alphabet (A to Z) and numbers (0 to 9) as well as some special characters (such as symbols).

• Constants

A constant is a fixed value which can be written directly into a program.

Example: In the expression $S=\pi r^2$, it becomes S=π✳R↑2 and 2 is the constant.

• Variables

A variable is a value that is used in a program but is input from the keyboard during execution and is used when the result of a calculation is unknown initially because the result is assigned during execution.
A variable consists of one uppercase letter (A to Z) or a combination of one upper case letter and one single digit number (A0, A1 to A9, B0 to T9 . . . when memory is expanded). Assignment can be made from within this range.

Example: In the expression $S=\pi r^2$, which becomes S=π✳R↑2, R is the variable.

Example:

```
Y = 2 ✳ X ↑ 2 + 3 ✳ X + 4
    |   |   |   |   |   |
V   C   V   C   C   V   C
```

V: Variable

C: Constant

That is, algebraics and constants used in mathematics correspond to variables and constants, respectively.
Also, in addition to the above, there are character constants and character variables. Character constants are strings of characters which are written in directly, such as "ABC" and "END", and enclosed in quotation marks and spelled out. Character variables are not a numeric value but variables which accept a character string. Each time a character string is given the contents will change.
A character string is composed of characters enclosed in quotation marks, such as "123" and is not a numerical value. Thus, "123" is just the numbers 1 and 2 and 3 written in sequence and is considered the same as "ABC" in quotation marks.
Character variables are general variables (such as A, B, X and Y) which have a $ sign attached. Selections can be made from within this range.

Example: A$, B$, C$, X$, Y$

Character variables can be compared or added to one another but other operations (such as subtraction, multiplication and division) cannot be performed.

Example: If A$ = "123" and B$ = "456"
Using C$ = A$ + B$
C$ becomes "123456"
(If C$ = B$ + A$, C$ becomes "456123")

A character string of up to 7 characters can be inserted into this character variable.
Also, besides these character variables, there are exclusive character variables. Exclusive character variables are displayed using a $ sign. 30 characters can be inserted in the character string.

Example: $ = "1234567890ABCDEFG"
These exclusive character variables can use character functions (MID functions) which will be explained later. These can be used more conveniently than other character variables.

■ Assignment Statements

BASIC assignment statements are in the form shown below.

Variable = Numeric Expression

In BASIC assignment statements, an expression having arithmetic operations (+, −, ✱, /) on the right side is called a numeric expression.

Example: For Y = 2✱X+3, the 2✱X+3 on the right side is a numeric expression.
This " = " does not mean "equals", it means "assign".

Example: For Y = 2✱X+3, the left side is the variable and the right side is the numeric expression. Thus, it does not mean, as in usual mathematics, that the Y on the left side and the 2✱X+3 on the right side are equal. It means to assign the result of 2✱X+3 to Y. (Y =2✱X+3 can be better understood if it is thought of as Y ← 2✱X+3)

Example Assignment Statements

A = B . . . . . Assign the value of B to A (the former value of A is deleted).
N = 2*M . . . . . Assign double the value of M to N.
X = Y + Z . . . . . Assign the result of the sum of Y and Z to X.
I = I + 1 . . . . . Assign the value of I + 1 to I (the number value in I will be increased by 1).

4-3 Program Writing and Execution

■ Program Writing

Storing a program in the calculator's memory system is called program writing.
This operation is performed by keying in inputs from the keyboard.

1. WRT mode designation
2. Program area designation
3. Program input by line units (writing)
4. The program area can be divided into 10 parts from P0 to P9. Programs are written somewhere in this program area.

(1) WRT Mode Designation

Program writing is performed in the WRT mode.
Press [MODE] [1]. "WRT" will be displayed.

(2) Program Area Designation

To designate a program area, press the [F1] key then a [0] to [9] numerical key.

[F1] [P0 0] → **P0**　　[F1] [P5 5] → **P5**
[F1] [P1 1] → **P1**　　[F1] [P6 6] → **P6**
[F1] [P2 2] → **P2**　　[F1] [P7 7] → **P7**
[F1] [P3 3] → **P3**　　[F1] [P8 8] → **P8**
[F1] [P4 4] → **P4**　　[F1] [P9 9] → **P9**

(3) Program Input (Writing)

Program writing is performed in line units. Up to 62 characters, including the line number, can be written. Finally, press the [EXE] key.

*** The Role of the [EXE] Key.**

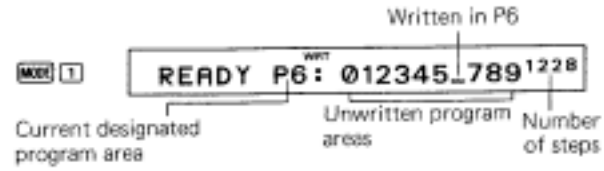
The [EXE] key is pressed to write programs, input data and to obtain the result of manual calculations. For program writing, press the [EXE] key after each line number's key input to store in the calculator. Program writing or written program content change, addition and deletion are all followed by pressing the [EXE] key as the final step. If the contents of the display are changed and the [EXE] key is not pressed, the written in contents will not be changed.

Example: Write the following program to P0

```
10 INP A,B
20 V=A+B
30 W=A-B
40 PRT V,W
50 END
```

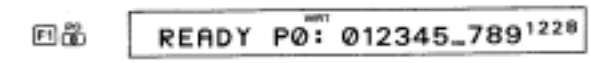
Operation:

1. **Designate WRT mode.**

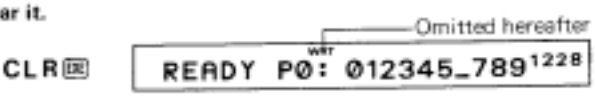


* Number of steps will be changed by the number of memories or amount of programs written.

2. **Designate program area P0**



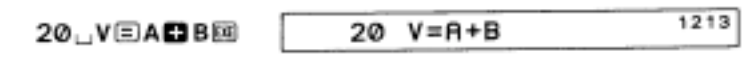
3. **If a previously written program remains, clear it.**



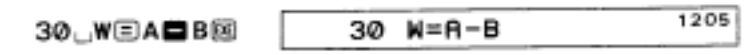
(When nothing is written, omit this step.)

4. **Write line number 10.**

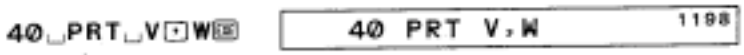
5. **Write line number 20.**



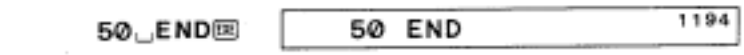
6. **Write line number 30.**



7. **Write line number 40**



8. **Write line number 50**



- A space is written between line numbers and commands and operands to make it easier to read the display. In BASIC language, except messages such as PRT statement, it has no special meaning.
- In this case, line numbers are input in units of 10. Line numbers can be freely selected from within a range of 1 to 9999. Selecting in units of 10 makes it easy to make additions and insertions later. Program execution is performed in line number sequence from small numerical value, so attach line numbers in program run sequence.

■ Program Execution

Program execution is performed in the RUN mode (press MODE 0. . . "RUN" will be displayed).
There are two methods for executing written programs.

(1) Program Execution Methods

1. Execution by program area designation

For this method, execution is begun at the same time as program area designation.

F1 { PO 0 / ⟆ / P9 9 } (Press one of the PO 0 to P9 9 keys after pressing F1)

Example: Start the program specified in the previous example.

Operation:

RUN mode (omitted hereafter)

F1 PO 0

RUN
?

* This "?" is because the INP statement is written at the beginning.

2. Execution by RUN command

RUN EXE (RUN can be input by pressing R U N or F2 V RUN)

?

* Continuing from the previous example, a "?" is displayed. In an input waiting situation, cancellation will not be effected by AC. After pressing MODE 0, operation 2 is performed.
Also, to execute along the way, after the RUN command, input the line number and press the EXE key.

Example: Start from line 20.

Operation: **RUN 20** EXE

* For method 1, it is not necessary to designate the program area to be executed. However, for method 2, it is necessary to designate the program area to be executed. (If the program area is different, the program written in that program area will be executed.)

(2) Key Input during Program Execution

Key input during program execution uses the INP statement and KEY function. Key input by the KEY function is 1 key input only but even when no key is input, execution continues.
For key input using the INP statement, a "?" is displayed and there is a pause awaiting input. After data input, execution is started by pressing the EXE key.

Example: Execute the program written in P0 in the previous example.

Operation: Execute program

F1 PO 0

?

- For this program, 2 variables are input. First, the value of variable A is input.

47 EXE

?

- Next, the value of variable B is input.

69 EXE

116

CONT

-22

In this manner, for key input during execution using the INP statement, input data using data EXE. Also, when awaiting input using the INP statement, if the EXE key is pressed directly without inputting a number, "STOP" will be displayed and a stop situation will occur. At this time, other operations such as manual calculation can be performed. To continue the stopped program, press the CONT key.
In addition, in a waiting input situation, to stop program execution, if EXE AC are pressed directly, a stop will occur.

4-4 Program Edit

- Program edit is used to make a program logically correct and makes it possible to make changes, additions and deletions or to accomplish line number rewrite.
- Program edit is performed using the LIST command to call out each line.
- LIST command is usable in both the RUN mode and the WRT mode but if the RUN mode is used, program contents will be displayed. If the WRT mode is used, program edit can be performed.

(1) Display of the Program List Using the RUN Mode.

(Display lasts about 2 seconds)

Operation: LIST EXE

(LIST can be L I S T or F2 W/LIST)

```
10 INP A,B
20 V=A+B
30 W=A-B
40 PRT V,W
50 END
```

Furthermore, if the list is not needed from the beginning, designate the line number.
To list from line number 30:

Operation: LIST 30 EXE

```
30 W=A-B
40 PRT V,W
50 END
```

* During the LIST command execution, the display will be made in sequence until the end. Press the STOP key to stop.
  Also, to continue the stopped LIST command, press the CONT key.

(2) Program Change, Addition and Deletion in the WRT Mode.

1. Changing

Using the LIST command, each line will be displayed in sequence from the designated line number each time the EXE key is pressed. Also, if the designated line number is omitted, the display will start from the first line.

a. Partial change

Example: Change the "+" to "*" on line 20 in the previous example.

Operation:

- In case the program area is not designated at P0, designate P0.

F1 P0

```
READY P0: _12345_789 1194
```

- Call out line number 20 using the LIST command.

LIST 20 EXE

```
20 V=A+B_                1194
```

- Move the cursor and set it at the location of the desired change (i.e. "+").

← ←

```
20 V=A+B                 1194
```

* If a cursor movement key (← →) remains pressed for more than 1 second, fast movement can be achieved.

- Make the correction.

* EXE

```
30 W=A-B_                1194
```

* Be sure to press the EXE key. If it is not pressed, only the display will be changed and the written program contents will not be changed.

- At this time, line 30 will be changed. Press the AC key to clear the display and the change is complete.

AC

```
-                    1194
```

* Operation of other keys at a line where no change is required causes them to be written in on that line. Therefore, do not press any keys other than EXE and AC.

- Use LIST to make sure of the change.

MODE 0
LIST EXE

```
READY P0
  10 INP A,B
  20 V=A*B
  30 W=A-B
  40 PRT V,W
  50 END
```

b. Changing an entire line

Input the line number to be changed. (In this manner, the previously input line number is cleared.)

Example: Change "W=A−B" to "W=V/2" on line 30.

Operation: MODE 1

```
READY P0: _12345_789 1194
```

- Write the new line 30.

30 ␣ W = V / 2 EXE

```
30 W=V/2             1194
```

- Check the program list.

MODE 0
LIST EXE

```
READY P0
  10 INP A,B
  20 V=A*B
  30 W=V/2
  40 PRT V,W
  50 END
```

2. Addition

To add line units, use a line number that falls between the line numbers where the addition is desired.

Example: Add "U=V*2" between line 30 and line 40 in the previous example and change line 40 to "PRT V , W , U".

Operation: MODE 1

```
READY P0: _12345_789 1194
```

- To input between line 30 and line 40, input line number 35.

35 ␣ U = V * 2 EXE

```
35 U=V*2             1186
```

* Select a line number from 31 to 39 for input between line 30 and line 40.

- To change line 40, call it out using the LIST statement and add " , U".

LIST 40 EXE
▷ U EXE
AC

```
40 PRT V,W           1186
50 END               1184
-                    1184
```

● Check the list to make sure of the program additions.

MODE 0

LIST EXE

```
READY P0
  10 INP A,B
  20 V=A*B
  30 W=V/2
  35 U=V*2
  40 PRT V,W,U
  50 END
```

3. Deletion

a. Partial deletion

Example: Delete "V , " from line 40 in the previous example.

Operation: MODE 1

```
READY P0: _12345_789 1184
```

● As in the partial change method, call out line 40 using the LIST statement.

LIST 40 EXE

```
40 PRT V,W,U_                1184
```

● Move the cursor and set it below the W (to the right of the desired deletion position).

◁ ◁ ◁

```
40 PRT V,W,U                 1184
```

● Use the C key to delete "V , ".

C C

```
40 PRT W,U                   1184
```

EXE

```
50 END                       1186
```

* If the EXE key is not pressed, the program contents will not be changed.

AC

```
_                            1186
```

* Be sure to press the AC key to cancel the line number 50 change situation.

● Check the list to make sure the deletion was accomplished properly.

MODE 0

LIST EXE

```
READY P0
  10 INP A,B
  20 V=A*B
  30 W=V/2
  35 U=V*2
  40 PRT W,U
  50 END
```

b. Deletion of an entire line

Input the same line number as the one to be deleted and that entire line will be deleted.

Example: Delete line 30.

Operation: MODE 1

```
READY P0: _12345_789 1186
```

● Input the line number to be deleted, i.e. 30.

30 EXE

```
_                            1194
```

● Make sure of the deleted position.

MODE 0

LIST EXE

```
READY P0
  10 INP A,B
  20 V=A*B
  35 U=V*2
  40 PRT W,U
  50 END
```

4. Line number correction

Example: The following program is written in P2.

```
10 INP N
20 M=N↑2
30 L=N↑0.5
40 PRT M,L
50 END
```

Move line number 20 between line 30 and line 40.

Operation: MODE 1 F1 P2

```
READY P2: _1_345_789 1160
```

- Call out line number 20 using the LIST command.

LIST 20 EXE

```
20 M=N↑2_                1160
```

- Move the cursor beneath the 2 on line 20.

F1 DEL INS → →

```
20 M=N↑2                 1160
```

- Change the 20 to read 35 and input.

35 EXE

```
30 L=N↑0.5               1152
```

- To complete the change, press AC and cancel the change command.

AC

```
_                        1152
```

- Use LIST to see how the program contents were changed.

MODE 0

LIST EXE

```
READY P2
  10 INP N
  20 M=N↑2
  30 L=N↑0.5
  35 M=N↑2
  40 PRT M,L
  50 END
```

- At this time, the contents of line 20 were moved between line 30 and line 40, but line 20 still remains. So, delete the unneeded line.

MODE 1

20 EXE

```
READY P2: _1_345_789 1152
_                        1160
```

- This completes the line number move. Check the results using LIST.

MODE 0

LIST EXE

```
READY P2
  10 INP N
  30 L=N↑0.5
  35 M=N↑2
  40 PRT M,L
  50 END
```

■ Program Debug

(1) Program Debug System

This equipment's debug system is divided roughly into desk top debug and conversation type debug via display.

Debug Systems

- **I. Desk Top Debug**
  - **a. Complete Dubug**
    Program logic structure check.
  - **b. Partial Debug**
    Program line unit check.
- **II. Conversation type debug via display**
  Using the display, check program execution flow and errors in BASIC language composition using the automatic checking mechanism of the calculator.

The desk top debug is executed during programming.
Here the explanation of the conversation type debug via the display will be made.

(2) Conversation Type Debug

Any error made during program execution stage will be given on the display using an error message. These errors are displayed in line units and display the type of BASIC language error. Based upon the error message on the display, debug is accomplished by conversation using the manual.
Furthermore, for the meaning of the error messages, see Page 79 for the error message list.

Example:

```
10 INP X
20 Y=X↑2+3*X+15
30 PRT Y
40 END
```

Y=X↑2+3X+15 was entered on line 20 in this program by mistake.

Operation: If this program is executed, "?" is displayed using the line 10 INP statement.

RUN [EXE]

```
?
```

- At this time, input 45

45 [EXE]

```
ERR-2 IN P0-20
```

- This error message indicates that a statement structure error occurred at line 20 and the program contents must be confirmed.

[AC] [MODE] [1]
LIST 20 [EXE]

```
READY P0: _123456789
20 Y=X↑2+3X+15_
```

- Since "X" is missing between "3" and "X" on line 20, correction is made using the program edit method.

[←] [←] [←] [←] [F1] [INS/C]
[*] [EXE]

```
20 Y=X↑2+3_X+15
30 PRT Y
```

(3) Program Debug during Program Execution

Conversation type debug is debug that is performed using the information obtained from the calculator by an error message. However, if an error is not displayed but the calculation result is not as expected, repeat program execution and conduct debug by confirming the calculations along the way.
This method uses the STOP command to stop program progress and uses the TRACE mode to debug line by line.

- **Debug using the STOP command**

Example: The following program is written.

```
10 Y=0
20 INP N,X
30 FOR I=1 TO N
40 Y=Y+X↑2
50 NEXT I
60 PRT Y
70 END
```

To see the value of Y in the FOR·NEXT loop, see the result of each loop using the STOP statement.

Operation: The best place to input the STOP statement is right after the calculation formula, so write the STOP statement between line 40 and line 50.

[MODE] [1]	READY P0: 123456789
45 STOP [EXE]	45 STOP

- By doing this, the program progress is stopped after the completion of the calculation on line 40 and debug can be performed.

[MODE] [0]	READY P0
RUN [EXE]	?
4 [EXE]	?
87 [EXE]	_

Cursor blinks

- What is the value of Y at the time of this stop?

Y [EXE]	7569

- When the program starts again, it will stop at the next STOP statement and request the value of Y again.

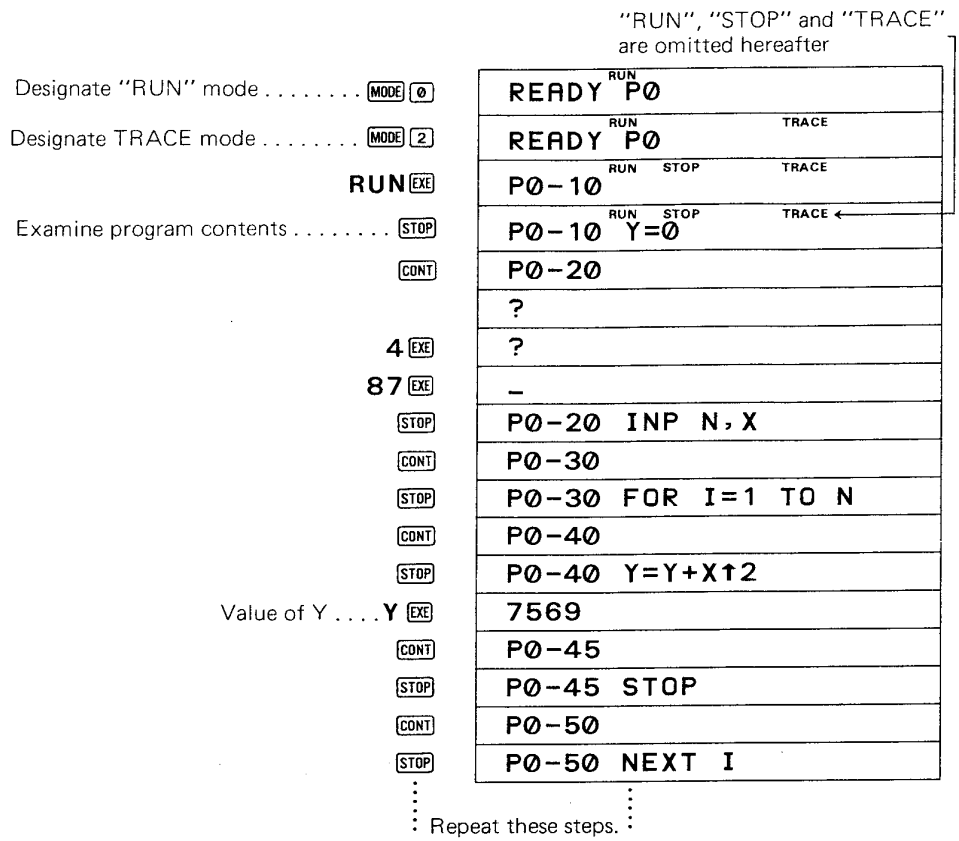
[CONT]	_
Y [EXE]	15138

- By repeating this operation, the calculation process can be seen.
  This example uses a simple program but when a complicated program is actually composed, it is very difficult to check the calculation process using desk top debug. So, if the check of variable is performed using this kind of STOP statement, program errors can be seen and corrected.

● Debug using TRACE mode

If program execution is performed using the TRACE mode (press MODE 2), it will execute each line and then stop and execution debug can be performed easily. Using the previous STOP command, let's perform a sample debug using the TRACE mode.

Operation:



This debug using the TRACE mode is the most proper method for examining the overall flow and is convenient for finding out where errors have occurred.

4-5 Program Commands

4-5-1 Jump and Loop

■ Jump Commands

Jump commands can be broadly separated into two commands. One is called a GOTO statement and is an unconditional jump. The other is a conditional jump combined with an IF statement.

● GOTO Statement

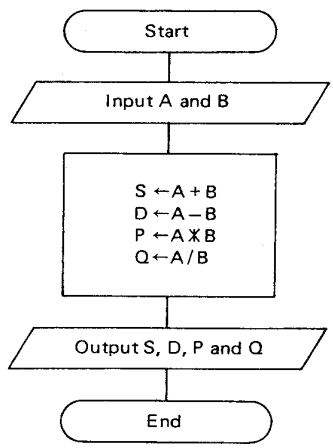
The GOTO statement is called an unconditional jump because it is a command that unconditionally advances the program to a designated location (line number).

Example 1: Add a GOTO statement in the program to determine the sum, difference, product and quotient of the data A and B.

First, the fundamental program is shown below.

```
10 INP A,B
20 S=A+B
30 D=A-B
40 P=A*B
50 Q=A/B
60 PRT S,D,P,Q
70 END
```

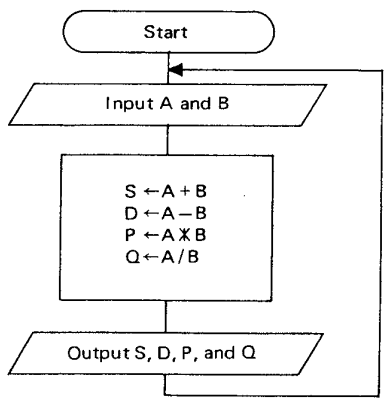
Flow Chart



Whenever this is executed, it must be allowed to "RUN" to insert data repeatedly. So a GOTO statement is input at line 70 instead of END so control will return to the proper location where the data is to be input. (Line 10 in this program)

This GOTO statement (GOTO 10) unconditionally jumps to line 10. The program is shown below.

```
10 INP A,B
20 S=A+B
30 D=A-B
40 P=A*B
50 Q=A/B
60 PRT S,D,P,Q
70 GOTO 10
```



In this program, once the data is input, it goes to an await input stage (? is displayed) and the next data can be input right away.

Execution: Input data 15 and 3, and 903 and 43.

Operation:

RUN [EXE]	?
15 [EXE]	?
3 [EXE]	18
[CONT]	12
[CONT]	45
[CONT]	5
[CONT]	?
903 [EXE]	?
43 [EXE]	946
[CONT]	860
[CONT]	38829
[CONT]	21

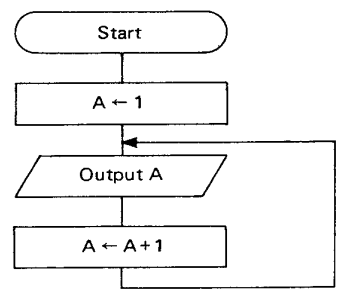
For this program, using the GOTO statement, which is input at the end, it returns to line number 10 "INP A, B". Also, as in this example, if a line number is written after GOTO, it will jump to the designated line. However, instead of a line number, if "#" and "0 to 9" are written, it will jump to a designated program area.

Example:

```
GOTO 10 (Jump to line 10 and execute from line 10)
GOTO #5 (Jump to program area P5 and execute P5 program)
```

Example 2: Make a program to increase the value of A by increments of 1.

```
10 A=1
20 PRT A
30 A=A+1
40 GOTO 20
```



Explanation:

Since A is increased in value sequentially in increments of 1, it is necessary to assign an initial value of 1 to A. That is, "A=1" on line 10.

Next, PRT (print) A

At line 30, the result when 1 is added to A is assigned to A. That is, "A=A+1". Then since the program process causes a jump using the GOTO statement, it is necessary to jump to line 20 instead of returning to the beginning. So it becomes GOTO 20.

In this manner, the GOTO statement causes the program process to jump unconditionally to a designated location.

Note: When using the GOTO statement, it is necessary to be sure to designate the correct destination line number. If the line number is omitted from the GOTO statement, an error will occur.

PRT Statement Application

The PRT statement used in this program changes the manner of display in accordance with the delimiter designations following the operands.

First, for example 1, S, D, P and Q are separated by a "," (comma).

Display stops after S. To display the D, press the [CONT] key. That is, if items are separated by a comma, the results will be displayed one at a time. If the "," is a ";", what will the display be like?

The result will be as follows.

Operation:

RUN [EXE]	?
15 [EXE]	?
3 [EXE]	18 12 45 5
[CONT]	?
903 [EXE]	?
43 [EXE]	946 860 38829 21

The " ; " in this case (semicolon) operates when there are several results and causes everything to be displayed sequentially.
Furthermore, on the display, one space appears after each result. The "+" is not displayed at the sign column but is actually present.

␣18␣12␣45␣5
└──┴──┴──┴──sign column

Also, if example 2 is executed as is, the result will be as shown below.

Operation:

RUN [EXE]	1
[CONT]	2
[CONT]	3
[CONT]	4
⋮	⋮

However, if " ; " is input following "PRT A" on line 20,

20 PRT A;

the following result will be obtained.

RUN [EXE]	1 2 3 4 5 6 7 8 9 1
	1 2 3 4 5 6 7 8 9 10
	2 3 4 5 6 7 8 9 10
	2 3 4 5 6 7 8 9 10 1
	3 4 5 6 7 8 9 10 11
	⋮

Therefore, continuous results are displayed by using " ; ". Furthermore, there is a "WAIT" command for stopping the display for a certain period for PRT statement display stop. Refer to Page 59 for details.

- **GOTO Statement Indirect Designation**

The previous GOTO statement designated the jump designation directly and caused jump. However, the indirect designation determines the jump destination according to the value of the variable. It is used when the jump destination cannot be written at the beginning of the program as when the operation method is understood from the data.
Indirect designation consists of;

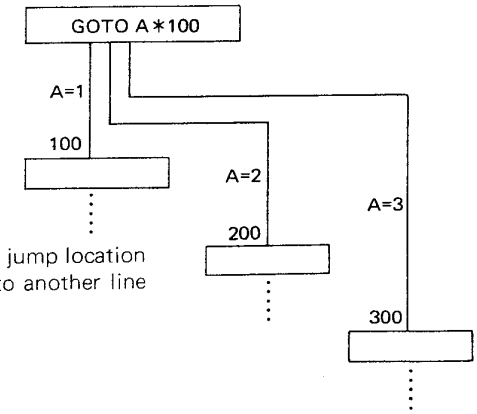
GOTO variable or numeric expression

GOTO # variable or numeric expression

This uses the value of the variable (A,B,X,Y, etc.) or the numeric expression (A+B, X+10, etc.) to determine which line number to jump to or which program area to jump to.

Example 1: GOTO A✱100

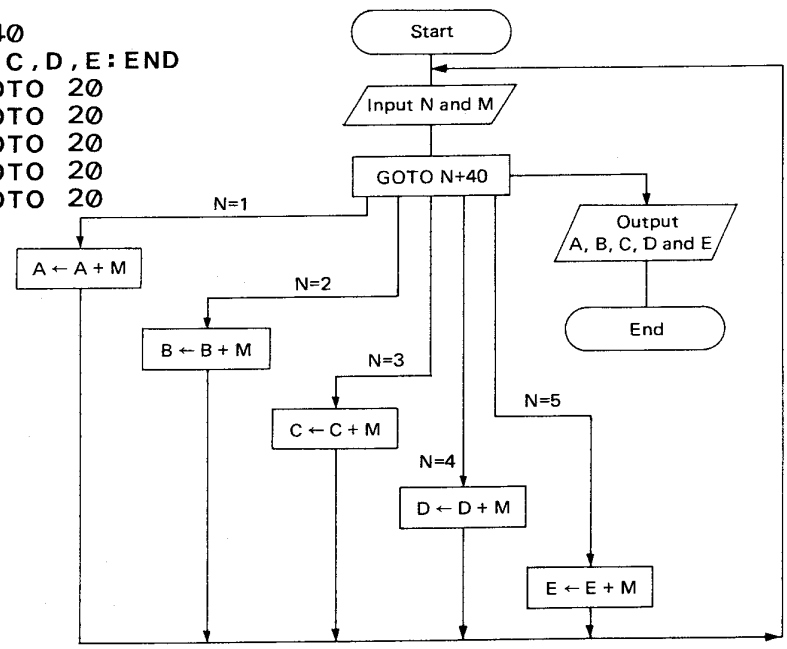
In this program, when A is 1, 2 or 3
For A=1, GOTO 100 means jump to line 100
For A=2, GOTO 200 means jump to line 200
For A=3, GOTO 300 means jump to line 300



If A is something other than 1, 2 or 3, then the jump location is unknown. This will cause an error or jump to another line number.

Example 2: Make a sorted totals program using indirect designation. (Classified here into 5 divisions)
The program will be as shown below.

```
10 VAC
20 INP N,M
30 GOTO N+40
40 PRT A,B,C,D,E:END
41 A=A+M:GOTO 20
42 B=B+M:GOTO 20
43 C=C+M:GOTO 20
44 D=D+M:GOTO 20
45 E=E+M:GOTO 20
```



The "VAC" command on line 10 is a command for clearing the data use memory (makes it 0). In this program, clearance in advance is necessary in order to total the data input on lines 41 to 45. At the next INP statement, code number (N) and amount of earnings are input.
Using line 30 GOTO statement, if the code number (N) is 1, jump to line 41, that is to say, the total earnings of code number 1.
In the case of code number 2, jump to line 42. In this manner, earnings (M) will be divided into code numbers 1 to 5.

N=1 : GOTO 41→41	A=A+M : GOTO 20 Execute
N=2 : GOTO 42→42	B=B+M : GOTO 20 Execute
N=3 : GOTO 43→43	C=C+M : GOTO 20 Execute
N=4 : GOTO 44→44	D=D+M : GOTO 20 Execute
N=5 : GOTO 45→45	E=E+M : GOTO 20 Execute

Therefore, when N is 0, it will jump to line 40. A, B, C, D and E sorted results are displayed and terminated. Furthermore, for this program, N must be input as 0 to 5. Any other number will cause an error. Let's demonstrate below using actual values.
Receipts are not in sequence so input them in sequence and sort the totals (within the 5 divisions).

Code	Earnings
3	2870
2	1960
5	3850
5	1250
1	2500
2	2310
3	1850
5	4370
3	5360
1	2220
2	1450
4	6120
1	3100

Code	Earnings
1	7820
2	5720
3	10080
4	6120
5	9470

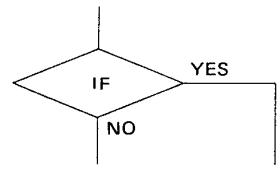
Operation:

RUN [EXE]	?
3 [EXE]	?
2870 [EXE]	?
2 [EXE]	?
1960 [EXE]	?
⋮	⋮
1 [EXE]	?
3100 [EXE]	?
0 [EXE]	?
0 [EXE]	7820
[CONT]	5720
[CONT]	10080
[CONT]	6120
[CONT]	9470

* The " : " used in lines 40 to 45 in this program is called "multistatement". It allows different commands to be written on one line. If commands are continuously performed, line numbers can be omitted and memory can be saved. Using this multistatement, many commands can be combined. However, the number of characters that can be written on one line, including the line number, is limited to 62 characters.

- **IF Statement**

An IF statement is called a conditional jump due to its nature. It executes its operation only when certain conditions are satisfied and is a command to jump to a designated location.
If this is written in a flow chart, it will take the following form.

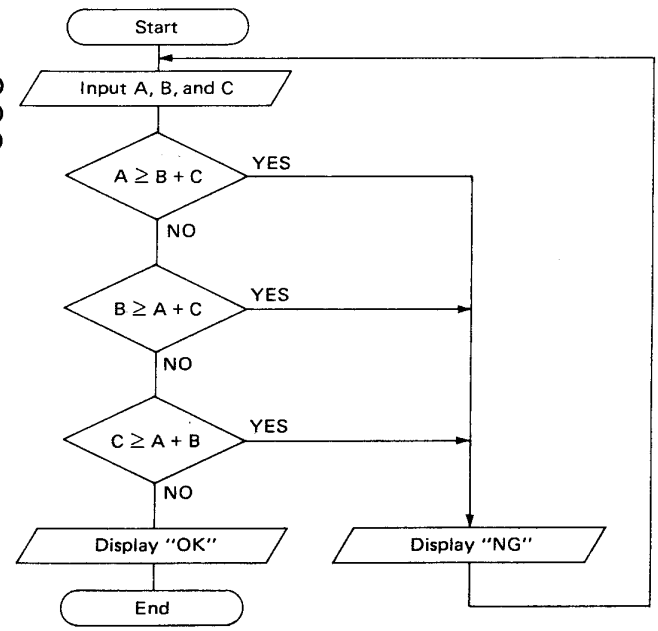


This means that if the IF statement is true, it goes to "YES", if it is not true, it goes to "NO".
In other words, an IF statement indicates a branch at which a decision will be made to determine the next operation according to the result.
This IF statement is used for terminating a loop when the number of data is not known, or when the next operation is changed according to the result of an operation, etc.

Example 1: Input the lengths of three sides of a triangle and determine if a triangle can be formed using the three sides.

The program is shown below.

```
10 INP A,B,C
20 IF A≥B+C THEN 70
30 IF B≥A+C THEN 70
40 IF C≥A+B THEN 70
50 PRT "OK"
60 END
70 PRT "NG"
80 GOTO 10
```



Three data are input for this program. Due to the nature of a triangle, the sum of the lengths of two sides is greater than the length of the remaining side. Each side is compared and if a triangle cannot be formed, "NG" will be displayed and the program will return to the data input operation. If a triangle can be formed, "OK" will be displayed and program is terminated.

The IF statement is composed using expressions as shown below.

IF comparison expression THEN line number (numerical expression) or #n (n= 0 to 9)

or

IF comparison expression ; command or assignment statement

The comparison expression following "IF" compares the right side of an equality or inequality sign with the left side. If YES, it will proceed after "THEN" or " ;". If NO, it proceeds to the next line. In other words, for line 20, if A is greater than or equal to the sum of B and C, a triangle cannot be formed. So, "THEN 70", in other words jump to line 70, command is performed.
This "THEN" includes the GOTO statement function.

If a line number is written following "THEN", it will jump to the designated line number. If " # " and 0 to 9 are written, it will jump to a program area (P0 to P9).
Also, when the comparison expression is YES, if a command or assignment statement is desired instead of a jump, " ; " is used instead of "THEN".
In these comparisons, constants, variables, numeric expressions, character constants and character variables can be used.

A > 10	variable and constant (YES if A is greater than 10)
X ≥ Y	variable and variable (YES if X is greater than or equal to Y)
N = L + 3	variable and numeric expression (YES if N is equal to the sum of L and 3)
A$ = "XYZ"	character variable and character constant (YES if the character string in A$ is equal to "XYZ")
P$ ≠ Q$	character variable and character variable (YES if the character string in P$ is unequal to the character string in Q$)

* Comparison of variables with character variables cannot be performed.
* Character string comparison applies to the ASCII codes.

"THEN" or " ; " can be used differently, depending on what follows them.

THEN 150	(line number)	; PRT A
THEN #9	(program area)	; Z = X + Y

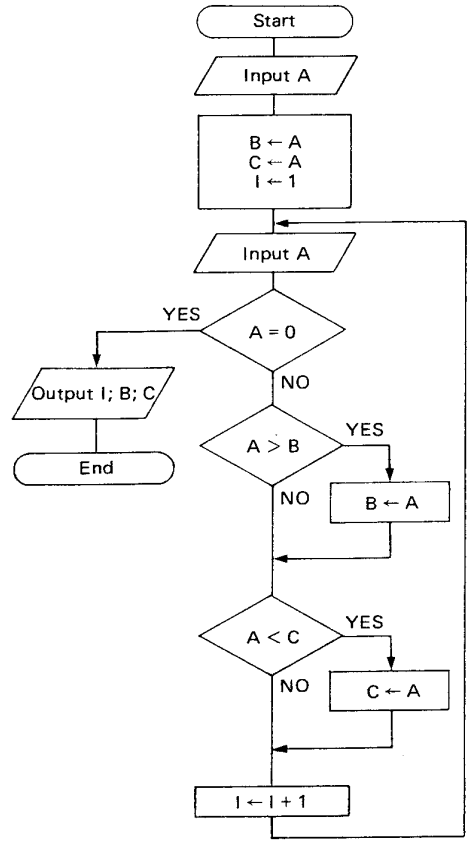
Application of the PRT statement
In the previous program, "OK" and "NG" will be displayed to show the result of the comparison. This is the application of the PRT statement.
If PRT A is written, the numeric value in variable A will be displayed.
If PRT "A" is written, the letter A will be displayed as is.
In other words, the item enclosed by quotation marks is handled as the character itself and displayed as is.

Example:			
	PRT A	PRT "A	PRT "ANSWER"
	⇓	⇓	⇓
	10 (if A = 10)	A	ANSWER
	PRT X	PRT "X"	PRT "N=" ; N
	⇓	⇓	⇓
	23 (if X = 23)	X	N = ␣15 (if N = 15)

Example 2: Obtaining maximum and minimum value is shown below.

```
 10 INP A
 20 B=A
 30 C=A
 40 I=1
 50 INP A
 60 IF A=0 THEN 110
 70 IF A>B;B=A
 80 IF A<C;C=A
 90 I=I+1
100 GOTO 50
110 PRT I;B;C
120 END
```

The INP statement on line 10 is used to input the initial data. Initial data is both the maximum value and the minimum value. Therefore, it is processed together with the data on line 20 and line 30 and the initial data is assigned with maximum value B and minimum value C.
Line 40 uses variable I to count the number of data. So, "1" is input as the initial data. The input statement of line 50 is used to input second and further data. So, the procedure is repeated by the GOTO statement on line 100. If data "0" is input using the IF statement on line 60, it will cause termination.
In other words, if "0" is input using the INP statement on line 50, control jumps to the PRT statement to output the result using line 110.
The IF statements on line 70 and 80 judge whether or not the input data is greater than the maximum value of the data already input or smaller than the minimum value and accomplishes replacement.
Upon completion of all data input, the number of data and the maximum and minimum value are displayed using line 110.



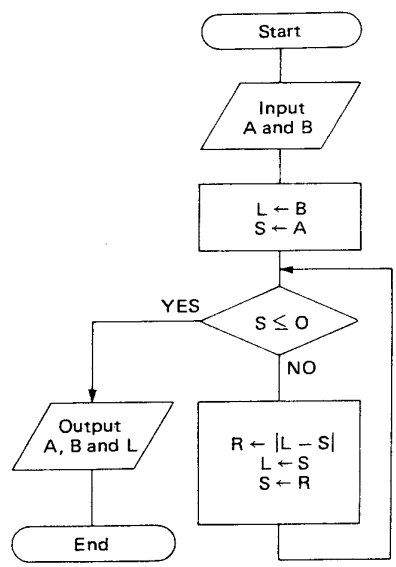
- **Various GOTO Statements and IF Statements**

Here we will take a look at some examples of applications using GOTO and IF statements.

Example 1: A program for determing the greatest common measure according to Euclid's algorithm.

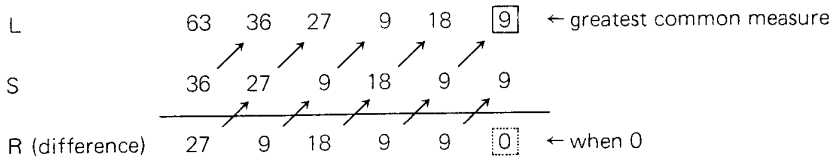
```
 10  INP  A , B
 20  L=A
 30  S=B
 40  IF  S≤Ø  THEN  90
 50  R=ABS(L-S)
 60  L=S
 70  S=R
 80  GOTO  40
 90  PRT  A;B;L
100  END
```

Flow chart



According to Euclid's algorithm, the second block of data is subtracted from the first block of data. Then, that result (difference) is subtracted from the second block. Then, subtract that difference from the preceding result. This is repeated until the difference becomes zero. When this happens, the previous difference is the greatest common measure. The greatest common measure of 63 and 36 is shown below.

Variable



In this program, the variables to obtain the differences are designated L and S respectively. The difference is assigned to R temporarily, then L and S are replaced by S and R respectively and the same operation is performed repeatedly (loop).
The IF statement on line 40 determines whether S, that is the difference after replacement, is zero or not. If zero, L represents the greatest common measure, and data A and B and their greatest common measure are displayed. If not zero, the difference is determined repeatedly as directed by the GOTO statement on line 80.
The IF statement is used to make decisions on which command (operation) is to be performed based on the result of computation.
Also, the replacement operations on lines 50 to 70 deserve special attention. Data A and B are initially assigned to L and S and their difference is temporarily assigned to R. Then the difference between L and S is determined again with L and S replaced by S and R respectively. This process is repeated until a difference of zero is obtained.

The relationships between L, S and R are shown below.

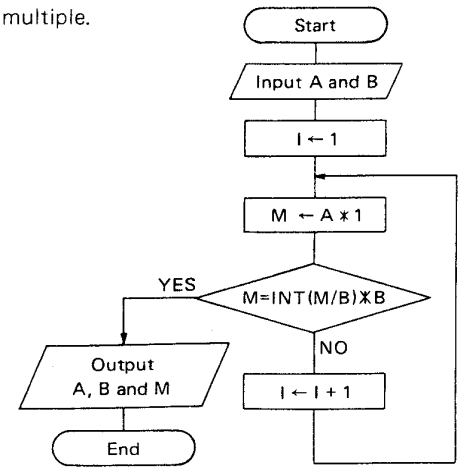
	L	S	R	
Initial	63	36	0	
Subtract (1st time)	63	36	27	R = ABS (L–S)
Substitute (1st time)	36	27	27	L = S, S=R
Subtract (2nd time)	36	27	9	R = ABS (L–S)
Substitute (2nd time)	27	9	9	L = S, S=R
Subtract (3rd time)	27	9	18	R = ABS (L–S)
Substitute (3rd time)	9	18	18	L = S, S=R
⋮		⋮		⋮
Subtract (6th time)	9	9	0	R = ABS (L–S)
Substitute (6th time)	[9]	0	0	L = S, S=R

↑
Greatest common measure

Note: If the order is changed in the above process, the significance of the substitution operations will be lost.

Example 2: Make a program to determine the least common multiple.

```
10 INP A,B
20 I=1
30 M=A*I
40 IF M=INT(M/B)*B THEN 70
50 I=I+1
60 GOTO 30
70 PRT A,B,M
80 END
```



This example program first inputs data A and data B and multiplies the initial value of A by one, two, three and so on until the result of that multiplication becomes equal to the least multiple of the value of B as tested by an IF statement.
The IF statement on line 40 compares variable "M" with the numerical expression "INT (M/B)*B". In other words, variable "M" is compared with the result of the numerical expression "INT (M/B)*B". This method is the same as:

```
N=INT(M/B)*B
IF M=N THEN 70
```

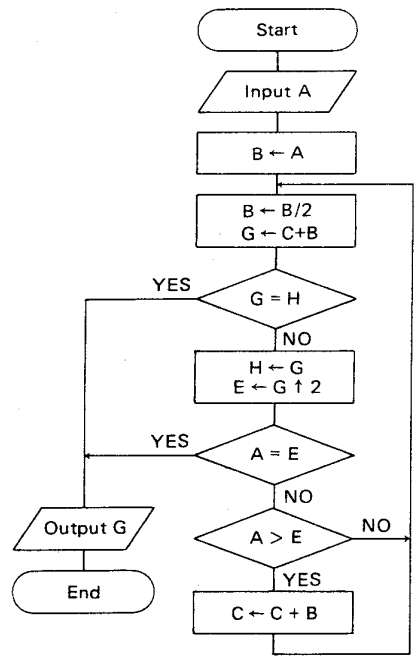
However, the method used here saves lines and the amount of memory.

INT (M/B) is a function command that makes the result of (M/B) an integer by removing the decimal portion.

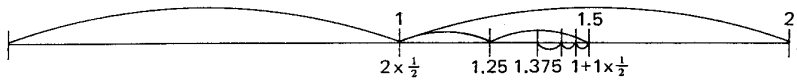
Then, multiplier I is increased one by one and, using the GOTO statement on line 60, is multiplied and judged again.
The variables used here are A, B, M and I. A and B are the two data. Since M is a multiple of A, it is easier to understand the value of M later.
Multiplier I is incremented one by one to produce a new multiple. It is often used as a variable which is incremented upon each occurrence of looping.

Example 3: Make a program to determine the square root according to dichotomy. The program is as shown below.

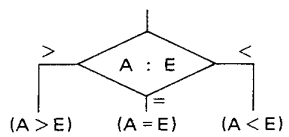
```
 10 VAC
 20 INP A
 30 B=A
 40 B=B/2
 50 G=C+B
 60 IF G=H THEN 120
 70 H=G
 80 E=G↑2
 90 IF A=E THEN 120
100 IF A>E;C=C+B
110 GOTO 40
120 PRT G
130 END
```



According to dichotomy, one half of the value of data A is first determined and the square of the one half is tested to see whether it is equal to the value of A or not. If not, one half of the one half is further determined and squared until the approximate value of A is obtained. The computation process is shown below.



In this program, the "VAC" command on line 10 is used to clear the memory. Then data A is input. This data is used later in the comparison operations of the IF statements. Direct changes cannot be made. So changes are made by assignment to B and B is changed.
The IF statement is used here to determine the approximate value at line 60.
The upper limit of the computing digits is determined. Then G and H are used to determine the point at which the highest limit is reached and when the result is equal to the previous result, the operation is terminated.
The IF statement on line 90 is the same as the IF statement in example 2. If the square root can be divided, the square of the result determined here and data A will be equal and the result will be displayed.
The IF statement on line 100 tests to determine in which of the two equally divided parts of the result the solution (approximate value) lies.
The IF statements in this program are used differently from before, but the function of each IF statement is the same as before.
The program itself is somewhat complicated, so a combination of IF statements is required. The IF statements on lines 90 and 100 both compare A and E with each other. The flowchart for this operation is shown below.



However, BASIC does not provide for the use of three-branch IF statements. Instead, a combination of dual IF statements is used.
The IF statements have different effects depending on where they are used in a program, as in Example 1 and Example 2.
In Example 1, an IF statement is used at the beginning of the loop so that it may cause a jump to END prior to the execution of the operation.
In Example 2, however, operations are executed and then testing of the specified condition is performed.
These two methods show that, depending on the data, the IF statement can be entered at the beginning of a loop, as in Example 1, or at the end of an operation, as in Example 2.

■ FOR-NEXT Statement

The FOR·NEXT statement is used when the number of occurrences of the same loop operation is known.

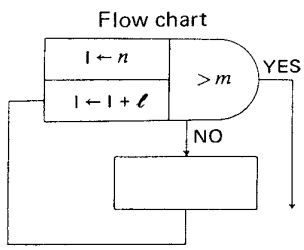
● FOR·NEXT Statement Function

The FOR·NEXT statement is composed as shown below.

FOR variable = *n* TO *m* STEP *ℓ*

initial value
terminal value
increment ⇒ variable or numeric expression

Flow chart



Operation command of numeric expression, etc.
NEXT variable

This statement commands the action specified between "FOR" and "NEXT" to be repeatedly executed while a variable changes from n to m in increments of ℓ. Control proceeds to the line next to "NEXT" when the variable is changed to m.
For example, the program below shows how to execute a given action while variable I increases from 1 to 10 in increments of 2.

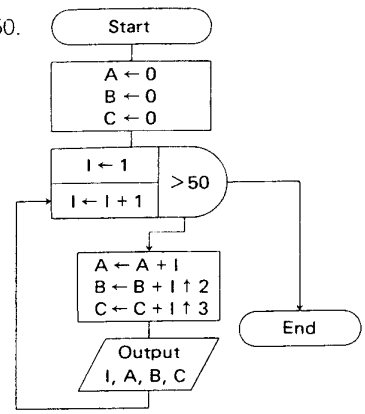
```
FOR I = 1 TO 10 STEP 2
.....
NEXT I
```

This STEP can be omitted if the variable increases in increments of one.

```
FOR I = 1 TO 10 STEP 1   is the same as
FOR I = 1 TO 10
```

* The FOR·NEXT statement only increases a variable in increments of a specified value and cannot decrease it. If a variable is to be decreased from 10 to 1, then it must be written as a negative number as in "STEP −1".

Example 1: Make a table of $\sum_{i=1}^{n} i$, $\sum_{i=1}^{n} i^2$, $\sum_{i=1}^{n} i^3$, with n ranging from 1 to 50.



This program is shown below.

```
10 VAC
20 FOR I=1 TO 50 STEP 1
30 A=A+I
40 B=B+I↑2
50 C=C+I↑3
60 PRT I,A,B,C
70 NEXT I
80 END
```

This example can only be done using a FOR·NEXT statement. The totals of i, i^2 and i^3 ($\sum_{i=1}^{n}$) are displayed while variable n is increased from 1 to 50 in increments of one.
Namely, the operation specified between "FOR . . . . . " on line 20 and "NEXT . . . . . " on line 70 is repeated 50 times as I is increased from 1 to 50 in increments of 1.
This example can be accomplished using an IF statement.

```
10 VAC
20 I=1
30 A=A+I
40 B=B+I↑2
50 C=C+I↑3
60 PRT I,A,B,C
70 I=I+1
80 IF I=51;END
90 GOTO 30
```

In this manner, if we compare the example using the FOR·NEXT statement with the example using the IF statement, the function of the FOR·NEXT statement will be clearly understood.

In other words, the FOR·NEXT statement combines the testing functions of an IF statement and a GOTO statement as well as an incrementing function.

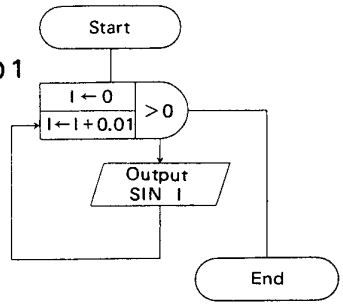
```
20 FOR I=1 TO 50 STEP 1 ⇨ { 20 I=1
                           { 70 I=I+1

80 NEXT I ⇨ { 80 IF I=51;END
            { 90 GOTO 30
```

In this manner, the FOR·NEXT statement has both an incrementing and testing function, so it is a very flexible and convenient command when the number of occurrences of a loop operation is known.

Example 2: Make a program to produce a table of sine functions from 0 to 1 in increments of 0.01.

This program is shown below.

```
10 MODE 4
20 FOR I=0 TO 1 STEP 0.01
30 PRT SIN I
40 NEXT I
50 END
```



This example determines the sine functions from 0 to 1 in increments of 0.01. The FOR•NEXT statement increments can be made using 0.01 increments. This incrementation can be accomplished because this equipment's internal computation system uses a decimal system. SIN I can be obtained and input using the statement on line 30.
"MODE 4" of line 10 is to specify "degree" for the unit of angle. "MODE 5" and "MODE 6" specify "radian" and "gradient" respectively.

• Various Loops

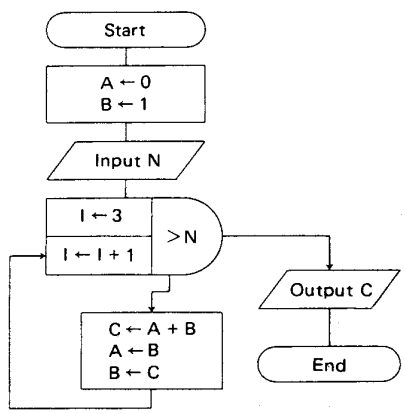
Example : Determine the nth number in a Fibonacci series.

A Fibonacci series is a series of integers in which each integer is equal to the sum of the two preceding integers. In other words, the sum of the first and second numbers is equal to the third number, the sum of the second and third numbers is equal to the fourth number, etc.

0 , 1 , 1 , 2 , 3 , 5 , 8 , 13, 21, 34,
0+1 1+1 1+2 2+3

This program is shown below.

```
10 A=0: B=1
20 INP N
30 FOR I=3 TO N
40 C=A+B
50 A=B
60 B=C
70 NEXT I
80 PRT C
90 END
```



Variable A, B and C represent the variables in the series. A is the value of the preceding number, B is the value of the next number and C is the total (the number following B).
Also, an initial value of 0 is assigned to A and an initial value of 1 is assigned to B. Then the operation is started at the third position.
The FOR•NEXT statement starts the loop beginning at 3 and continues it up to the desired point (Nth value). The initial value of the FOR•NEXT statement does not necessarily start from 1.

The substitution operations on lines 40 to 60 require special attention. The sequence cannot be changed. Once the sum of A and B are input into C, then B is input into A, then C is input into B.
Unless this sequence is followed, substitution cannot be performed correctly.
The manner in which this FOR•NEXT statement is used differs from the previous example. Therefore, variable N is used because the terminal value is changed based on later inputs.

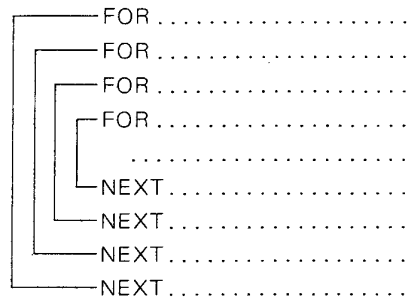
This example can be accomplished using an IF statement. A sample program using an IF statement is shown below.

```
10 A=0: B=1: I=3
20 INP N
30 C=A+B
40 A=B
50 B=C
60 IF I≠N; I=I+1: GOTO 30
70 PRT C
80 END
```

In this program, the IF statement on line 60 increments variable I while effecting the function of a GOTO statement. Initial values are set using a multistatement on line 10.

• Nesting

FOR•NEXT loops can be used to embed up to 8 levels. This embedding is called "nesting".

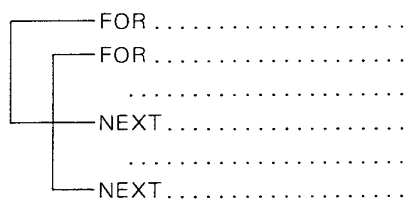


This is an example of 4 levels of nesting. One FOR•NEXT statement is input within another FOR•NEXT statement as shown.

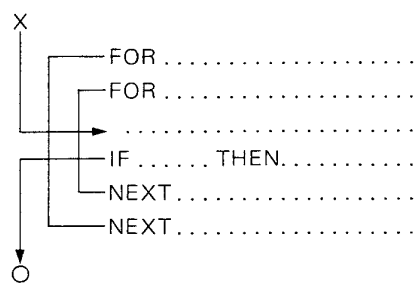
When embedding a number of these, care must be used concerning the NEXT statements which correspond to their FOR statements and their variables.

Also, the nesting must be done as shown above with the loops completely within one another. Overlapping FOR•NEXT loops cannot be used with a portion of the loop falling outside as in the following example of an incorrect nesting.

This type of FOR•NEXT loop cannot be used.



Furthermore, control can exit a FOR•NEXT loop but cannot enter a FOR•NEXT loop.



4-5-2 Arrays

Arrays, fall into two categories: one-dimensional (list) and two-dimensional (table) depending on the elements, both of which may be used.
A one-dimensional array is represented by A(i) or B(j), with (i) or (j) being subscripts. A two-dimensional array is represented by $x(i,j)$ or $y(n,m)$, with (i,j) or (n,m) being two-variable subscripts.

Examples of the two types are shown below.

One-dimensional array

i	A
0	A0
1	A1
2	A2
3	A3
4	A4
5	A5
6	A6
7	A7
8	A8
9	A9
10	B0
⋮	

Two-dimensional array

i \ j	0	1	2	3	4	5	6	7	8	9
0	A0	A1	A2	A3	A4	A5	A6	A7	A8	A9
1	B0	B1	B2	....	....	....	....	....	B8	B9
2	C0	C1	....	....	....	....	....	....	....	C9
3	D0	....	....	....	....	....	....	....	....	D9
4										
5										
6										
7										
8										
9										
10										
⋮										

This equipment's arrays use memory A0 to A9 to T9 increased using memory expansion.
One-dimensional array is represented by A(0) to A(199).
Two-dimensional array is represented by A(0,0) to A(19, 9).
In other words, A(0) and A0 are the same memory, A(10) and B0 are the same memory, A(1,0) and B0 are the same memory, and A(18,7) and S7 are the same memory.
Furthermore, only A can be used as the variable in the array.

* When the array is used, it is necessary to expand the number of memories according to the size of the array.
  This operation is performed using the manual.

  DEFM n (n is in a range of 0 to 19 and should be 1 or greater when used in an array)
  [EXE] (See Page 12 for details.)

This operation should be done. If the number of memories is smaller than the size of the array, an error will result when operated.

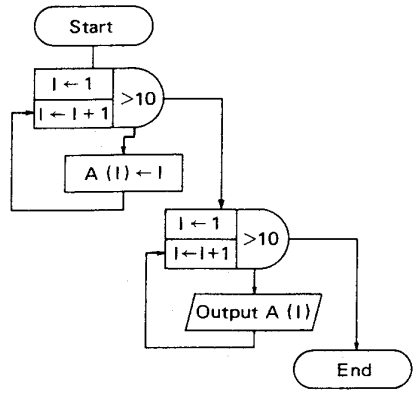
■ One-dimensional Arrays

A one-dimensional array is an arrangement of 0 to 199 elements with subscripts attached.

Example: A(0), A(1), ... A(199)

Example 1: Make a program to display the value of i in a one dimensional array A(i) as it changes from 1 to 10.

Flow chart



This program is shown below.

```
10 FOR I=1 TO 10
20 A(I)=I
30 NEXT I
40 FOR I=1 TO 10
50 PRT A(I)
60 NEXT I
70 END
```

The FOR•NEXT statements on lines 10 to 30 have been explained previously. However, A(I) is used on line 20 and a value of I is assigned. Also, the FOR•NEXT statement on lines 40 to 60, in a similar manner, display the value of the array contents using the PRT statement on line 50.

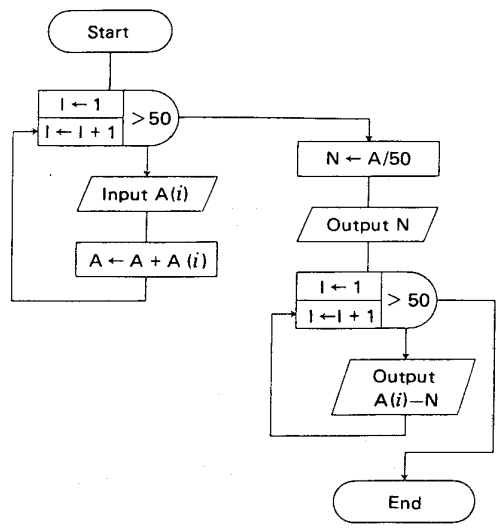
Furthermore, before executing this program, it is necessary to expand the memories to 20.

DEFM 2 [EXE]

In this manner, the array stores data as separate variables simply by changing the elements without changing the names of the variables. So, this is a convenient function when used with the FOR•NEXT statement.

Example 2: Make a program to display the difference between the average score and the individual scores by inputting the test scores of 50 students.

```
10 A=0
20 FOR I=1 TO 50
30 INP A(I)
40 A=A+A(I)
50 NEXT I
60 N=A/50
70 PRT N
80 FOR I=1 TO 50
90 PRT A(I)-N
100 NEXT I
110 END
```



For this program, the scores are input on line 30 using array A(I).
Sum total is obtained using line 40. An array is used for score input (separated). Then, the average score is calculated and the difference between the stored data (scores) and the average score is obtained and displayed.
In this manner, if an array is used to make a program when many data are input, the data can be input simply using a short program.

If an array is not used, it will be as follows:

A(1) ⟷ A
A(2) ⟷ B
A(3) ⟷ C
⋮ ⋮

and the input statement will be

```
INP A,B,C, . . . . . .
```

and the separation of the data will require a lot of time.

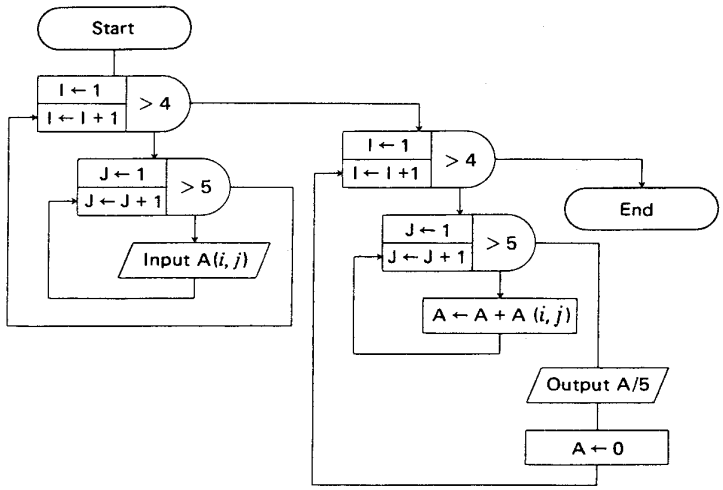
■ Two-dimensional Arrays

A two-dimensional array is a combination of multiple one-dimensional arrays. The format is A(i,j), A being the name of the array and i and j being elements. The variables used as variable names are similar to the variables used in a one-dimensional array. The elements are i and j, which are numeric expressions.

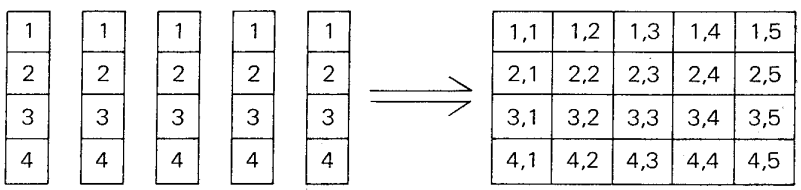
Example 1: Make a program to obtain the average score of each subject using the sum total of the scores of 5 students in language, mathematics, physics and sociology.

This program is shown below.

```
10 VAC
20 FOR I=1 TO 4
30 FOR J=1 TO 5
40 INP A(I,J)
50 NEXT J
60 NEXT I
70 FOR I=1 TO 4
80 FOR J=1 TO 5
90 A=A+A(I,J)
100 NEXT J
110 PRT J
120 A=0
130 NEXT I
140 END
```



The input portion and output portion can be separated into 2 parts. FOR•NEXT statements are used twice on lines 20 to 60 and perform input into the two-dimensional array. FOR•NEXT statements are again used twice on lines 70 to 130 and the average score is output at this time.
In other words, a two-dimensional array is a combination of multiple one-dimensional arrays.



4 + 4 + 4 + 4 + 4 = 20　　4 X 5 = 20

The two variables which form the elements of a two-dimensional array may be changed either concurrently or separately. On these occasions, it is convenient to use FOR•NEXT statements twice. In this program, i and j are used. First, determine i, then change j. Subsequently, i is changed, then j is changed again. This operation is repeated over and over. Either i or j may come first but the sequence of data input will differ.

```
FOR I=1 TO 4
FOR J=1 TO 5 ──→ A(1,1)
                 A(1,2)
                 A(1,3)
                 A(1,4)
                 A(1,5)
                 A(2,1)
                   :
                   :
                 A(4,4)
                 A(4,5)
```

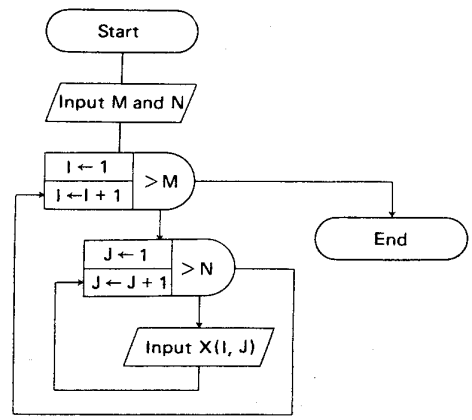
```
FOR J=1 TO 5
FOR I=1 TO 4 ------> A(1,1)
                    A(2,1)
                    A(3,1)
                    A(4,1)
                    A(1,2)
                      :
                      :
                      :
                    A(3,5)
                    A(4,5)
```

Let's consider a two-dimensional array with variables like A(1,1) or A(2,3) arranged in a matrix format.

Example 2: Make a program for collection of data to obtain the sum total of the matrix. Furthermore, the number of data will be input later.

This program is used to store all of the data for obtaining the sum total of the matrix in a two-dimensional array.

```
10 VAC
20 INP N,M
30 FOR I=1 TO M
40 FOR J=1 TO N
50 INP A(I,J)
60 NEXT J
70 NEXT I
80 END
```

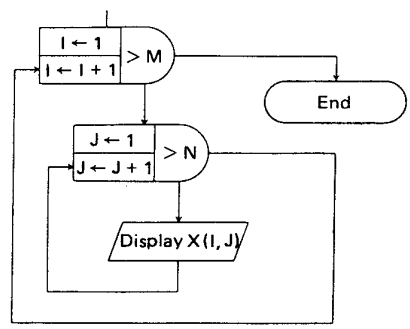


Here, the program is only for input. Actually, it does not terminate after line 80, but the program continues.
The FOR·NEXT statements on lines 30 to 70 are used to input data in the same manner as shown in Example 1.

Since this program is only for the input portion, the program for computation and output is added according to the actual use:

For example, to display all the data.

```
10 VAC
20 INP N,M
30 FOR I=1 TO M
40 FOR J=1 TO N
50 INP A(I,J)
60 NEXT J
70 NEXT I
80 FOR I=1 TO M
90 FOR J=1 TO N
100 PRT A(I,J)
110 NEXT J
120 NEXT I
130 END
```



This is added from line 80 on. The method of using is the same as the FOR·NEXT statements on lines 30 to 70. The only difference is that the output is substituted for the input.

In the above manner, a two-dimensional array is a kind of variable. It has 2 elements which are subscripts attached to the variable name.
For example, the variable is determined by the numerical value of i and j in A(i,j).

4-5-3 Input/Output Commands

This chapter comprehensively explains the input/output commands used with this equipment.

■ Input Commands

An input command is a command used to input data during program execution and uses the INP statement and KEY functions.

• INP Statement

The INP Statement is used to manually input data into the variables during program execution when "?" is displayed during awaiting input condition.
The INP statement is composed of

INP ["character string"], variable, ["character string"], variable . . . . . .

These character string can be omitted but once they are input, the character string is displayed before the awaiting input condition. It can be made into a message. Variables are numerical variables, character variables and exclusive character variable ($), and input data are assigned.

Example:

When `INP A`

When `INP "A=",A`

?
A=?

The process will proceed to the next step by inputting data and pressing the [EXE] key during the awaiting input condition using the INP statement.

Example: Add a message to the INP statement for the program for determining the sum, difference, product and quotient of 2 variables.

This program is shown below.

```
10 INP "A=",A,"B=",B
20 S=A+B
30 D=A-B
40 P=A*B
50 Q=A/B
60 PRT S,D,P,Q
70 END
```

Program execution is shown below.

Operation:

RUN [EXE]	A=?
45 [EXE]	B=?
23 [EXE]	68
[CONT]	22
[CONT]	1035
[CONT]	1.956521739

In this manner, the character strings which are written in the INP statement are displayed as messages and key input is easier.

For this INP statement, if character data is input for the numerical variables, an error will occur. If the error is cancelled using the AC key, "?" will be displayed again and return to awaiting input condition.
Also, the input for the numerical variable can be input as a numerical expression.
Furthermore, since the awaiting input condition cannot be cancelled by just pressing the AC key, use the following procedure to cancel it.

1. If the EXE key is directly pressed without making any input, "STOP" will be displayed and program execution will stop. Then, if the AC key is pressed, the awaiting input condition can be cancelled. Press the CONT key to continue the program.
2. If MODE 0 are pressed to redesignate the RUN mode, the awaiting input condition will be cancelled. In this case, the program cannot be continued.

• KEY Functions

This function reads one character for each key which has been pressed during program execution.
This function, unlike the INP statement, cannot be accomplished during the awaiting input condition. The program progresses as scheduled, so if there is no key input, nothing will happen.
The KEY function is composed as shown below.

Character variable = KEY

The character read by the KEY function is assigned to the designated character variable.

Example:

```
10 A$=KEY:IF A$="" THEN 10
20 IF A$="1" THEN 100
30 IF A$="2" THEN 200
40 IF A$="3" THEN 300
50 GOTO 10
    :
    :
```

This program is only for data input and separation portion, using the KEY function on line 10, it tests to see if the character data read was keyed in using the next IF statement.
Even if the EXE key is not pressed, this KEY function is capable of reading only the first key input.
However, it does not stop like the INP statement. So, by combining with the next IF statement, it becomes an awaiting input condition.
Also, lines 20 to 40 compare the contents in the read character variables and determines the jump destinations. In this manner, the KEY function can only read the character of one key.

■ Output Commands

In the output command, there are a PRT statement to display calculation results or data and a DMS statement to display the value of the numerical expression after conversion to sexagesimal degrees, minutes and seconds. Also, there is a WAIT statement to determine the display time.

• PRT Statement

A PRT statement is composed as shown below.

PRT [**Output control function** { , ; }] { **Numerical expression** / **Character expression** } [{ , ; } .....]

{ } either one of the enclosed contents can be used
[ ] enclosed contents can be omitted

This PRT statement displays the value of the numerical expression or the numerical variable and displays the character strings enclosed with " " or character variable contents.

Example:

```
10 INP "A=",A,"B=",B
20 C=INT (A/B)
30 D=A-B*C
40 PRT A;"/";B;"=";C;"...";D
50 GOTO 10
```

This program computes the remainder, as in the quotient and remainder when A is divided by B.
The values of the variables are displayed in the manner as on line 40.
The items enclosed by " " are displayed as the characters themselves.
Furthermore, the manner in which the " , " and " ; " are displayed between the variables and character strings is different.
In the case of " ; ", the next data is displayed following the previous display.
In the case of " , ", the next data is displayed after the previous display is cleared once.
When the display time is not designated by using the WAIT statement in the case of " , ", "STOP" is displayed and the process stops. To continue the display press the [CONT] key.

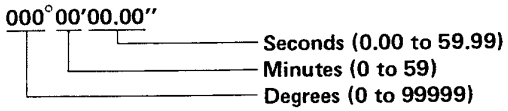
- **DMS Statement**

The DMS statement is a command to display the value of the numerical expression or the variable after conversion to sexagesimal. It displays decimal items in degrees, minutes and seconds.

Example:

```
10 INP N
20 DMS N
30 GOTO 10
```

This program inputs some data. That data is converted as is to sexagesimal. Even if the "PRT" command is not used, the display can be given only by the DMS statement.
The display is as shown below.



Furthermore, when the value of the numerical expression is greater than 99999, it is displayed as a decimal. The decimal portion of the seconds will be displayed rounded off to the third digit.

- **WAIT Statement**

The WAIT statement is a command to determine the display time using a PRT statement or a DMS statement. Display continuation using the [CONT] key is not required. It displays for a certain time only and then proceeds to the next program.
The WAIT statement is in the form as shown below.

WAIT numerical expression $(0 \leq \text{numerical expression} < 1000)$

The decimal portion of the numerical expression is cut off. The display will stop for the time which is designated by multiplying the value by approximately 0.05 seconds.
When no designation is made using this WAIT statement or when the value of the numerical expression is greater than 1000, it will come to a complete stop. "STOP" will be displayed and go to an await input condition.

Example:

```
10 WAIT 5
20 FOR I=1 TO 100
30 PRT I
40 NEXT I
```

This program displays the value from 1 to 100. When no designation is made using the WAIT statement, the process will stop after displaying the value of I. The next value is displayed by pressing the [CONT] key. However, when the display time is designated using the WAIT statement, display will be made approximately every 0.25 seconds.
The designation using this WAIT statement is cancelled when power is turned OFF (including auto power off) or at program execution starting time.

■ Output Control Functions

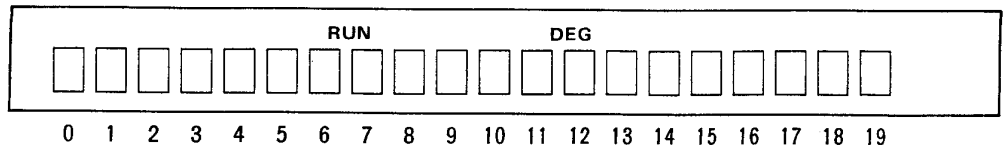
Output control functions designate the location of the output and the type of output.

• CSR Function

This is a control function which is used in a PRT statement. It designates the location of the output on the display (20 positions) using the PRT statement.
The format of the CSR function is shown below.

PRT CSR $\left\{\begin{matrix}, \\ ;\end{matrix}\right\}$ **numerical expression** (The decimal portion of the numerical expression is cut off and the value is 0 to 19.)

It uses the value of the numerical expression to determine at which unit from the left of the display to begin to output the data.
Furthermore, the method for counting the units on the display is shown below.



Example:

```
10 INP X
20 PRT X;CSR 8;X↑2
30 GOTO 10
```

This program obtains the square of data already input and displays the result as well as the data. The CSR function is used in the PRT statement on line 20.
The initial data "X" is displayed as is from the left side and the next "X↑2" is displayed from the eighth position from the left using the CSR function.

The program, when executed, will be as shown below.

Operation:

RUN[EXE]	?
7[EXE]	7 49
[CONT]	?
45[EXE]	45 2025
[CONT]	?
852[CONT]	852 725904

In this manner, since the two types of output are constantly displayed from a certain position, it is easier to see.

* When there is a " ; " following "CSR", it is displayed from the designated position but when " , " is used, display will not be made unless the display is cleared once. So the display will not be made continuously.

- **SET Statement**

The SET statement explained the number of effective digits designation and decimal designation for manual calculation but here we will explain how designation is made by writing in the program.
In the SET statement example shown below, E designates the number of effective digits, F designates the decimal and N cancels the designations.

$$\text{SET} \begin{Bmatrix} \text{E}\,n \\ \text{F}\,n \\ \text{N} \end{Bmatrix} \qquad (n \text{ is 0 to 9})$$

Example 1:

```
10 SET E4
20 MODE 4:WAIT 10
30 FOR X=0 TO 180 STEP 5
40 PRT SIN X
50 NEXT X
60 SET N
70 END
```

This program obtains sin x in 5 degree increments of x from 0 to 180 degrees.
In order to obtain the number of effective digits up to 4 digits in this example, sin x is obtained using "SET E4".
If the SET statement is written at the beginning of the program, all of the subsequent displays (result output) will be displayed with an effective number of digits of 4 digits.
If calculation is continued, the result can only be obtained up to 4 effective digits. So, "SET N" is used on line 60 in order to cancel the designation.

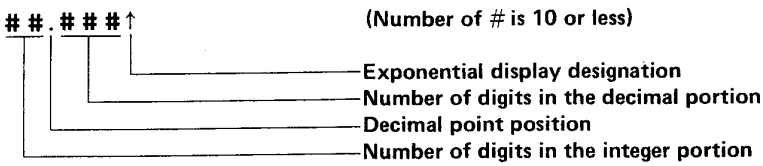
Example 2: Designate the number of decimal positions to be 4 digits for the program in Example 1.

```
10 SET F4
20 MODE 4:WAIT 10
30 FOR X=0 TO 180 STEP 5
40 PRT SIN X
50 NEXT X
60 SET N
70 END
```

In this manner, the number of output digits can be controlled using the SET statement similar to manual calculation.

- **Output Formatting**

Computation results may not always be output in a specified format on the display because the positions of the decimal points or number of digits involved differ according to the values of the results. In this case, the output format can be specified using formatting to make it easier to read.
The symbols used for formatting are "#" and "↑". "#" is used to specify the number of digits in a mantissa and "↑" is used to designate the display of exponents.



The "#" to the left of the decimal point designates the number of digits in the integer portion.
The "#" to the right of the decimal point designates the number of digits in the decimal portion.
The final "↑" designates the display for attached exponents.

Format	Explanation	Display when X = 0.123456789 Y = 12.3456789	
①#	Integer portion 1 digit, no decimal portion	X = 0	Y = #
②#.###	Integer portion 1 digit, decimal portion 3 digits	X = 0.123	Y = #.###
③##.#####	Integer portion 2 digits, decimal portion 5 digits	X = 0.12346	Y = 12.34568
④##.##↑	Integer portion 2 digits, decimal portion 2 digits, exponential portion attached	X = 1.23E−01	Y = 1.23E 01
⑤#.###↑	Integer portion 1 digit, decimal portion 3 digits, exponential portion attached	X = 1.235E−01	Y = 1.235E 01

* For the case as shown in Example ① and Example ② , when the number of integers in the mantissa portion overflow, display becomes impossible and the output format itself is output.

Example:

```
10 MODE 4:WAIT 10
20 FOR X=30 TO 180 STEP 30
30 PRT SIN X;COS X
40 NEXT X
50 END
```

This program is used to obtain sin x and cos x every 30 degrees. Both are displayed continuously. However, if left as it is, the display of both will cause an irregular display which will be difficult to read. In this case, to straighten out the display and when 10 digits are not needed for the mantissa portion, the display can be made easier to read by using formatting.
When formatting is added to the program it will be as shown below.

```
10 MODE 4:WAIT 10
20 FOR X=30 TO 180 STEP 30
30 PRT ##.##↑;SIN X;CSR 10;COS X
40 NEXT X
50 END
```

Using this formatting, both results can be displayed at the same time and the output positions can be arranged.
Therefore, the display will be easier to read.

Program execution is shown below:

Operation:

RUN [EXE]

```
5.00E-01  8.66E-01
8.66E-01  5.00E-01
1.00E 00  0.00E 00
8.66E-01 -5.00E-01
5.00E-01 -8.66E-01
0.00E 00 -1.00E 00
```

4-5-4 Character Functions

Character functions are related to character variables (A$, B$, $, etc.). The handling method for these is different because they are used to input characters instead of numeric values. Therefore, these character functions determine the size of the character variables and operate to extract some characters from character strings in character variables.

LEN and MID

A LEN function is a command to count the number of characters in a character variable. The size of the character variable can thus be known.
A MID function is a command to extract some characters from the exclusive character variable ($). It is used for rearranging the character variables.
Since the LEN function can tell the size of the character variable, it is a convenient means of reserving output space for character variables or for dividing them when the size is not fixed.

Example:

```
10 A$="123"
20 B$="456"
30 C$=A$+B$
40 D=LEN(C$)
50 PRT D
60 END
```

This program displays the size (number of characters) of the character variable by adding character variables A$ and B$.
Character variables can only perform addition. However, since the size of the character variable is not known using that result, the LEN function is used to determine the size of that addition.
Only character variables can be used for the LEN function.

The format of the MID function is shown below.

MID (*m*[,*n*]) (*m*, *n*: numerical expression)

It extracts the *n* character portion from the *m*th character of the character string stored in the exclusive character variable ($).
If *n* is omitted, all the characters after the *m*th character will be extracted.
Furthermore, the decimal is cut off for both *m* and *n*, and their values are 1 to 30. If the sum of *m* and *n* is greater than the stored number of character variables plus 1, an error will occur.

Example: when $ = "ABCDEFGHIJ"

To extract four characters from the third character on → MID (3,4) . . . CDEF
To extract two characters from the first character on → MID (1,2) . . . AB
To extract all characters from the fifth character on → MID (5) . . . EFGHIJ

Example:

```
10 INP $
20 N=LEN($)/2
30 X$=MID(1,N):Y$=MID(N+1)
40 PRT X$,Y$
50 END
```

This program divides the input character strings in two using the LEN function and the MID function and displays.
This determines the size of the exclusive character variable $ on line 20. Those halves are assigned to N and the first half and the second half are divided using the MID function.
In this manner, when the size of the input character string is not fixed, the halves or thirds cannot be extracted, so the size is determined using the LEN function.
Furthermore, in this case, the character string that can be input into $ is limited to 30 characters.
However, since the number of characters in X$ and Y$ is limited to 7 characters, the number of characters input into $ is limited to 14 characters.

4-5-5 Subroutines

A subroutine (also called a subprogram) has a main routine which is different from the programs (also called main routines) considered previously. The subroutine is called from the main routine and, upon completion of the operation, returns to the original position in the main routine.
In other words, using the command (GSB) to jump from within the main routine to the subroutine, it jumps from that position to the designated subroutine. After the subroutine operation is completed, it returns to the position (GSB command) in the main routine where the jump to the subroutine occurred and the main routine process is continued.

```
          :
  100  GSB  1000      subroutine callout (jump) command
          :
 1000  A=B                                  }
          :                                  } subroutine area
 1100  RET           command to return to main routine
```

The commands required for this subroutine are "GSB" and "RET" statements. The command to jump (callout) to the subroutine is the GSB statement and the command to return from the subroutine to the main routine is the RET statement.
This GSB statement is used in the following formats.

(1) **GSB numerical expression** (the numerical expression is cut off at the decimal and uses line numbers 1 to 9999)

(2) **GSB #numerical expression** (the numerical expression is cut off at the decimal and uses $0 \leq n < 10$)

The first method uses direct designation to write the line number directly following "GSB" and can indirectly designate the subroutine location using the value contained in the variable.
The second method uses a subroutine in another program area (P0 to P9) to designate directly and indirectly.
Furthermore, neither will return unless the RET statement is written at the end of the subroutine.

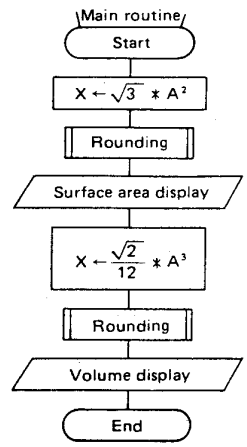
■ Subroutine Fundamental Programs

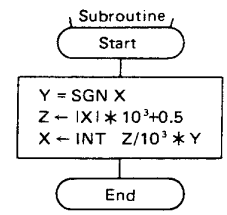
Here we will explain how to arrange the subroutines using actual program examples.

Example 1: Make a program to determine the surface area and volume of a regular tetrahedron, in which the length of the sides is A, computed down to three decimal places. Provided that a rounding program has been incorporated into a subroutine.

$$\begin{cases} S = \sqrt{3}a^2 \\ V = \dfrac{\sqrt{2}}{12}a^3 \end{cases}$$

```
  10  INP  A
  20  X=SQR  3*A↑2
  30  GSB  1000
  40  PRT  X
  50  X=SQR  2/12*A↑3
  60  GSB  1000
  70  PRT  X
  80  END
1000  Y=SGN  X
1010  Z=ABS  X*10↑3+0.5
1020  X=INT  Z/10↑3*Y
1030  RET
```





The main routine for computing the surface area and volume of the regular tetrahedron can be easily understood, as it is similar to the programs considered previously.
The difference is the rounding program included in the main routine as a subroutine from line 1000 on. This subroutine has the function of computing the surface area and volume, as determined on lines 20 and 50, down to three decimal places by rounding.

The subroutine can be used as many times as desired.
It is used twice in this program, first for rounding the value of the surface area and second for rounding the value of the volume.
If a subroutine is not used and only a main routine is used as in the past, the program will be longer, as shown below.

```
 10 INP A
 20 X=SQR 3*A↑2
 30 Y=SGN X
 40 Z=ABS X*10↑3+0.5
 50 X=INT Z/10↑3*Y
 60 PRT X
 70 X=SQR 2/12*A↑3
 80 Y=SGN X
 90 Z=ABS X*10↑3+0.5
100 X=INT Z/10↑3*Y
110 PRT X
120 END
```

This program and the program using the subroutine described above perform the same processing (operations) but the program using the subroutine is shorter and simpler.
In other words, the subroutine is used to simplify the program by eliminating repetitive operations.

* In this example, since the rounding program is incorporated into a subroutine, "PRT X" is not included in the subroutine. Actually, in this program, lines 30 to 60 and lines 80 to 110 are the same and this portion is incorporated into a subroutine.

This subroutine is also a part of the main routine, so it is not incorporated within the main routine but is incorporated after the end ("END") and must be ended with an RET statement. If it is incorporated along the way in the main routine, it will be read as the main routine and processed repeatedly, thereby causing an error upon execution of the RET statement.
Therefore, as in this example, the subroutine area must be separated by starting it at line 1000 or 5000.
The subroutine is similar to the GOTO command (GOTO statement), and is considered a jump command but differs in that an RET statement is always attached so that it returns to the GSB statement after completion.

Example 2: Make a program to determine the surface area and volume of a regular tetrahedron and a regular octahedron in which the length of sides is A, computed down to four decimal places. Provided that the main routine and the subroutine for determining the area and volume are incorporated into P0 and the rounding subroutine is incorporated into P1.

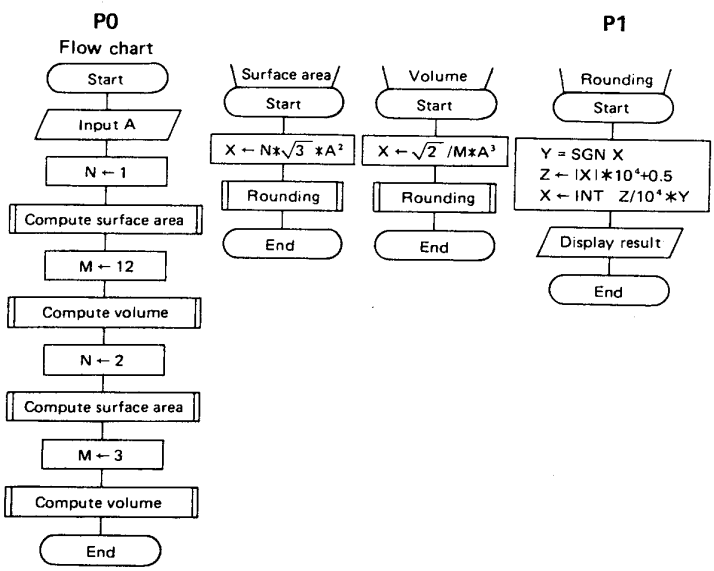
Regular tetrahedron

$S = \sqrt{3}a^2$, $V = \frac{\sqrt{2}}{12}a^3$

Regular octahedron

$S = 2\sqrt{3}a^2$, $V = \frac{\sqrt{2}}{3}a^3$

```
P0 10 INP A
   20 N=1
   30 GSB 1000
   40 M=12
   50 GSB 2000
   60 N=2
   70 GSB 1000
   80 M=3
   90 GSB 2000
  100 END
 1000 X=N*SQR 3*A↑2
 1010 GSB #1
 1020 RET
 2000 X=SQR 2/M*A↑3
 2010 GSB #1
 2020 RET
```



```
P1    10 Y=SGN X
      20 Z=ABS X*10↑4+0.5
      30 X=INT Z/10↑4*Y
      40 PRT X
      50 RET
```

Three subroutines are used in this program: two common portions for computing surface areas and volumes, and a rounding program.

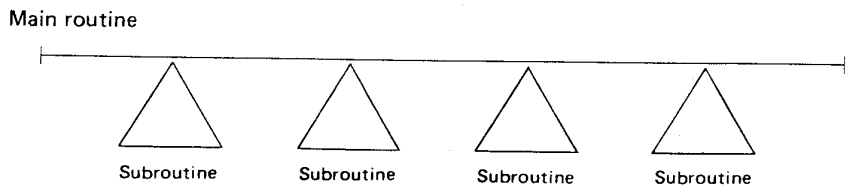
Using this method, commonly used computational operations can be incorporated into a subroutine for repetitive execution just by changing the values of N and M. In this manner, if common portions are incorporated into subroutines, memory can be saved and contents and computation method can be understood more easily.

Therefore, on lines 1010 and 2010 of this program, a subroutine is called out from a subroutine. This method is called "nesting." It calls out a subroutine from a subroutine similar to calling out a subroutine from a main routine.

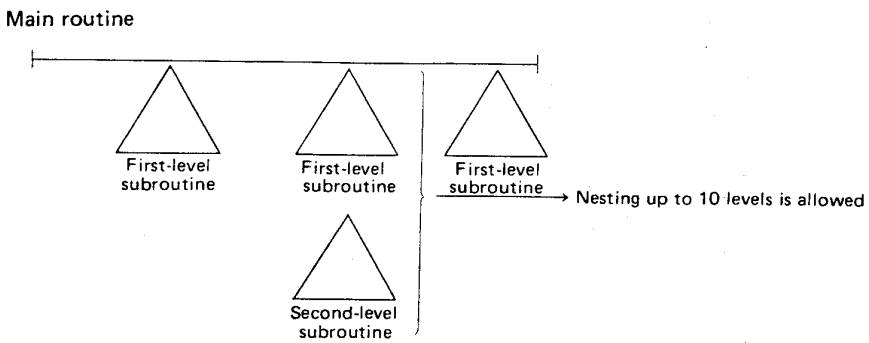
This nesting can be performed up to 10 levels (10 times), so any nesting beyond that causes an error.
In other words, it goes from the main routine to the first level subroutine. Then that subroutine acts as the main routine and the next (second level) subroutine is called out. Even at this time, the RET statement on which the subroutine is based must be written at the end of each subroutine without fail.
Similar to the rounding subroutine, this subroutine can be incorporated into another program area (in this case, P1). This method is convenient because it can be used as a subroutine in common with another program (a program written in another program area).

In this manner, many subroutines can be used in one program.

In the case of "nesting", however, up to 10 levels can be accomplished.



Using this method, many subroutines can be used.



In this manner, there is a way to omit the common portion using a subroutine. Also, when incorporated in a complicated program, a subroutine is used when portions are separated into groups.

The GSB command in a subroutine is considered similar to a GOTO statement. In other words, it determines the subroutine area to which it jumps from the main routine and then operates to return to the original position.

Furthermore, when the subroutine is written in the same program, it is necessary to write END at the end of the main routine, and if it is repeated, it is necessary to write a GOTO statement.

■ Indirect Designation of a Subroutine

Similar to the explanation for the GOTO statement, indirect designation can also be used for a subroutine.
The method of using is similar to that of a GOTO statement. It jumps to a designated subroutine and then returns to the original position.

Example 1:

```
 10 INP N
 20 GSB N+100
 30 PRT X*6
 40 END
101 X=5:RET
102 X=10:RET
103 X=15:RET
104 X=20:RET
105 X=25:RET
106 X=30:RET
107 X=35:RET
108 X=40:RET
109 X=45:RET
110 X=50:RET
```

In this program, the subroutine is designated indirectly using the value of input N. When N is 1 to 10, it goes to a subroutine between 101 and 110 and determines the value of variable X and displays the computation result of X * 6.

In this manner, since the indirect designation of the GSB statement determines the jump destination (subroutine) using the value of the variable, when the variable is one, it jumps to the first subroutine (line 101).
When the variable is two, it jumps to the second subroutine (line 102), and so on. It designates indirectly depending on the value of the variable.

Example 2: Use the indirect designation of the GSB statement to make the sorted totals program which was given in a previous example and which used indirect designation of the GOTO statement (5 divisions).

```
 10 VAC
 20 INP I
 30 GSB I+100
 40 GOTO 20
100 PRT A,B,C,D,E:END
101 INP J:A=A+J:RET
102 INP J:B=B+J:RET
103 INP J:C=C+J:RET
104 INP J:D=D+J:RET
105 INP J:E=E+J:RET
```

This program is almost the same as the previous example of an indirect designation of a GOTO statement. However, the GOTO statement on line 30 is replaced by a GSB statement.
This method also basically inputs code numbers 1 to 5 on line 20 and uses indirect designation on line 30 and separates using lines 101 to 105 subroutine. Each department is totaled in the subroutine. For this program as well, each subroutine is made one line using a multistatement. Each line becomes one subroutine, thereby making it easier to read.
Even in this case, the code number input on line 20 is 1 to 5. If any other number is used, an error will occur when there is no jump destination.

4-5-6 General Functions

General functions include trigonometric functions such as sine, cosine and tangent, and built-in functions. These functions are formatted by combining alphabetic characters on the keyboard. They can also be entered with the "one-key command" using a single key and can be used as easily as operating a small electronic calculator.
General functions may be used manually or incorporated within programs.

Function name		Format	Example
Trigonometric function	$\sin x$ $\cos x$ $\tan x$	SIN x COS x TAN x	SIN **30** SIN **A** SIN **(N+5)** COS **3.14** COS **I** COS **(L−3)** TAN **70** TAN **F** TAN **(F * 2)**
Inverse trigonometric function	$\sin^{-1} x$ $\cos^{-1} x$ $\tan^{-1} x$	ASN x ACS x ATN x	ASN **0.07** ASN **P** ASN **(Z+Y)** ACS $\pi/5$ ACS **D** ACS **(C−X)** ATN **6.5** ATN **V** ATN **(Q+0.5)**
Hyperbolic function	$\sinh x$ $\cosh x$ $\tanh x$	HSN x HCS x HTN x	HSN **26.8** HSN **T** HSN **(T+E)** HCS **1.45** HCS **0** HCS **(R/2)** HTN **0.861** HTN **H** HTN **(U+0.9)**
Inverse hyperbolic function	$\sinh^{-1} x$ $\cosh^{-1} x$ $\tanh^{-1} x$	AHS x AHC x AHT x	AHS **76.2** AHS **G** AHS **(J+1)** AHC **16** AHS **W** AHS **(S * 2)** AHT **0.91** AHT **M** AHT **(H−E)**
Square root	$\sqrt{x}$	SQR x	SQR **69** SQR **A** SQR **(R * S)**
Exponential function	e^x	EXP x	EXP **8** EXP **F** EXP **(D+L)**
Natural logarithm	$\log_e x$	LN x	LN **43** LN **P** LN **(U+T)**
Common logarithm	$\log_{10} x$	LOG x	LOG **24.6** LOG **R** LOG **(G+15)**
Integration (Maximum integer not exceeding x)	INT x	INT x	INT **347.457** INT **V** INT **(Q+U)** INT **−45.43**
Fractionalization (x with its integer part removed)	FRAC x	FRAC x	FRAC **73.54** FRAC **N** FRAC **(H+B)**
Absolute value	$\lvert x \rvert$	ABS x	ABS **−9.43** ABS **L** ABS **(K/P)**
Sign (If $x > 0$, 1) (If $x = 0$, 0) (If $x < 0$, −1)		SGN x	SGN **79** SGN **E** SGN **(P−0)**
Conversion of sexagesimal to decimal		DEG (Degree, Minute, Second)	DEG **(39 , 34 , 15)** DEG **T** DEG **(R+5)** Degree Minute Second
Conversion of decimal to sexagesimal		DMS x	DMS **12.582** DMS **Z** DMS **(P+0.05)**
Coordinate conversions Rectangular to polar Polar to rectangular		 RPC x, y PRC r, θ	 RPC **14,20.7** RPC **A,B** PRC **4.5,2/π** PRC **F,F**

Function name		Format	Example
Significant digit specification (x is determined down to the 10^y-th significant digit place by rounding)		RND (x, y)	RND (123.456, 2) RND (A, C) RND (F+E, G−5)
Random number generation (Uniform random number generation in the range $0 < x < 1$)		RAN #	RAN #
Factorial	x !	x !	8 ! A ! (I+J) !
Unit of angular measure	Degree	MODE 4	One right angle = 90 degrees
	Radian	MODE 5	One right angle = $\frac{\pi}{2}$ radians
	Gradient	MODE 6	One right angle = 100 gradients

* x and y are constants, variables or numerical expressions.

Since these functions are built in they can be used in a program at once.

Example 1: Make a program to obtain the length of one side of a triangle using the angle enclosed by the other two sides.

[$C = \sqrt{a^2 + b^2 - 2ab\cos\theta}$]

```
10 MODE 4
20 INP A,B,C
30 D=SQR (A↑2+B↑2-2*A*B*COS DEG C)
40 PRT D
50 GOTO 20
```

Since the angular unit "degree" is used in the trigonometric function, MODE 4 appears on line 10 in this program. Then, the lengths of the two sides and the enclosed angle are input.
In accordance with the formula, square root SQR, and cosine COS are written in the numeric expression.

Example 2: Make a program to obtain the amplifier gain dB with input voltage X and output voltage Y.

[$dB = 20 \cdot \log_{10} \frac{Y}{X}$]

```
10 INP X,Y
20 Z=20*LOG (Y/X)
30 PRT Z
40 GOTO 10
```

If the input voltage is X and the output voltage is Y, the formula can be written as shown on line 20 to obtain the gain (Z).

Example 3: Make a program to generate three-digit random numbers using a random number generating function and a significant digit specification function.

```
10 N=RND(RAN#,-4)*1000
20 PRT N
30 END
```

Since random numbers can be generated as 10-digit mantissa in the range $0 < x < 1$, three digits must be taken as significant digits and multiplied by 1000 to extract three-digit numbers.

4-5-7 Statistical Processing

Since statistical processing capabilities are incorporated in the calculator as well as the built-in functions, it can be used easily.
These statistical processing capabilities include standard deviation calculation and regression analysis calculation for determination of correlation coefficients, etc.

■ Standard Deviation Calculation

Standard deviation calculation can be executed manually or written in a program. This chapter explains standard deviation calculation using programs.
Furthermore, for manual calculation, refer to the section on manual calculation.

Example 1: Make a program to obtain the difference between each data and the mean by storing all the data using a one-dimensional array.

```
10 SAC
20 INP "N=",N
30 FOR I=1 TO N
40 INP D
50 STAT D
60 A(I)=D
70 NEXT I
80 PRT "MX=";MX
90 FOR I=1 TO N
100 PRT "NO.";I;"=";A(I)-MX
110 NEXT I
120 END
```

The SAC command on line 10 in this program is a command to clear ("0") the statistical summation memory. The VAC command does not clear the statistical memory. Prior to performing statistical calculation, it is necessary to use this command. The number of data is input on line 20 and, using the FOR•NEXT statement, the number of times the data is input is determined.

In the FOR•NEXT statement following that, using the data input command "STAT" for standard deviation calculation, data is totaled and all the data is stored in array A(i).

After data input, since MX is the mean value callout command, again, using a FOR•NEXT statement, the mean value is subtracted from each data stored in the array and the result is displayed.

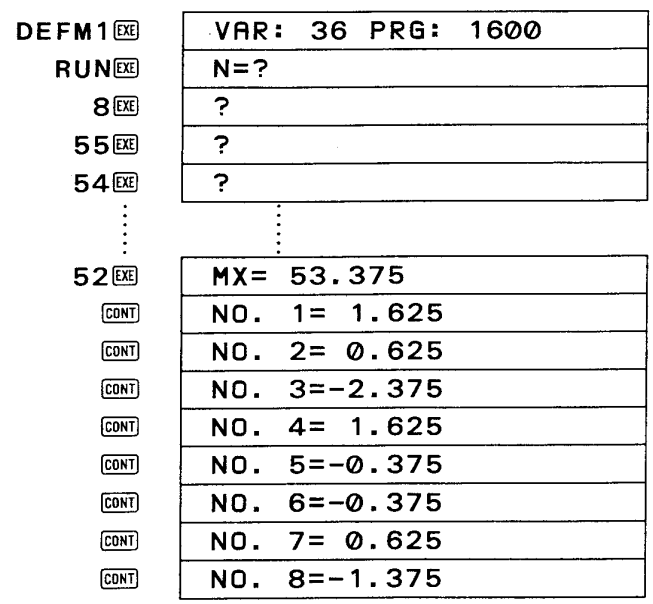
Let's perform the following example using this program.

Example: number of data = 8

Data: 55, 54, 51, 55, 53, 53, 54, 52

If this data is executed, the result will be as shown below.

Operation:



* When the array is used in this manner, be sure to expand the memory capacity.

Furthermore, the commands to callout the result of the standard deviation in this example are shown below.

CNT . . . . . **number of data** (n)
SX . . . . . **sum** (Σx)
SX2 . . . . . **sum of squares** (Σx^2)
MX . . . . . **mean** ($\bar{x}$)
SDX . . . . . **standard deviation** (σ_{n-1})
SDXN . . . . . **standard deviation** (σ_n)

Example 3: Make a program to obtain mean test score and standard deviation σ_n and each student's score deviation value.

Refer to example 1.

Deviation value SS = $\dfrac{(x-\bar{x})\cdot 10}{\sigma_n} + 50$

```
10 SAC
20 INP N
30 FOR I=1 TO N
40 INP D
50 STAT D
60 A(I)=D
70 NEXT I
80 PRT "MX=";MX,"SDXN=";SDXN
90 FOR I=1 TO N
100 S=(A(I)-MX)*10/SDXN+50
110 PRT "NO.";I;"=";S
120 NEXT I
130 END
```

In this program, line numbers 10 to 70 are the same as Example 1, however, the program for calculation and display from line 80 on is different.
Deviation value SS is obtained at line 100. This deviation value is obtained by each student's score using the FOR•NEXT statement.

In the above manner, using the built-in standard deviation function, the mean or standard deviation can be directly incorporated into the program.
Furthermore,
To input with frequency, it is performed similarly to manual calculation. Also, when incorporating data input into a program, the deletion and correction of erroneously input data must be accomplished manually.

■ Regression Analysis and Correlation Coefficients

As a statistical management function, in addition to standard deviation calculation, a regression analysis (2 variable statistics) calculation function is also incorporated.

For regression analysis calculation, only linear regression is built-in. By using this linear regression, however, logarithmic regression, exponential regression and power regression can be obtained using the program.

Linear regression calculation

- The linear regression formula is $y = A + Bx$. Constant term A and regression coefficient B are calculated as shown below.

$$A = \frac{\Sigma y - B\cdot\Sigma x}{n} \qquad B = \frac{n\cdot\Sigma xy - \Sigma x\cdot\Sigma y}{n\cdot\Sigma x^2 - (\Sigma x)^2}$$

- Correlation coefficient r of the input data pair is calculated using the formula shown below.

$$r = \frac{n\cdot\Sigma xy - \Sigma x\cdot\Sigma y}{\sqrt{\{n\cdot\Sigma x^2 - (\Sigma x)^2\}\{n\cdot\Sigma y^2 - (\Sigma y)^2\}}}$$

- These are handled in the program in the same manner as standard deviation calculation.

Example: Data input is performed manually. Using the output computation result, assemble a program to calculate constant term A, regression coefficient B, correlation coefficient r and coefficient of determination (r^2), and covariance $\left(\frac{\Sigma xy - n \cdot \overline{x} \cdot \overline{y}}{n-1}\right)$.

```
10 PRT "A=";LRA,"B=";LRB,"R=";COR,"R↑2=";COR↑2
20 X=(SXY-CNT*MX*MY)/(CNT-1)
30 PRT X
40 END
```

Since this program is only a simple display program, data input is performed manually.

Data input methods are shown below.

x **data** , y **data** [STAT]
or
x **data** , y **data** ; **frequency** [STAT]

The "LRA" used here is constant term A, "LRB" is regression coefficient B and "COR" is correlation coefficient r. Also, the coefficient of determination and covariance are calculated using calculation formulas and displayed.

Furthermore, the functions for displaying the results of regression analysis are shown below.

CNT . . . . . number of data (n)
SX . . . . . sum of x (Σx)
SY . . . . . sum of y (Σy)
SX2 . . . . . sum of squares of x (Σx^2)
SY2 . . . . . sum of squares of y (Σy^2)
SXY . . . . . sum of products of data (Σxy)
MX . . . . . mean of x ($\overline{x}$)
MY . . . . . mean of y ($\overline{y}$)
SDX . . . . . standard deviation of x ($x\sigma_{n-1}$)
SDY . . . . . standard deviation of y ($y\sigma_{n-1}$)
SDXN . . . . . standard deviation of x ($x\sigma_n$)
SDYN . . . . . standard deviation of y ($y\sigma_n$)
LRA . . . . . constant term (A)
LRB . . . . . regression coefficient (B)
COR . . . . . correlation coefficient (r)
EOX . . . . . estimated value of x ($\hat{x}$)
EOY . . . . . estimated value of y ($\hat{y}$)

In this manner, since regression analysis is automatically calculated using only the input data, it is fully usable for programs or display programs.
Furthermore, when data is input manually, be sure to press [F1] [SAC] prior to input.

Logarithmic regression calculation

- The logarithmic regression formula is y=A+B·ln$x$.
- This logarithmic regression is performed using the linear regression calculation function. Data input method is different from linear regression, so it is convenient for input in the program.

Example: Assemble a program using the linear regression analysis function to perform logarithmic regression analysis only by input data x and y.

Furthermore, analysis result output is performed manually and not included in the program.

```
10 SAC
20 INP X,Y
30 STAT LN X,Y
40 GOTO 20
```

This program is a simple program with only an input portion. It is composed of the SAC command which clears the data summation memory, the INP statement and data input STAT statement.

This STAT statement employs logarithm ln of x. Because the regression formula is y=A+B·ln$x$, y can be input as is. However, x must be input as the logarithm of x.

In this manner, a program is assembled for the input portion only. Output can also be performed manually. However, for this program, after data input, it is necessary to stop the program (press MODE 0) and press the function key for output.
Also, the number of data must be input initially to obtain automatic termination and to display the analysis results such as the standard deviation example.

Furthermore, since this program has 1 as a frequency, to make it possible to input frequencies, assemble it as shown below.

```
10 SAC
20 INP X,Y,Z
30 STAT LN X,Y;Z
40 GOTO 20
```

However, since the frequency input cannot be omitted in this program, even when there is only one, be sure to input "1".

Exponential regression calculation

- The exponential regression formula is $y=A\cdot e^{B\cdot x}$ ($\ln y=\ln A+B\cdot x$).
- This exponential regression is performed using the linear regression analysis function the same as logarithmic regression.

Example: Assemble an input/output program for exponential regression analysis. The output displays constant terms, coefficients and correlation coefficients.

```
10 SAC
20 INP "DATA=",N
30 FOR I=1 TO N
40 INP "X=",X,"Y=",Y
50 STAT X, LN Y
60 NEXT I
70 PRT "A=";EXP LRA,"B=";LRB,"R=";COR
80 END
```

Lines 10 to 60 of this program use a method similar to previous programs. However, data Y is not used as is but is input as logarithm ln. Because the regression formula is ln$y$=lnA+B·x, y must be input as the logarithm of y (ln$y$). This input portion, initial input of number of data is performed using a FOR·NEXT statement.
The output portion constant term A, in order to be used as is, is computed as ln A. To obtain A, it is necessary to obtain the anti-logarithm of ln A and calculate EXP A (e^A) using the exponential function.

The result becomes constant term A.
Furthermore, regression coefficient B and correlation coefficient r may be used as they are.

Power regression calculation

- The power regression formula is $y=A\cdot x^B$ ($\ln y=\ln A+B\cdot\ln x$).
- This power regression is also performed using the linear regression analysis function.

Example: Assemble a program for power regression analysis that displays constant terms, regression coefficients and correlation coefficients as analysis results and thus obtains estimated values.

```
10 SAC
20 INP "DATA=",N
30 FOR I=1 TO N
40 INP "X=",X,"Y=",Y
50 STAT LN X,LN Y
60 NEXT I
70 PRT "A=";EXP LRA,"B=";LRB,"R=";COR
80 INP "X OR Y",Z$
```

```
 90 IF Z="Y" THEN 130
100 INP "Y=",Y
110 PRT EXP EOX LN Y
120 GOTO 80
130 INP "X=",X
140 PRT EXP EOY LN X
150 GOTO 80
```

Lines 10 to 70 in this program have the same type of input/output portion but the data input using the STAT statement on line 50 is different from previously.

It inputs both data x and y as logarithm ln. Since the regression formula is $\ln y = \ln A + B \cdot \ln x$, it is necessary to input the logarithm of data x and y.

Since A in the output portion is also output in the form of logarithm lnA, anti-logarithm A is obtained. And, the estimated value calculation program has two cases. One estimates $\hat{Y}$ from data x. The other estimates $\hat{X}$ from data y. First, judgement is performed on lines 80 and 90. In other words, if Z$ is X, input data y to obtain $\hat{X}$. If Z$ is Y, input data x to obtain $\hat{Y}$.

In this manner, even power regression calculation can be accomplished easily using the program.
If an error is made when composing the data input portion of the program, it is necessary either to perform manually or to include an error recovery operation in the program.

4-5-8 Password

For this equipment, passwords can be assigned to protect composed programs. A password is a "code name." When the program composer or equipment operator does not want other persons to know about program contents, or does not want the program to be changed by others, if a password is attached, "LIST" or "CLR" commands cannot be used.

A password is in the form as shown below.

PASS "ABC"

Letters or numbers in quotation marks, like "ABC", become the password. Any number of letters or numbers up to 8 characters can be used.

Examples: PASS "CASIO"
PASS "#$"
PASS "AZ−1"

Since passwords are effective for each program area (P0 to P9), passwords can only be attached to only P0 or P9.
The password can be cleared by clearing the entire program using the "CLR ALL" command or by re-inputting the same password.
If a password is input following "LIST," the "LIST" command can be temporarily cleared.

Example:

PASS "CASIO" [EXE] . . . . . Password is set
LIST "CASIO" [EXE] . . . . . Password is temporarily cancelled (or LIST 20 "CASIO" [EXE]).

This temporary password cancellation is only effective for the LIST command at the time of input. If the password is not input again following that, a password error will occur.

4-5-9 Option Specifications

■ CMT (cassette magnetic tape)

An MT tape recorder can be connected to this equipment for use as an external memory device. Since this MT can store programs and data, important programs, etc., can be saved. Another type of recorder can also be used for recording. Also, if the recorder has a remote control terminal, remote control can be accomplished from the main unit through the optional adaptor FA-2, so it is more convenient if a remote control terminal is available. Refer to the FA-2 instruction manual for detailed instructions on how to connect this calculator to a tape recorder.

Furthermore, since the MT described here is a CMT (cassette tape recorder) with a remote terminal, if an MT is used without a remote terminal, the MT must be operated manually according to this equipment's operation.

Also, start and stop for recording and playback can be remote controlled but fast forward and rewind cannot be remote controlled. (Hereafter referred to as CMT.)

• Program SAVE/LOAD

The command for saving a program is shown below.

SAVE [#*n* **"filename"**] (*n* is 0 to 9, items enclosed in brackets can be omitted)

#*n* indicates the program area. If it is #0, the program in P0 will be stored. If it is #3, the program in P3 will be stored.

Up to 8 identical characters composed of letters, numbers or symbols can be used for a file name.

Example: "A"
"BBB"
"********"

This #*n* and file name can be omitted. If #*n* is omitted, the program in the program area being used when the SAVE command is executed will be stored.
Furthermore, the program stored using this SAVE command is the program in the designated program area. If all the programs written in all program areas (P0 to P9) are to be stored at the same time, use the SAVE ALL command (See page 77).

Operation procedure:

(1) Mount a tape and note the counter number.
(2) Start the CMT in the record position.
(3) Operate SAVE [#*n* "filename"] EXE .
(4) When program recording is complete, the tape will stop automatically. (Stop the tape manually if the CMT does not have a remote control terminal.)

* Since the SAVE command can only be executed manually, it cannot be used by writing it in the program.

The command to callout a program recorded on the CMT is shown below.

LOAD [#*n* **"filename"**] (*n* is 0 to 9, items enclosed in brackets can be omitted)

#*n* indicates the program area of the program to be called out. If it is P1 the program will be loaded in P1. The file name is similar to SAVE command. It can be omitted. If omitted, the program written initially after start will be loaded.

Also, if #*n* is omitted, the program in the program area being used when the LOAD command is executed will be loaded.

This LOAD command can be accomplished manually or written in the program.

Operating procedure:

When performed manually:

(1) Mount the tape just prior to the designated position (2 to 3 numbers before) using the counter.
(2) Start the CMT with a remote terminal using the playback position.
(3) Operate LOAD [#*n* "filename"] EXE .
(4) For a CMT without a remote terminal, start it following the LOAD command in the playback position.
(5) When program callout is complete, the tape will stop automatically. (Stop the tape manually if the CMT does not have a remote control terminal.)

Display during program load . . . . . .

```
PF: AAA
```

program file —— file name

Display when program load is complete . . . . . .

```
READY Pn
```

* Even if a previous program remains in a program area where you want to load a program, the program on the tape can be loaded correctly. In this case, the old program contents will be cancelled from the initial line number of the new program on and the new program will be loaded.

Example:

Program remaining in P0

```
 10  INP A
      ⟨
 90  PRT B
100  FOR I=1 TO 100
      ⟨
300  PRT C
```

+

Newly loaded program

```
100  INP X
      ⟨
200  PRT Y
```

→

Program after LOAD

```
 10  INP A
      ⟨
 90  PRT B
100  INP X
      ⟨
200  PRT Y
```

For writing the LOAD command in a program:

(1) Write the LOAD command in the program.
(2) Start the CMT with a remote terminal using the playback position.
(3) The program with the LOAD command written in is started in the RUN mode (MODE 0).
(4) For a CMT without a remote terminal, start it following the program start using the playback position.
(5) When program callout is complete, the tape will stop automatically and the loaded program will start.

When the LOAD command is executed by writing in the program, after LOAD is completed, the written program will be executed in sequence.

Example:

Execution program	Filename "A"	Filename "B"	Filename "C"
P0: 10 LOAD#0"A"	100 INP X	100 INP Y	100 INP Z
20 LOAD#0"B"	⟨	⟨	⟨
30 LOAD#0"C"	500 GOTO 20	500 GOTO 30	500 END

For this kind of program, first, if line number 10 is executed, the program with filename "A" will be read. Upon completion, program "A" will be executed. Upon completion of program "A", control returns to line 20 and the program with file name "B" will be read and executed. In this manner, when written in the program, read and execution will be accomplished in sequence.

- **Data PUT/GET**

Data storage is performed using the PUT command.

PUT ["filename"] variable (items enclosed in brackets can be omitted)

The variables are $,A and B to T9.

To store variables from A to Z, write "A , Z".

Examples: To SAVE $,A,B,C,D . . . . . .PUT ["filename"] $, A , D
To SAVE A to Z . . . . . .PUT ["filename"] A , Z
To SAVE A to Z and A0 to T9 . . .PUT ["filename"] A , T9
or
PUT ["filename"] A , A(199)

* For variable writing in this case, "A , Z" means from A to Z. Since storage is from A to Z in sequence, larger to smaller sequence, such as Z to A cannot be written.

Example: PUT H , A ——————→ error

Data callout is performed using the GET command.

GET ["filename"] variable (items enclosed in brackets can be omitted)

This variable is written as "$, A" or "$, A , Z" in the similar way to the PUT example.

PUT command and GET command can be executed manually or written in the program.

Manual operation procedure can be performed similar to program SAVE and LOAD.

For execution by writing into the program, start and stop can be accomplished automatically for a CMT with a remote terminal. However, for a CMT without a remote terminal, it is necessary to use good timing.

Example: To store

```
490 PUT "D" A , Z
500 END
```

To callout

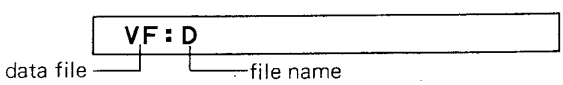
```
10 GET "D" A , Z
```

For a program as shown above, first, data storage is accomplished as shown below.

(1) Mount the tape and note the counter number.
(2) Start the CMT in the record position.
(3) Start the program with the PUT command written.
(4) When the PUT command is read, the tape will start and will stop automatically upon completion.

Then, data callout is accomplished as shown below.
(1) Mount the tape slightly before the designated position using the tape counter number.
(2) Start the CMT in the playback position.
(3) Start the program with the GET command written in.
(4) When the GET command is read, the tape will start and will stop automatically upon completion.

Display during LOAD . . . . . .



Furthermore, the file name can be omitted similar to program SAVE/LOAD.

- **SAVE/LOAD of entire program/data**

For the previous SAVE/LOAD, the program is written in one program area and called out. However, even for 1 program which is composed and spread over several program areas such as P0 to P3 or P2 to P6, or for the data required for the program, the entire program area and data can be saved/loaded.

The file commands for this entire memory are shown below.

SAVE ALL ["filename"]
LOAD ALL ["filename"] (items enclosed in brackets can be omitted)

This operation is performed similar to program SAVE/LOAD.

Display during entire memory file LOAD . . . . . PVF:ZZ

entire memory file — filename

* If this LOAD ALL command is executed, since all of the programs and data stored on the tape will be newly read in, the DEFM command is also used to expand tha data memory when LOAD ALL is executed.

- **File check written on tape**

To check the accuracy of the program or data written on the tape, a VER command is used.

VER ["filename"] (items enclosed in brackets can be omitted)

The operating procedure is similar to program LOAD.

■ Printer

A mini-printer (FP-10) can be connected to this equipment. If this printer is connected, program list and data list can be obtained. Also, calculation result output, etc., at the time of execution can be printed.

Furthermore, refer to the FP-10 instruction manual for connection and operation procedures.

To print program list or data list, press [MODE] [7] and PRT mode ("PRT" display) will be selected.
Also, for calculation result print, the PRT mode is designated manually or [MODE] [7] can be written in the program and partial printing can be accomplished.

PRT mode cancellation can be manually accomplished by pressing [MODE] [8] or accomplished by writing [MODE] [8] in the program.

- **Program list**

To print a program list, execute the LIST command in the RUN mode.

LIST [{**Line number** / **#0 to 9**}] [**"password"**] (items enclosed in brackets [ ] can be omitted)

When the line number is written, the list will be printed from that line number. If #0 to 9 is written, that program areas (P0 to P9) list will be printed.
Furthermore, if a password is added, the password is required. However, if there is no password, it is not required.

- **Data list**

To print the data list, execute the LIST command in the RUN mode.

LIST V

Using this command, all data (variable) contents will be printed.

- **Entire program/data list**

To print the P0 to P9 program list and data list at the same time, execute the LIST ALL command in the RUN mode.

LIST ALL

The LIST ALL command prints the entire P0 to P9 program list. However, print areas having no input will not be printed.
Also, for programs with a password, PASS will be printed.

- **Calculation results**

For printing of calculation results, if the program is executed following PRT mode selection, after that, the output results will be printed at the same time they are displayed using PRT statement and DMS statement.
Also, if the PRT mode designation is written in the program, only the result desired to be printed will be printed.
Furthermore, for calculation result output, if the designation using the CSR function is prior to the present print position, since the print head cannot move backwards, printing will be continued.

Error Message List

Error code	Meaning	Cause	Corrective measure
1	Memory overflow or system stack overflow.	• Number of steps insufficient. Program cannot be written in. • Stack overflow	• Clear unneeded programs or reduce the number of memories. • Divide the formulas and make them simpler.
2	Syntax error	• Format error in program, etc. • Left-hand and right-hand formats differ in an assignment statement, etc.	• Correct error in input program, etc.
3	Mathematical error	• The result of a numeric expression calculation exceeds 10^{100} • Outside the numeric function argument input range • Result is indefinite or impossible	• Correct the calculation formula or data • Verify the data
4	Undefined line number error	• No designated line number for GOTO statement or GOSUB statement	• Correct the designated line number
5	Argument error	• For a command function that requires an argument, the argument is outside the input range. • The array argument is outside 0 to 199 in the one-dimensional array or outside (0, 0) to (19, 9) in the two-dimensional array.	• Correct the argument error
6	Variable error	• A variable is used which is not defined by the DEFM command.	• Increase the variable if necessary.
7	Nesting error	• RET statement comes out when subroutine is not being executed • NEXT statement comes out when not in FOR loop • Subroutine levels exceed 10 levels • FOR•NEXT loops exceed 8 levels	• Remove unneeded RET statements or NEXT statements • Keep the subroutines or FOR•NEXT statement loops within required levels
8	Password error	• A password is set and an unusable command such as CLR or LIST was used • A password is set and a different password was input	• Input the correct password
9	Option error	• No printer or CMT is connected and execution is attempted in the PRT mode or optional command such as SAVE is executed	• Connect printer or CMT • Cancel PRT mode

Program Command List

Classification	Command name	Format	Function
Input statement	INP	INP variable string	Causes data to be entered from the keyboard during execution of a program. The program execution is stopped until after the end of input. **P. 57**
	KEY	Character variable = KEY	Reads a character entered during execution of a program and assigns it to a character variable. Since the program is not stopped by this command, nothing is assigned to the character variable if no key-in entry is made. **P. 58**
Output statement	PRT	PRT output control function {;, output element [{;,} ....]	Outputs a specified output element in a specified format. **P. 58**
	DMS	DMS variable	Converts a variable value to the sexagesimal notation, and displays it. **P. 59**
	WAIT	WAIT n * $0 \leqq n < 1000$	Determines the display time using a PRT statement or a DMS statement **P. 59**
Branching	GOTO	GOTO {line number, variable}	Causes control to jump to the specified line number. **P. 39**
	IF...{THEN, ;}...	IF comparison {THEN line number, ; command}	Causes control to jump to the line number following THEN, or executes the command following " ; ", if the result of the comparison is true. Causes control to proceed to the next line number if the result of the comparison is false. **P. 44**
	GSB	GSB {line number, variable}	Calls the subroutine with the specified line number for execution. After the subroutine is executed, control returns to the GSB statement by the RET statement to proceed to the command following that statement. **P. 64**
	RET	RET	*Signifies the end of a subroutine; returns control to a line number next to the GSB statement.* **P. 64**

Classification	Command name	Format	Function
Looping	FOR	FOR $v = e_1$ TO e_2 [STEP e_3] * v denotes a numerical variable, and e_1, e_2 and e_3 represent a numerical expression respectively.	Declares the beginning of a loop in which numerical value v changes from initial value e_1 to terminal value e_2 in increments of e_3. The loop is repeated " $\left[\frac{e_2 - e_1}{e_3}\right] + 1$ " times between the FOR and NEXT statements. If the increment e_3 is omitted, e_3 is regarded as " 1 ". **P. 49**
	NEXT	NEXT	Signifies the end of a FOR loop. If the result of v plus e_3 is equal to or smaller than e_2, the loop is repeated again. If it is greater than e_2, control proceeds to the line next to the NEXT statement. **P. 49**
Execution stop	STOP	STOP	Stops the execution of a program temporarily to bring the system into a key-in wait state. The execution can be continued by pressing the [CONT] key. **P. 37**
Execution end	END	END	Signifies the end of a program, the system returning to its pre-execution state. The execution of a program, once ended, cannot be continued even if the [CONT] key is pressed. **P. 30**
Data clearing	VAC	VAC	Clears all variable data for a program. **P. 42**
Program listing	LIST	LIST [line number]	Displays a listing of all the statements in a program from the specified line number downward. **P. 32**
Data listing	LIST V	LIST V	Displays a listing of the data in memories. **P. 78**
Program/data listing	LIST ALL	LIST ALL	Displays a listing of all the statements in a program and the data in memories. **P. 78**
Program execution	RUN	RUN [line number]	Causes a program to start from the specified line number. **P. 31**
Program erasing	CLR	CLR	Clears the currently specified program area of a program. **P. 30**
	CLR ALL	CLR ALL	Clears all the programs. **P. 30**
Program protection	PASS	PASS "password"	Upon entry of a password, such operations as LIST and LOAD cannot be performed. If a password has already been entered, it is cancelled. **P. 74**

Specifications

■ **Type**

FX-702P

■ **Basic calculation functions**

Negative numbers, exponents, arithmetic operations including parentheses (addition, subtraction, multiplication and division, with priority sequence discrimination function — true algebraic logic)

■ **Built-in functions**

Trigonometric and inverse trigonometric functions (unit of angular measure in degrees, radians or gradients), hyperbolic and inverse hyperbolic functions, logarithmic and exponential functions, factorial, square root, powers, coordinate conversion, integerize, remove integer portion, absolute value, symbolize, decimal ↔ sexagesimal conversion, effective number of digits designation, decimal designation, random number, π

■ **Statistical calculation functions**

standard deviation: number of data, sum, square sum, mean, standard deviation (2 kinds)

linear regression: number of data, sum of x, sum of y, square sum of x, square sum of y, data product sum, mean of x, mean of y, standard deviation of x (2 kinds), standard deviation of y (2 kinds), constant term, regression coefficient, correlation coefficient, estimated value of x, estimated value of y

■ **Commands**

INP, KEY, PRT, DMS, IF-THEN, GOTO, FOR, NEXT, GSB, RET, WAIT, STAT, DEL, SAC, MODE, SET, STOP, END, SAVE, SAVE ALL, LOAD, LOAD ALL, GET, PUT, VER, PASS, RUN, LIST, LIST V, LIST ALL, CLR, CLR ALL

■ **Character functions**

LEN, MID

■ **Output control function**

CSR

■ **Calculation range**

$\pm 1 \times 10^{-99}$ to $\pm 9.999999999 \times 10^{99}$ and 0

Internal calculation uses 12 digit mantissa

Functional digit capacity	**Input range**	**Result accuracy**
$\sin x$, $\cos x$, $\tan x$	$\|x\| < 1440°$ (8π rad, 1600gra)	10th digit ±1
$\sin^{-1} x$, $\cos^{-1} x$	$\|x\| \leqq 1$	— " —
$\tan^{-1} x$		— " —
$\sinh x$, $\cosh x$	$\|x\| \leqq 230$	— " —
$\tanh x$		— " —
$\sinh^{-1} x$	$\|x\| \leqq 10^{99}$	— " —
$\cosh^{-1} x$	$1 \leqq x \leqq 10^{99}$	— " —
$\tanh^{-1} x$	$\|x\| < 1$	— " —
$\log x$, $\ln x$	$x > 0$	— " —
e^x	$\|x\| \leqq 230$	— " —
$\sqrt{x}$	$x \geqq 0$	— " —
$x!$	$x \leqq 69$	— " —
x^y $(x \uparrow y)$	When $x < 0$, y is a natural number	— " —
R → P	$\|x\| \leqq 10^{99}$, $\|y\| \leqq 10^{99}$	— " —
P → R	$\|\theta\| < 1440°$ (8π rad, 1600gra)	— " —
decimal → sexagesimal	Within ±99999.9999999	— " —

■ **Program system**

stored system

■ **Program language**

BASIC

■ **Number of steps**
80 to maximum of 1680 steps (with power back-up)

■ **Number of built-in programs**
maximum of 10 groups (P0 to P9)

■ **Number of memories**
26 to maximum of 226 memories plus exclusive character variable ($) (with power back-up)

■ **Number of stacks**
subroutine — 10 levels
FOR•NEXT loop — 8 levels
numerical value — 10 levels
calculation elements — 20 levels

■ **Display system and method**
10-digit mantissa (including minus sign), 2-digit exponential portion, liquid crystal display, also sexagesimal display, F1, F2, ARC, HYP, RUN, WRT, STOP, DEG, RAD, GRA, TRACE, PRT situation display

■ **Display elements**
20-digit dot matrix display (liquid crystal)

■ **Main components**
C-MOS LSI and others

■ **Power consumption**
0.01 W (max.)

■ **Power source**
2 lithium batteries (CR2032).
The unit gives approximately 240 hours (approx. 200 hours with optional equipments) continuous operation on type CR2032.

■ **Auto power-off**
Power is turned off automatically approximately 8 minutes after last operation

■ **Ambient temperature range**
0°C to 40°C (32°F to 104°F)

■ **Dimensions**
17H x 165W x 82mmD (5/8"H x 6-1/2"W x 3-1/4"D)

■ **Weight**
176 g (6.2 oz) including batteries

GUIDELINES LAID DOWN BY FCC RULES FOR USE OF THE UNIT IN THE U.S.A. (not applicable to other areas).

A.

This equipment generates and uses radio frequency energy and if not installed and used properly, that is, in strict accordance with the manufacturer's instructions, may cause interference to radio and television reception. It has been type tested and found to comply with the limits for a Class B computing device in accordance with the specifications in Subpart J of Part 15 of FCC Rules, which are designed to provide reasonable protection against such interference in a residential installation. However, there is no guarantee that interference will not occur in a particular installation. If this equipment does cause interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

. . . reorient the receiving antenna
. . . relocate the computer with respect to the receiver
. . . move the computer away from the receiver
. . . plug the computer into a different outlet so that computer and receiver are on different branch circuits.

If necessary, the user should consult the dealer or an experienced radio/television technician for additional suggestions. The user may find the following booklet prepared by the Federal Communications Commission helpful: "How to Identify and Resolve Radio-TV Interference Problems". This booklet is available from the US Government Printing Office, Washington, D.C., 20402, Stock No. 004-000-00345-4.

B. ***When connected with the mini-printer FP-10 (on sale in the near future).***

WARNING: This equipment generates, uses, and can radiate radio frequency energy and if not installed and used in accordance with the instructions manual, may cause interference to radio communications. As temporarily permitted by regulation it has not been tested for compliance with the limits for Class A computing devices pursuant to Subpart J of Part 15 of FCC Rules, which are designed to provide reasonable protection against such interference.

Operation of this equipment in a residential area is likely to cause interference in which case the user at his own expense will be required to take whatever measures may be required to correct the interference.

CASIO®

111C SA (英) Printed in Japan