# Chapter 8

# Programming

This unit comes with approximately 144 kbytes of memory.

- You can check how much memory has been used and how much remains by entering the SYSTEM Mode from the Main Menu, and then pressing F1 (Mem). See “9-2 Memory Operations” for details.

# 8-1 Basic Programming Steps

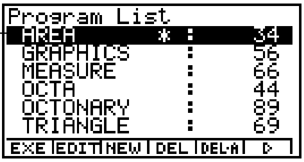
## Description

Commands and calculations are executed sequentially, just like manual calculation multistatements.

## Set Up

1. From the Main Menu, enter the PRGM Mode. When you do, a program list appears on the display.

Selected program area
(use ▲ and ▼ to move)



```
Program List
 AREA      * :    34
 GRAPHICS    :    56
 MEASURE     :    66
 OCTA        :    44
 OCTONARY    :    89
 TRIANGLE    :    69
EXE |EDIT|NEW |DEL |DEL·A| ▷
```

Files are listed in the alphabetic sequence of their names.

## Execution

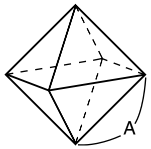
2. Register a file name.
3. Input the program.
4. Run the program.

# If there are no programs stored in memory when you enter the PRGM Mode, the message "**No Programs**" appears on the display and only the NEW item (F3) is shown in the function menu.
# The values to the right of the program list indicate the number of bytes used by each program.
# A file name can be up to eight characters long.
# The following are the characters you can use in a file name:
A through Z, $r$, $\theta$, spaces, [, ], {, }, ', ", ~, 0 through 9, ., +, –, ×, ÷
# Registering a file name uses 24 bytes of memory.
# The file name input screen remains on the display if you press EXE without inputting a file name.
# To exit the file name input screen and return to the program list without registering a file name, press ESC.

● ● ● ● ●

**Example 1 To calculate the surface area ($cm^2$) and volume ($cm^3$) of three regular octahedrons when the length of one side is 7, 10, and 15 cm, respectively.**

Store the calculation formula under the file name OCTA.



The following are the formulas used for calculating surface area S and volume V of a regular octahedron for which the length of one side A is known.

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

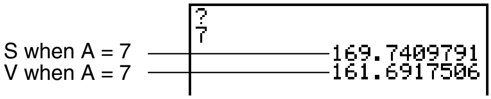
## Procedure

① MENU PRGM

② F3 (NEW) O C T A EXE [*1]

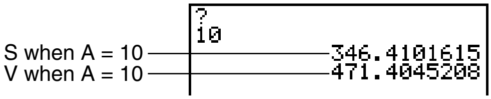
③ SHIFT VARS (PRGM) F3 (?) → ALPHA X,θ,T (A) F6 (▷) F6 (▷) F3 (:) [*2]
2 × SHIFT $x^2$ (√ ) 3 × ALPHA X,θ,T (A) $x^2$ F6 (▷) F4 (◢)
SHIFT $x^2$ (√ ) 2 ÷ 3 × ALPHA X,θ,T (A) ^ 3
ESC ESC

④ F1 (EXE) or EXE
7 EXE (Value of A)
EXE



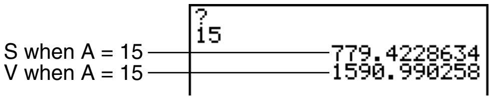
```
?
7
                169.7409791
                161.6917506
```

EXE
EXE 1 0 EXE
EXE



```
?
10
                346.4101615
                471.4045208
```

EXE
EXE 1 5 EXE
EXE [*3]



```
?
15
                779.4228634
                1590.990258
```

[*1] Press F3 (NEW) and the cursor changes form to indicate alpha character input.

[*2] The following shows how the calculation of the surface area and volume of a regular octahedron would be calculated using a manual calculation.

Surface Area S ... 2 × SHIFT $x^2$ (√ ) 3 × <value of A> $x^2$ EXE

Volume V ............ SHIFT $x^2$ (√ ) 2 ÷ 3 × <value of A> ^ 3 EXE

[*3] Pressing EXE while the final result of a program is on the display changes to the program list.

# You can also run a program while in the **RUN • MAT Mode** by inputting: Prog "<file name>" EXE.

# Pressing EXE while the final result of a program executed using this method is on the display re-executes the program.

# An error occurs if the program specified by Prog "<file name>" cannot be found.

# 8-2 Program Mode Function Keys

- {**NEW**} ... {new program}

## • When you are registering a file name

- {**RUN**}/{**BASE**} ... {general calculation}/{number base} program input
- {**ꟹO**} ... {password registration}
- {**SYBL**} ... {symbol menu}

## • When you are inputting a program —— F1(RUN) … default

- {**JUMP**} ... {top}/{bottom} of program
- {**SRC**} ... {search}
- {**MAT**}/ {**STAT**}/{**LIST**}/{**GRPH**}/{**DYNA**}/{**RECR**}
  ... {matrix}/{statistic}/{list}/{graph}/ {Dynamic Graph}/{recursion} menu

• Pressing SHIFT VARS (PRGM) displays the following PRGM (PROGRAM) menu.

- {**Prog**} ... {program recall}
- {**JUMP**} ... {jump command menu}
- {**?**}/{**◢**} ... {input}/{output} command
- {**I/O**} ... {I/O control/transfer command menu}
- {**IF**}/{**FOR**}/{**WHLE**}/{**CTRL**}/{**LOGIC**}
  ... {conditional jump}/{loop control}/{conditional loop control}/{program control}/ {logical operation} command menu
- {**CLR**}/{**DISP**} ... {clear}/{display} command menu
- {**:**} ......... {separator for expressions and commands}

See “8-5 Command Reference” for full details on each of these commands.

• Pressing CTRL F3 (SET UP) displays the mode command menu shown below.

- {**ANGL**}/{**DISP**}/{**CPLX**}/{**GRPH**}/{**STAT**}/{**DERIV**}/{**T-VAR**}/{**Σ DSP**}

See “SET UP Screen Function Key Menus” on page 1-7-1 for details about each of these commands.

## • When you are inputting a program —— F2(BASE)*1

- {**JUMP**}/{**SRC**}
- {**d~o**} ... {decimal}/{hexadecimal}/{binary}/{octal} value input
- {**LOG**} ... {logical operators}
- {**DISP**} ... conversion of displayed value to {decimal}/{hexadecimal}/{binary}/{octal}
- {**SYBL**} ... {symbol menu}

• Pressing SHIFT VARS (PRGM) displays the following PRGM (PROGRAM) menu.

- {**Prog**}/{**JUMP**}/{**?**}/{**◢**}
- {**= ≠ <**} ... {logical operator menu}
- {**:**} ......... {separator for expressions and commands}

• Pressing CTRL F3 (SET UP) displays the mode command menu shown below.

- {**Dec**}/{**Hex**}/{**Bin**}/{**Oct**}

- {**EXE**}/{**EDIT**}
  ... program {execute}/{edit}
- {**NEW**} ... {new program}
- {**DEL**}/{**DEL·A**}
  ... {specific program}/{all program} delete
- {**SRC**}/{**REN**}
  ... file name {search}/{change}

*1 Programs input after pressing F2(BASE) are indicated by B to the right of the file name.

# 8-3 Editing Program Contents

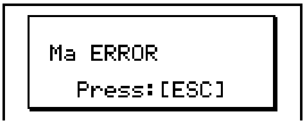
## ■ Debugging a Program

A problem in a program that keeps the program from running correctly is called a "bug," and the process of eliminating such problems is called "debugging." Either of the following symptoms indicates that your program contains bugs that require debugging.

- Error messages appearing when the program is run
- Results that are not within your expectations

### • To eliminate bugs that cause error messages

An error message, like the one shown below, appears whenever something illegal occurs during program execution.



```
Ma ERROR
  Press:[ESC]
```

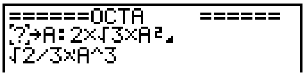
When such a message appears, press ESC to display the place in the program where the error was caused. The cursor will be flashing at the location of the problem. Check the "Error Message Table" (page $\alpha$-1-1) for steps you should take to correct the situation.

- Note that pressing ESC does not display the location of the error if the program is password protected. Instead, it returns to the program list screen.

### • To eliminate bugs that cause bad results

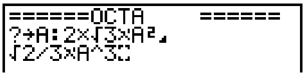
If your program produces results that are not what you normally expect, check the contents of the program and make necessary changes.
The F1(JUMP) key is also useful when editing program contents.

F1(JUMP) 1(Top) ....... Moves the cursor to the top of the program



```
======OCTA    ======
?→A:2×√3×A²◢
√2/3×A^3
```

F1(JUMP) 2(Bottom)… Moves the cursor to the bottom of the program

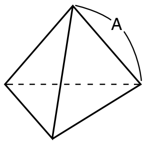


```
======OCTA    ======
?→A:2×√3×A²◢
√2/3×A^3
```

## ■ Using an Existing Program to Create a New Program

Sometimes you can input a new program by using a program already in memory as a base. Simply recall the existing program, make the changes you need, and then execute it.

●●●●●
**Example 2 To use the OCTA program (page 8-1-2) to create a program that calculates the surface area (cm$^2$) and volume (cm$^3$) of regular tetrahedrons when the length of one side is 7, 10, and 15 cm**
Use TETRA as the file name.



The following are the formulas used for calculating surface area S and volume V of a regular tetrahedron for which the length of one side A is known.

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

Use the following key operations when inputting the program.

Length of One Side A .. SHIFT VARS (PRGM) F3 (?) → ALPHA X,θ,T (A) F6 (▷) F6 (▷) F3 (:)
Surface Area S ............ SHIFT $x^2$ (√‾) 3 × ALPHA X,θ,T (A) $x^2$ F6 (▷) F4 (◢)
Volume V ..................... SHIFT $x^2$ (√‾) 2 ÷ $\underline{\text{1 2}}$ × ALPHA X,θ,T (A) ^ 3

Compare this with the program for calculating the surface area and volume of a regular octahedron.

Length of One Side A .. SHIFT VARS (PRGM) F3 (?) → ALPHA X,θ,T (A) F6 (▷) F6 (▷) F3 (:)
Surface Area S ............ $\utilde{\text{2 ×}}$ SHIFT $x^2$ (√‾) 3 × ALPHA X,θ,T (A) $x^2$ F6 (▷) F4 (◢)
Volume V ..................... SHIFT $x^2$ (√‾) 2 ÷ $\underline{\text{3}}$ × ALPHA X,θ,T (A) ^ 3

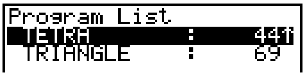
As you can see, you can produce the TETRA program by making the following changes in the OCTA program.

- Deleting 2 × (underlined using a wavy line above)
- Changing 3 to 1 2 (underlined using a solid line above)

Now edit OCTA to produce the TETRA program.

1. Edit the program name.

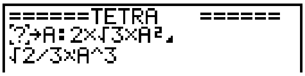
   F6(▷) F2(REN) AC T E T R A EXE



```
Program List
 TETRA          :    441
 TRIANGLE       :     69
```

2. Edit the program contents.

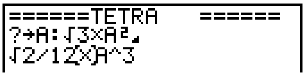
   F2(EDIT)



```
======TETRA   ======
?→A:2×√3×A²◢
√2/3×A^3
```

   ▶ ▶ ▶ ▶ DEL DEL

   ▼ ◀ DEL 1 2



```
======TETRA   ======
?→A:√3×A²◢
√2/12×A^3
```

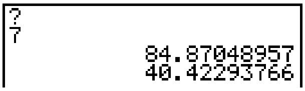
   ESC

3. Try running the program.

   F1(EXE) or EXE

   7 EXE(Value of A)

   EXE

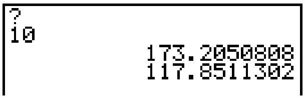


```
?
7
             84.87048957
             40.42293766
```

   EXE

   EXE 1 0 EXE

   EXE

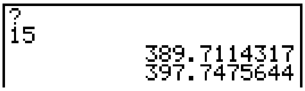


```
?
10
             173.2050808
             117.8511302
```

   EXE

   EXE 1 5 EXE

   EXE



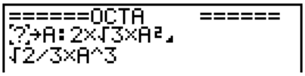
```
?
15
             389.7114317
             397.7475644
```

## ■ Searching for Data Inside a Program

● ● ● ● ●
**Example To search for the letter "A" inside the program named OCTA**

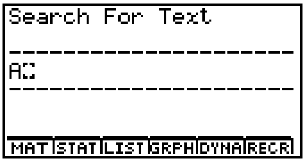
1. Recall the program.

2. Press F2(SRC) or EXE and input the data you want to find.



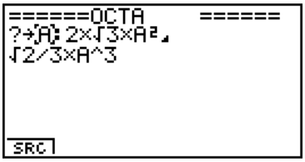
```
======OCTA     ======
?→A:2×√3×A²◢
√2/3×A^3
```

F2(SRC)
ALPHA X,θ,T(A)



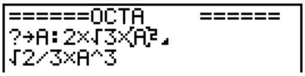
```
Search For Text
---------------------
A
---------------------
MAT STAT LIST GRPH DYNA RECR
```

3. Press EXE to begin the search. The contents of the program appear on the screen with the cursor located at the first instance of the data you specified.*1



```
======OCTA     ======
?→A:2×√3×A²◢
√2/3×A^3
SRC
```

4. Each press of EXE or F1(SRC) causes the cursor to jump to the next instance of the data you specified.*2



```
======OCTA     ======
?→A:2×√3×A²◢
√2/3×A^3
```

*1 The message "**Not Found**" appears when the search data you specify cannot be found in the program.

*2 If there are no more instances of the data you specified, the search operation ends and the cursor returns to the point from which you started your search.

# You cannot specify the newline symbol (↵) or display command (◢) for the search data.

# Once the contents of the program are on the screen, you can use the cursor keys to move the cursor to another location before searching for the next instance of the data. Only the part of the program starting from the current cursor location is searched when you press EXE.

# Once the search finds an instance of your data, inputting characters or moving the cursor causes the search operation to be cancelled.

# If you make a mistake while inputting characters to search for, press AC to clear your input and re-input from the beginning.

# 8-4 File Management

## ■ Searching for a File

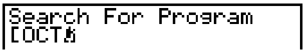
### • To find a file using initial character search

● ● ● ● ●

**Example To use initial character search to recall the program named OCTA**

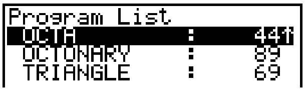
1. While the program list is on the display, press F6(▷)F1(SRC) and input the initial characters of the file you want to find.

   F6(▷) F1(SRC)
   O C T



```
Search For Program
[OCT
```

2. Press EXE to search.



```
Program List
 OCTA         :   44
 OCTONARY     :   89
 TRIANGLE     :   69
```

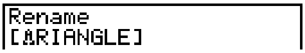
• The name that starts with the characters you input highlights.

# If there is no program whose file name starts with the characters you input, the message "**Not Found**" appears on the display. If this happens, press ESC to clear the error message.

## Editing a file name

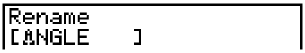
●●●●●
**Example To change the name of a file from TRIANGLE to ANGLE**

1. While the program list is on the display, use ▲ and ▼ to move the highlighting to the file whose name you want to edit and then press F6 (▷) F2 (REN).


```
Rename
[ARIANGLE]
```

2. Make any changes you want.

   DEL DEL DEL


```
Rename
[ANGLE    ]
```

3. Press EXE to register the new name and return to the program list.

The program list is resorted according to the changes you made in the file name.

## Deleting a Program

### • To delete a specific program

1. While the program list is on the display, use ▲ and ▼ to move the highlighting to the name of the program you want to delete.

2. Press F4 (DEL).

3. Press EXE (Yes) to delete the selected program or ESC (No) to abort the operation without deleting anything.

# If the modifications you make result in a file name that is identical to the name of a program already stored in memory, the message "**Already Exists**" appears. When this happens, you can perform either of the following two operations to correct the situation.

- Press ESC to clear the error and return to the file name editing screen.
- Press AC to clear the input file name and input a new one.

**• To delete all programs**

1. While the program list is on the display, press F5(DEL·A).

2. Press EXE(Yes) to delete all the programs in the list or ESC(No) to abort the operation without deleting anything.

• You also can delete all programs by entering the SYSTEM Mode from the Main Menu, and then pressing F1(Mem) to display the memory management screen.
See "9-2 Memory Operations" for details.

## ■ Registering a password

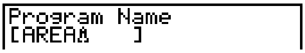
When inputting a program, you can protect it with a password that limits access to the program contents to those who know the password.

• You do not need to input the password to run a program.

● ● ● ● ●
**Example** **To create a program file under the name AREA and protect it with the password CASIO**

1. While the program list is on the display, press F3(NEW) and input the file name of the new program file.

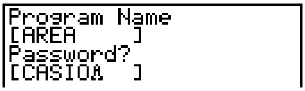
F3(NEW)
A R E A



```
Program Name
[AREA    ]
```

2. Press F5(🔑) and then input the password.

F5(🔑)
C A S I O

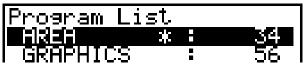


```
Program Name
[AREA    ]
Password?
[CASIO   ]
```

# The password input procedure is identical to that used for file name input.

3. Press EXE to register the file name and password. Now you can input the contents of the program file.

4. After inputting the program, press SHIFT ESC (QUIT) to exit the program file and return to the program list. Files that are password protected are indicated by an asterisk to the right of the file name.



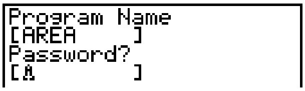
```
Program List
 AREA       * :     34
 GRAPHICS     :     56
```

## ■ Recalling a Password Protected Program

● ● ● ● ●
**Example** **To recall the file named AREA which is protected by the password CASIO**

1. In the program list, use ⏶ and ⏷ to move the highlighting to the name of the program you want to recall.

2. Press F2 (EDIT).



```
Program Name
[AREA     ]
Password?
[A        ]
```

3. Input the password and press EXE to recall the program.

# Pressing EXE without inputting a password while saving a new program causes the file to be saved without a password. Pressing EXE without inputting a password registers the file name only, without a password.

# Inputting the wrong password when recalling a password protected program causes the message "**Mismatch**" to appear. Press ESC to return to the password input screen.

# 8-5 Command Reference

## ■ Command Index

The following are conventions that are used in this section when describing the various commands.

| | |
|---|---|
| Boldface Text | Actual commands and other items that always must be input are shown in boldface. |
| {Curly Brackets} | Curly brackets are used to enclose a number of items, one of which must be selected when using a command. Do not input the curly brackets when inputting a command. |
| [Square Brackets] | Square brackets are used to enclose items that are optional. Do not input the square brackets when inputting a command. |
| Numeric Expressions | Numeric expressions (such as 10, 10 + 20, A) indicate constants, calculations, numeric constants, etc. |
| Alpha Characters | Alpha characters indicate literal strings (such as AB). |

## ■ Basic Operation Commands

### ? (Input Command)

**Function:** Prompts for input of values for assignment to variables during program execution.

**Syntax:** ? → <variable name>, "<prompt>" ? → <variable name>

**Example:** ? → A

**Description:**

- This command momentarily interrupts program execution and prompts for input of a value or expression for assignment to a variable. If you do not specify a prompt, execution of this command causes "?" to appear indicating the calculator is standing by for input. If a prompt is specified, "<prompt>?" appears to prompt input. Up to 255 bytes of text can be used for a prompt.

- Input in response to the input command must be a value or an expression, and the expression cannot be a multi-statement.

- You can specify a list name, matrix name, function memory (fn), graph (Yn), etc. as a variable name.

## ◢ (Output Command)

**Function:** Displays an intermediate result during program execution.

**Description:**

- This command momentarily interrupts program execution and displays alpha character text or the result of the calculation immediately before the command.

- The output command should be used at locations where you would normally press the EXE key during a manual calculation.

## : (Multi-statement Command)

**Function:** Connects two statements for sequential execution without stopping.

**Description:**

- Unlike the output command (◢), statements connected with the multi-statement command are executed non-stop.

- The multi-statement command can be used to link two calculation expressions or two commands.

- You can also use a carriage return indicated by ↵ in place of the multi-statement command.

## ↵ (Carriage Return)

**Function:** Connects two statements for sequential execution without stopping.

**Description:**

- Operation of the carriage return is identical to that of the multi-statement command.

- You can create a blank line in a program by inputting a carriage return only. Using a carriage return in place of the multi-statement command makes the displayed program easier to read.

## ' (Comment Text Delimiter)

**Function:** Indicates comment text inserted inside a program.

**Description:** Anything following the apostrophe is treated as non-executable comment text.

## ■ Program Commands (COM)

### If~Then~(Else~)IfEnd

**Function:** The Then-statement is executed only when the If-condition is true (non-zero). The Else-statement is executed when the If-condition is false (0). The IfEnd-statement is always executed following either the Then-statement or Else-statement.

**Syntax:**

$$\text{If} \quad \underset{\text{numeric expression}}{\underline{\text{<condition>}}} \left\{\begin{matrix}\text{↵}\\ :\\ \text{◢}\end{matrix}\right\} \text{Then <statement>} \left[\left\{\begin{matrix}\text{↵}\\ :\\ \text{◢}\end{matrix}\right\} \text{<statement>}\right]$$

$$\left\{\begin{matrix}\text{↵}\\ :\\ \text{◢}\end{matrix}\right\} \left(\text{Else <statement>} \left[\left\{\begin{matrix}\text{↵}\\ :\\ \text{◢}\end{matrix}\right\} \text{<statement>}\right] \left\{\begin{matrix}\text{↵}\\ :\\ \text{◢}\end{matrix}\right\}\right) \text{IfEnd}$$

**Parameters:** condition, numeric expression

**Description:**

(1) If ~ Then ~ IfEnd
- When the condition is true, execution proceeds with the Then-statement and then continues with the statement following IfEnd.
- When the condition is false, execution jumps to the statement following IfEnd.

(2) If ~ Then ~ Else ~ IfEnd
- When the condition is true, execution proceeds with the Then-statement and then jumps to the statement following IfEnd.
- When the condition is false, execution jumps to the Else-statement and then continues with the statement following IfEnd.

### For~To~(Step~)Next

**Function:** This command repeats everything between the For-statement and the Next-statement. The starting value is assigned to the control variable with the first execution, and the value of the control variable is changed according to the step value with each execution. Execution continues until the value of the control variable exceeds the ending value.

**Syntax:**

$$\text{For <starting value>} \rightarrow \text{<control variable name> To <ending value>} \left(\text{Step <step value>}\right) \left\{\begin{matrix}\text{↵}\\ :\\ \text{◢}\end{matrix}\right\}$$

Next

**Parameters:**

- control variable name: A to Z
- starting value: value or expression that produces a value (i.e. sin $x$, A, etc.)
- ending value: value or expression that produces a value (i.e. sin $x$, A, etc.)
- step value: numeric value (default: 1)

**Description:**

- The default step value is 1.

- Making the starting value less than the ending value and specifying a positive step value causes the control variable to be incremented with each execution. Making the starting value greater than the ending value and specifying a negative step value causes the control variable to be decremented with each execution.

## Do~LpWhile

**Function:** This command repeats specific commands as long as its condition is true (non-zero).

**Syntax:**

$$\text{Do}\ \left\{\begin{array}{c}\hookleftarrow\\ :\\ \text{◢}\end{array}\right\}\ \text{<statement>}\ \left\{\begin{array}{c}\hookleftarrow\\ :\\ \text{◢}\end{array}\right\}\ \text{LpWhile}\quad \underset{\text{numeric expression}}{\text{<}\underline{\text{condition}}\text{>}}$$

**Parameters:** expression

**Description:**

- This command repeats the commands contained in the loop as long as its condition is true (non-zero). When the condition becomes false (0), execution proceeds from the statement following the LpWhile-statement.

- Since the condition comes after the LpWhile-statement, the condition is tested (checked) after all of the commands inside the loop are executed.

### While~WhileEnd

**Function:** This command repeats specific commands as long as its condition is true (non-zero).

**Syntax:**

$$\text{While}\quad \underset{\text{numeric expression}}{\langle\underline{\text{condition}}\rangle} \left\{\begin{matrix}↵\\ :\\ ◢\end{matrix}\right\} \langle\text{statement}\rangle \left\{\begin{matrix}↵\\ :\\ ◢\end{matrix}\right\} \text{WhileEnd}$$

**Parameters:** expression

**Description:**

- This command repeats the commands contained in the loop as long as its condition is true (non-zero). When the condition becomes false (0), execution proceeds from the statement following the WhileEnd-statement.

- Since the condition comes after the While-statement, the condition is tested (checked) before the commands inside the loop are executed.

## ■ Program Control Commands (CTL)

### Break

**Function:** This command breaks execution of a loop and continues from the next command following the loop.

**Syntax:** Break

**Description:**

- This command breaks execution of a loop and continues from the next command following the loop.

- This command can be used to break execution of a For-statement, Do-statement, and While-statement.

## Prog

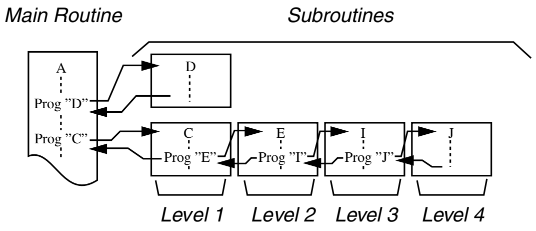
**Function:** This command specifies execution of another program as a subroutine. In the RUN•MAT Mode, this command executes a new program.

**Syntax:** Prog ”file name”

**Example:** Prog ”ABC”

**Description:**

- Even when this command is located inside of a loop, its execution immediately breaks the loop and launches the subroutine.
- This command can be used as many times as necessary inside of a main routine to call up independent subroutines to perform specific tasks.
- A subroutine can be used in multiple locations in the same main routine, or it can be called up by any number of main routines.



- Calling up a subroutine causes it to be executed from the beginning. After execution of the subroutine is complete, execution returns to the main routine, continuing from the statement following the Prog command.
- A Goto~Lbl command inside of a subroutine is valid inside of that subroutine only. It cannot be used to jump to a label outside of the subroutine.
- If a subroutine with the file name specified by the Prog command does not exist, an error occurs.
- In the **RUN•MAT Mode**, inputting the Prog command and pressing EXE launches the program specified by the command.

## Return

**Function:** This command returns from a subroutine.

**Syntax:** Return

**Description:**

Execution of the Return command inside a main routine causes execution of the program to stop. Execution of the Return command within a subroutine terminates the subroutine and returns to the program from which the subroutine was jumped to.

## Stop

**Function:** This command terminates execution of a program.

**Syntax:** Stop

**Description:**

- This command terminates program execution.

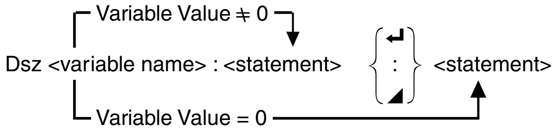
- Execution of this command inside of a loop terminates program execution without an error being generated.

## ■ Jump Commands (JUMP)

### Dsz

**Function:** This command is a count jump that decrements the value of a control variable by 1, and then jumps if the current value of the variable is zero.

**Syntax:**



**Parameters:** variable name: A to Z, $r$, $\theta$

[Example] Dsz B : Decrements the value assigned to variable B by 1.

**Description:**

This command decrements the value of a control variable by 1, and then tests (checks) it. If the current value is non-zero, execution continues with the next statement. If the current value is zero, execution jumps to the statement following the multi-statement command (:), display command (◢), or carriage return (↵).

## Goto~Lbl

**Function:** This command performs an unconditional jump to a specified location.

**Syntax:** Goto <label name> ~ Lbl <label name>

**Parameters:** label name: value (0 to 9), variable (A to Z, *r*, *θ*)

**Description:**

- This command consists of two parts: Goto *n* (where *n* is a parameter as described above) and Lbl *n* (where *n* is the parameter referenced by Goto *n*). This command causes program execution to jump to the Lbl-statement whose *n* parameter matches that specified by the Goto-statement.

- This command can be used to loop back to the beginning of a program or to jump to any location within the program.

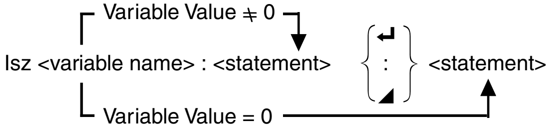
- This command can be used in combination with conditional jumps and count jumps.

- If there is no Lbl-statement whose value matches that specified by the Goto-statement, an error occurs.

### Isz

**Function:** This command is a count jump that increments the value of a control variable by 1, and then jumps if the current value of the variable is zero.

**Syntax:**



**Parameters:** variable name: A to Z, $r$, $\theta$

[Example] Isz A : Increments the value assigned to variable A by 1.

**Description:**

This command increments the value of a control variable by 1, and then tests (checks) it. If the current value is non-zero, execution continues with the next statement. If the current value is zero, execution jumps to the statement following the multi-statement command (:), display command (◢), or carriage return (↵).

## ■ Clear Commands (CLR)

### ClrGraph

**Function:** This command clears the graph screen and returns View Window settings to their INIT values.

**Syntax:** ClrGraph

**Description:** This command clears the graph screen during program execution.

### ClrList

**Function:** This command deletes list data.

**Syntax:** ClrList <list name>

ClrList

**Parameters:** list name: 1 to 20, Ans

**Description:** This command deletes the data in the list specified by “list name”. All list data is deleted if nothing is specified for “list name”.

### ClrText

**Function:** This command clears the text screen.

**Syntax:** ClrText

**Description:** This command clears text from the screen during program execution.

### ClrMat

**Function:** This command deletes matrix data.

**Syntax:** ClrMat <matrix name>
ClrMat

**Parameters:** matrix name: A to Z, Ans

**Description:** This command deletes the data in the matrix specified by "matrix name". All matrix data is deleted if nothing is specified for "matrix name".

## ■ Display Commands (DISP)

### DispF-Tbl, DispR-Tbl — No parameters

**Function:** These commands display numeric tables.

**Description:**

- These commands generate numeric tables during program execution in accordance with conditions defined within the program.
- DispF-Tbl generates a function table, while DispR-Tbl generates a recursion table.

### DrawDyna — No parameters

**Function:** This command executes a Dynamic Graph draw operation.

**Description:** This command draws a Dynamic Graph during program execution in accordance with current Dynamic Graph parameters.

## DrawFTG-Con, DrawFTG-Plt — No parameters

**Function:** This command uses values in a generated table to graph a function.

**Description:**

- This command draws a function graph in accordance with current conditions.
- DrawFTG-Con produces a connect type graph, while DrawFTG-Plt produces a plot type graph.

## DrawGraph — No parameters

**Function:** This command draws a graph.

**Description:**

- This command draws a graph in accordance with current conditions.

## DrawR-Con, DrawR-Plt — No parameters

**Function:** These commands use values in a generated table to graph a recursion expression with $a_n$($b_n$ or $c_n$) as the vertical axis and $n$ as the horizontal axis.

**Description:**

- These commands graph recursion expressions in accordance with current conditions, with $a_n$($b_n$ or $c_n$) as the vertical axis and $n$ as the horizontal axis.

- DrawR-Con produces a connect type graph, while DrawR-Plt produces a plot type graph.

## DrawRΣ-Con, DrawRΣ-Plt

**No parameters**

**Function:** These commands use values in a generated table to graph a recursion expression with $\Sigma a_n$($\Sigma b_n$ or $\Sigma c_n$) as the vertical axis and $n$ as the horizontal axis.

**Description:**

- These commands graph recursion expressions in accordance with current conditions, with $\Sigma a_n$($\Sigma b_n$ or $\Sigma c_n$) as the vertical axis and $n$ as the horizontal axis.
- DrawRΣ-Con produces a connect type graph, while DrawRΣ-Plt produces a plot type graph.

## DrawStat

**Function:** This draws a statistical graph.

**Syntax:** See "8-6-9 Using Statistical Calculations and Graphs in a Program".

**Description:**

This command draws a statistical graph in accordance with current statistical graph conditions.

## DrawWeb

**Function:** This command graphs convergence/divergence of a recursion expression (WEB graph).

**Syntax:** DrawWeb <recursion type>, <number of lines>

**Example:** DrawWeb $a_{n+1}$ ($b_{n+1}$ or $c_{n+1}$), 5

**Description:**

- This command graphs convergence/divergence of a recursion expression (WEB graph).
- Omitting the number of lines specification automatically specifies the default value 30.

## ■ Input/Output Commands (I/O)

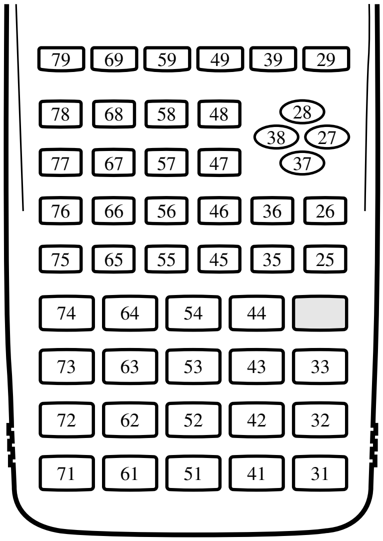
### Getkey

**Function:** This command returns the code that corresponds to the last key pressed.

**Syntax:** Getkey

**Description:**

- This command returns the code that corresponds to the last key pressed.



- A value of zero is returned if no key was pressed previous to executing this command.

- This command can be used inside of a loop.

## Locate

**Function:** This command displays alpha-numeric characters at a specific location on the text screen.

**Syntax:** Locate <column number>, <line number>, <value>

Locate <column number>, <line number>, <numeric expression>

Locate <column number>, <line number>, ”<string>”

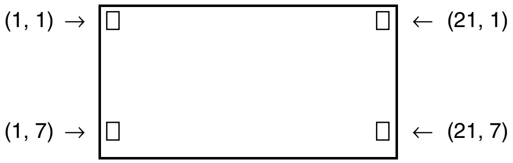
[Example] Locate 1, 1, ”AB”↵

**Parameters:**

- line number: number from 1 to 7
- column number: number from 1 to 21
- value and numeric expression
- string: character string

**Description:**

- This command displays values (including variable contents) or text at a specific location on the text screen. If there is a calculation input, that calculation result is displayed.
- The line is designated by a value from 1 to 7, while the column is designated by a value from 1 to 21.



**Example:** Cls↵

Locate 7, 1, ”CASIO FX”

This program displays the text “CASIO FX” in the center of the screen.

- In some cases, the ClrText command should be executed before running the above program.

## Receive ( / Send (

**Function:** This command receives data from and sends data to a connected device.

**Syntax:** Receive (<data>) / Send (<data>)

**Description:**

- This command receives data from and sends data to a connected device.

- The following types of data can be received (sent) by this command.
  - Individual values assigned to variables
  - Matrix data (all values - individual values cannot be specified)
  - List data (all values - individual values cannot be specified)

## ■ Conditional Jump Relational Operators (REL)

| =, ≠, >, <, ≥, ≤ |
|---|

**Function:** These relational operators are used in combination with the conditional jump command.

**Syntax:**

<left side> <relational operator> <right side>

**Parameters:**

left side/right side: variable (A to Z, $r$, $\theta$), numeric constant, variable expression (such as: A × 2)

relational operator: =, ≠, >, <, ≥, ≤

# 8-6 Using Calculator Functions in Programs

## ■ Text Display

You can include text in a program by simply enclosing it between double quotation marks. Such text appears on the display during program execution, which means you can add labels to input prompts and results.

| **Program** | **Display** |
|---|---|
| ”CASIO” | CASIO |
| ? → X | ? |
| ”X =” ? → X | X = ? |

- If the text is followed by a calculation formula, be sure to insert a display command (◢) between the text and calculation.
- Inputting more than 21 characters causes the text to move down to the next line. The screen scrolls automatically if the text exceeds 21 characters.
- You can specify up to 255 bytes of text for a comment.

## ■ Using Matrix Row Operations in a Program

These commands let you manipulate the rows of a matrix in a program.

- For this program, enter the **RUN • MAT Mode** and then use the MAT Editor to input the matrix, and then enter the **PRGM Mode** to input the program.

### • To swap the contents of two rows (Swap)

● ● ● ● ●
**Example 1 To swap the values of Row 2 and Row 3 in the following matrix:**

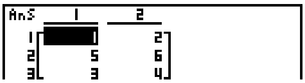
$$\textbf{Matrix A} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}$$

The following is the syntax to use for this program.

Swap $\underbrace{A}_{\textit{Matrix name}}$, $\underbrace{2, 3}_{\textit{Rows to be swapped}}$ ↵

Mat A

Executing this program produces the following result.



```
Ans     1       2
  1[    1       2]
  2     5       6
  3     3       4]
```

## • To calculate a scalar multiplication (∗Row)

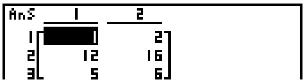
● ● ● ● ●

**Example 2 To calculate the product of Row 2 of the matrix in Example 1 and the scalar 4**

The following is the syntax to use for this program.

```
∗Row 4, A, 2↵
     |  |  └ Row
     |  └──── Matrix name
     └─────── Multiplier
Mat A
```

Executing this program produces the following result.



```
Ans    1     2
 1 [         2 ]
 2 [   12   16 ]
 3 [    5    6 ]
```

## • To calculate a scalar multiplication and add the results to another row (∗Row+)

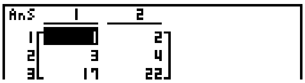
● ● ● ● ●

**Example 3 To calculate the product of Row 2 of the matrix in Example 1 and the scalar 4, then add the result to row 3**

The following is the syntax to use for this program.

```
∗Row+ 4, A, 2, 3↵
      |  |  |  └ Rows to be added
      |  |  └─── Row for which scalar multiplication is to be calculated.
      |  └────── Matrix name
      └───────── Multiplier
Mat A
```

Executing this program produces the following result.



```
Ans    1     2
 1 [         2 ]
 2 [    3    4 ]
 3 [   17   22 ]
```

### • To add two rows (Row+)

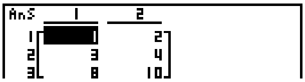
● ● ● ● ●

**Example 4 To add Row 2 to Row 3 of the matrix in Example 1**

The following is the syntax to use for this program.

```
Row+  A, 2, 3 ↵
      │  │  └──── the row number to be added to
      │  └─────── the row number to be added
      └────────── Matrix name
Mat A
```

Executing this program produces the following result.



```
Ans    1     2
 1 [   1     2 ]
 2 [   3     4 ]
 3 [   8    10 ]
```

## ■ Using Graph Functions in a Program

You can incorporate graph functions into a program to draw complex graphs and to overlay graphs on top of each other. The following shows various types of syntax you need to use when programming with graph functions.

• View Window

View Window –5, 5, 1, –5, 5, 1 ↵

• Graph function input

Y = Type ↵ ..................... Specifies graph type.

”$X^2$ – 3” → Y1 ↵

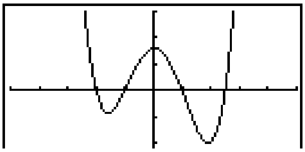
• Graph draw operation

DrawGraph ↵

**Example Program**

| | |
|---|---|
| ① ClrGraph ↵ | ① SHIFT VARS F6 F6 F1 2 ESC |
| ② View Window –10, 10, 2, –120, 150, 50 ↵ | ② SHIFT OPTN F1 ESC |
| ③ Y = Type ↵ | ③ F6 F1 3 1 |
| ”X^4–X^3–24$X^2$ + 4X + 80” ➡ $\underset{④}{\underline{\text{Y1}}}$ ↵ | ④ VARS F4 1 ESC |
| ⑤ G SelOn 1 ↵ | ⑤ F6 F1 1 |
| ⑥ DrawGraph | ⑥ SHIFT VARS F6 F6 F2 2 |

Executing this program produces the result shown here.



## • Syntax of other graphing functions

• V-Window

View Window <Xmin>, <Xmax>, <Xscale>, <Ymin>, <Ymax>, <Yscale>, <T$\theta$min>, <T$\theta$max>, <T$\theta$pitch>

StoV-Win <area of V-Win> .............. area: 1 to 6

RclV-Win <area of V-Win> .............. area: 1 to 6

• Zoom

Factor <X factor>, <Y factor>

ZoomAuto ........... Non-parameter

• Pict

StoPict <area of picture> ................ area: 1 to 20

RclPict <area of picture> ................ area: 1 to 20

• Sketch

PlotOn <X-coordinate>, <Y-coordinate>

PlotOff <X-coordinate>, <Y-coordinate>

PlotChg <X-coordinate>, <Y-coordinate>

PxlOn<line number>, <column number>

PxlOff<line number>, <column number>

PxlChg<line number>, <column number>

PxlTest( <line number>, <column number>[)]

F-Line <X-coordinate 1>, <Y-coordinate 1>, <X-coordinate 2>, <Y-coordinate 2>

Text <line number>, <column number>, ”<text>”

Text <line number>, <column number>, <expression>

Tangent <function>, <X-coordinate>

Normal <function>, <X-coordinate>

Inverse <function>

Circle <center point X-coordinate>, <center point Y-coordinate>, <radius R value>

Vertical <X-coordinate>

Horizontal <Y-coordinate>

## Using Dynamic Graph Functions in a Program

Using Dynamic Graph functions in a program makes it possible to perform repeated Dynamic Graph operations. The following shows how to specify the Dynamic Graph range inside a program.

**• Dynamic Graph range**

1 → D Start↵

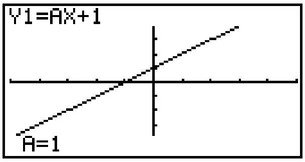
5 → D End↵

1 → D pitch↵

**Example Program**

| | |
|---|---|
| ClrGraph↵ | |
| View Window –5, 5, 1, –5, 5, 1↵ | |
| Y = Type↵ | |
| ”AX + 1” → $\underset{①}{\underline{Y}}$1↵ | ① VARS F4 1 ESC |
| ② D SelOn 1↵ | ② F6 F2 1 |
| ③ D Var A↵ | ③ F2 3 |
| 1 → ④ D Start↵ | ④ VARS F5 1 |
| 5 → ⑤ D End↵ | ⑤ F5 2 |
| 1 → ⑥ D pitch↵ | ⑥ F5 3 |
| ⑦ DrawDyna | ⑦ SHIFT VARS F6 F6 F2 3 |

Executing this program produces the result shown here.



⋮ ↑
↓ ⋮

## ■ Using Table & Graph Functions in a Program

Table & Graph functions in a program can generate numeric tables and perform graphing operations. The following shows various types of syntax you need to use when programming with Table & Graph functions.

• Table range setting

1 → F Start↵

5 → F End↵

1 → F pitch↵

• Numeric table generation

DispF-Tbl↵

• Graph draw operation

Connect type: DrawFTG-Con↵

Plot type: DrawFTG-Plt↵

**Example Program**

ClrGraph↵

ClrText↵

View Window 0, 6, 1, –20, 106, 10↵

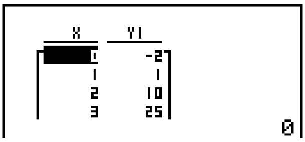
Y = Type↵

”$3X^2$ – 2” → Y1↵

| | |
|---|---|
| ① G SelOn 1↵ | ① $\boxed{\text{F6}}$ $\boxed{\text{F1}}$ $\boxed{1}$ |
| 0 → ②F Start↵ | ② $\boxed{\text{VARS}}$ $\boxed{\text{F6}}$ $\boxed{\text{F1}}$ $\boxed{1}$ |
| 6 → ③F End↵ | ③ $\boxed{\text{F1}}$ $\boxed{2}$ |
| 1 → ④F pitch↵ | ④ $\boxed{\text{F1}}$ $\boxed{3}$ |
| ⑤ DispF-Tbl◢ | ⑤ $\boxed{\text{SHIFT}}$ $\boxed{\text{VARS}}$ $\boxed{\text{F6}}$ $\boxed{\text{F6}}$ $\boxed{\text{F2}}$ $\boxed{4}$ $\boxed{1}$ |
| ⑥ DrawFTG-Con | ⑥ $\boxed{\text{SHIFT}}$ $\boxed{\text{VARS}}$ $\boxed{\text{F6}}$ $\boxed{\text{F6}}$ $\boxed{\text{F2}}$ $\boxed{4}$ $\boxed{2}$ |

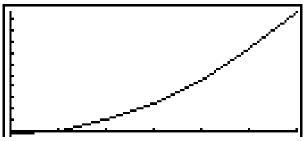
Executing this program produces the results shown here.

Numeric Table



```
    X      Y1
    0      -2
    1       1
    2      10
    3      25
                  0
```

Graph



## ■ Using Recursion Table & Graph Functions in a Program

Incorporating Recursion Table & Graph functions in a program lets you generate numeric tables and perform graphing operations. The following shows various types of syntax you need to use when programming with Recursion Table & Graph functions.

- Recursion formula input

  $a_{n+1}$ Type↵ ..... Specifies recursion type.

  ”3$a_n$ + 2” → $a_{n+1}$↵

  ”4$b_n$ + 6” → $b_{n+1}$↵

- Table range setting

  1 → R Start↵

  5 → R End↵

  1 → $a_0$↵

  2 → $b_0$↵

  1 → $a_n$ Start↵

  3 → $b_n$ Start↵

- Numeric table generation

  DispR-Tbl↵

- Graph draw operation

  Connect type: DrawR-Con↵, DrawRΣ-Con↵

  Plot type: DrawR-Plt↵, DrawRΣ-Plt↵

- Statistical convergence/divergence graph (WEB graph)

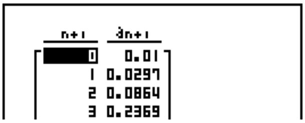
  DrawWeb $a_{n+1}$, 10↵

**Example Program**

| | |
|---|---|
| View Window 0, 1, 1, –0.2, 1, 1↵ | |
| ① $a_{n+1}$ Type↵ | ① F6 F3 6 2 |
| ”–3$\overset{②}{a}_n^2$ + 3$a_n$” → $\overset{③}{a}_{n+1}$↵ | ② F3 1 2 |
| 0 → $\overset{④}{\text{R}}$ Start↵ | ③ F3 1 3 |
| 6 → $\overset{⑤}{\text{R}}$ End↵ | ④ VARS F6 F2 2 1 |
| 0.01 → $\overset{⑥}{a}_0$↵ | ⑤ F2 2 2 |
| 0.01 → $\overset{⑦}{a}_n$ Start↵ | ⑥ F2 2 3 |
| ⑧ DispR-Tbl◢ | ⑦ F2 2 C |
| ⑨ DrawWeb $\overset{⑩}{a}_{n+1}$, 30 | ⑧ SHIFT VARS F6 F6 F2 5 1 |
| | ⑨ F2 5 2 ESC |
| | ⑩ F6 F3 1 3 |

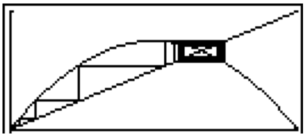
Executing this program produces the results shown here.

Numeric Table



```
 n+1    an+1
   0    0.01
   1  0.0297
   2  0.0864
   3  0.2369
```

Recursion graph



## ■ Using List Sort Functions in a Program

These functions let you sort data in lists into ascending or descending order.

• Ascending order

$\overset{①}{\text{SortA}}$ ($\overset{②}{\text{L}}$ist 1, List 2, List 3)
*Lists to be sorted (up to six can be specified)*

① F5 1 ② F4 4

• Descending order

$\overset{③}{\text{SortD}}$ (List 1, List 2, List 3)
*Lists to be sorted (up to six can be specified)*

③ F5 2

## ■ Using Solve Calculation Function in a Program

The following is the syntax for using the Solve function in a program.

$$\text{Solve(}\ f(x),\ \underbrace{n}_{\textit{Initial estimated value}},\ \underbrace{a}_{\textit{Lower limit}},\ \underbrace{b}_{\textit{Upper limit}})$$

**Example Program**

①Solve( 2X$^2$ + 7X – 9, 1, 0, 1) ① $\boxed{\text{OPTN}}$ $\boxed{\text{F4}}$ $\boxed{7}$

- In the function $f(x)$, only X can be used as a variable in the expression. Other variables (A through Z, $r$, $\theta$) are treated as constants, and the value currently assigned to that variable is applied during the calculation.
- Input of the closing parenthesis, lower limit $a$ and upper limit $b$ can be omitted.

## ■ Using Statistical Calculations and Graphs in a Program

Including statistical calculations and graphing operations in a program lets you calculate and graph statistical data.

### • To set conditions and draw a statistical graph

Following “StatGraph”, you must specify the following graph conditions:

- Graph draw/non-draw status (DrawOn/DrawOff)
- Graph Type
- $x$-axis data location (list name)
- $y$-axis data location (list name)
- Frequency data location (list name)
- Mark Type

# Solutions obtained using Solve may include errors.

# You cannot use a differential, quadratic differential, integration, Σ , maximum/minimum value or Solve calculation expressions inside of a Solve calculation term.

The graph conditions that are required depends on the graph type. See “Changing Graph Parameters” (page 6-1-2).

- The following is a typical graph condition specification for a scatter diagram or *xy*Line graph.

  S-Gph1 DrawOn, Scatter, List 1, List 2, 1, Square ↵

  In the case of an *xy* line graph, replace “Scatter” in the above specification with “*xy*Line”.

- The following is a typical graph condition specification for a normal probability plot.

  S-Gph1 DrawOn, NPPlot, List 1, Square ↵

- The following is a typical graph condition specification for a single-variable graph.

  S-Gph1 DrawOn, Hist, List 1, List 2 ↵

  The same format can be used for the following types of graphs, by simply replacing “Hist” in the above specification with the applicable graph type.

| | |
|---|---|
| Histogram: | Hist |
| Median Box: | MedBox |
| Modified Box: | Modified |
| Normal Distribution: | N-Dist |
| Broken Line: | Broken |

- The following is a typical graph condition specification for a regression graph.

  S-Gph1 DrawOn, Linear, List 1, List 2, List 3 ↵

  The same format can be used for the following types of graphs, by simply replacing “Linear” in the above specification with the applicable graph type.

| | |
|---|---|
| Linear Regression: | Linear |
| Med-Med: | Med-Med |
| Quadratic Regression: | Quad |
| Cubic Regression: | Cubic |
| Quartic Regression: | Quart |
| Logarithmic Regression: | Log |
| Exponential Regression: | Exp |
| Power Regression: | Power |

• The following is a typical graph condition specification for a sinusoidal regression graph.

S-Gph1 DrawOn, Sinusoidal, List 1, List 2 ↵

• The following is a typical graph condition specification for a logistic regression graph.

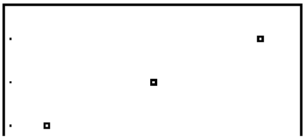
S-Gph1 DrawOn, Logistic, List 1, List 2 ↵

**Example Program**

```
ClrGraph ↵
①
S-Wind Auto ↵

{1, 2, 3} → List 1 ↵

{1, 2, 3} → List 2 ↵
②      ③       ④                  ⑤
S-Gph1 DrawOn, Scatter, List 1, List 2, 1, Square ↵
⑥
DrawStat
```

① CTRL F3 F5 1 1 ESC
② F4 1 1
③ F4 2 1
④ F4 3 1
⑤ F4 5 1
⑥ SHIFT VARS F6 F6 F2 1

Executing this program produces the scatter diagram shown here.



## ■ Performing Statistical Calculations

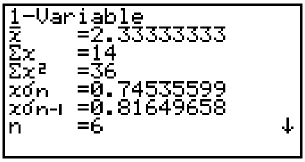
• Single-variable statistical calculation

①1-Variable $\underline{\text{List 1}}$, $\underline{\text{List 2}}$

List 2 — *Frequency data (Frequency)*

List 1 — *$x$-axis data (XList)*

① F4 6 1



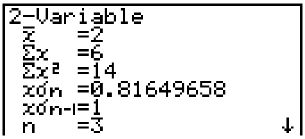
```
1-Variable
x̄      =2.33333333
Σx     =14
Σx²    =36
xσn    =0.74535599
xσn-1  =0.81649658
n      =6            ↓
```

• Paired-variable statistical calculation

①2-Variable List 1, List 2, List 3

*Frequency data (Frequency)*
*$y$-axis data (YList)*
*$x$-axis data (XList)*

① F4 6 2



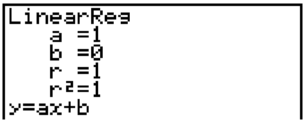
```
2-Variable
 x̄    =2
 Σx   =6
 Σx²  =14
 xσn  =0.81649658
 xσn-1=1
 n    =3           ↓
```

• Regression statistical calculation

①LinearReg List 1, List 2, List 3

*Calculation type**
*Frequency data (Frequency)*
*$y$-axis data (YList)*
*$x$-axis data (XList)*

① F4 6 3



```
LinearReg
   a =1
   b =0
   r =1
   r²=1
y=ax+b
```

* Any one of the following can be specified as the calculation type.

| | |
|---|---|
| LinearReg | linear regression |
| Med-MedLine | Med-Med calculation |
| QuadReg | quadratic regression |
| CubicReg | cubic regression |
| QuartReg | quartic regression |
| LogReg | logarithmic regression |
| ExpReg | exponential regression |
| PowerReg | power regression |

• Sinusoidal regression statistical calculation

SinReg List 1, List 2

*$y$-axis data (YList)*
*$x$-axis data (XList)*

• Logistic regression statistical calculation

LogisticReg List 1, List 2

*$y$-axis data (YList)*
*$x$-axis data (XList)*

# 8-7 Program Mode Command List

## RUN Program

| Level 1 | Level 2 | Level 3 | Command |
|---|---|---|---|
| MAT | Swap | | **Swap_** |
| | *Row | | ***Row_** |
| | *Row+ | | ***Row+_** |
| | Row+ | | **Row+_** |
| STAT | S-GPH | S-Gph1 | **S-Gph1_** |
| | | S-Gph2 | **S-Gph2_** |
| | | S-Gph3 | **S-Gph3_** |
| | DRAW | On | **DrawOn** |
| | | Off | **DrawOff** |
| | GRAPH | Scat | **Scatter** |
| | | xyLine | **xyLine** |
| | | NPPlot | **NPPlot** |
| | | Hist | **Hist** |
| | | Box | **MedBox** |
| | | ModBox | **ModifiedBox** |
| | | N-Dist | **N-Dist** |
| | | Broken | **Broken** |
| | | Linear | **Linear** |
| | | MedMed | **Med-Med** |
| | | Quad | **Quad** |
| | | Cubic | **Cubic** |
| | | Quart | **Quart** |
| | | Log | **Log** |
| | | Exp | **Exp** |
| | | Power | **Power** |
| | | Sin | **Sinusoidal** |
| | | Lgstic | **Logistic** |
| | List | | **List_** |
| | MARK | □ | **Square** |
| | | × | **Cross** |
| | | • | **Dot** |
| | CALC | 1VAR | **1-Variable_** |
| | | 2VAR | **2-Variable_** |
| | | Linear | **LinearReg_** |
| | | MedMed | **Med-MedLine_** |
| | | Quad | **QuadReg_** |
| | | Cubic | **CubicReg_** |
| | | Quart | **QuartReg_** |
| | | Log | **LogReg_** |
| | | Exp | **ExpReg_** |
| | | Power | **PowerReg_** |
| | | Sin | **SinReg_** |
| | | Lgstic | **LogisticReg_** |
| LIST | SortA | | **SortA(** |
| | SortD | | **SortD(** |
| GRPH | SelOn | | **G_SelOn_** |
| | SelOff | | **G_SelOff_** |
| | TYPE | Y= | **Y=TYPE** |
| | | r= | **r=TYPE** |
| | | Param | **ParamTYPE** |
| | | X=c | **X=cTYPE** |
| | | Y> | **Y>Type** |
| | | Y< | **Y<Type** |
| | | Y≧ | **Y≥Type** |
| | | Y≦ | **Y≤Type** |
| | GMEM | Store | **StoGMEM** |
| | | Recall | **RclGMEM** |
| DYNA | SelOn | | **D_SelOn_** |
| | SelOff | | **D_SelOff_** |
| | Var | | **D_Var_** |
| | TYPE | Y= | **Y=Type** |
| | | r= | **r=Type** |
| | | Param | **ParamType** |
| RECR | n,an.. | n | **n** |
| | | an | **an** |
| | | an+1 | **an+1** |
| | | bn | **bn** |
| | | bn+1 | **bn+1** |
| | | cn | **cn** |
| | | cn+1 | **cn+1** |
| | SelOn | | **R_SelOn_** |
| | SelOff | | **R_SelOff_** |
| | Sel a0 | | **Sel_a0** |
| | Sel a1 | | **Sel_a1** |
| | TYPE | an | **anType** |
| | | an+1 | **an+1Type** |
| | | an+2 | **an+2Type** |

| [OPTN] key | | | |
|---|---|---|---|
| Level 1 | Level 2 | Level 3 | Command |
| LIST | List | | **List_** |
| | Dim | | **Dim_** |
| | Seq | | **Seq(** |
| | Min | | **Min(** |
| | Max | | **Max(** |
| | Mean | | **Mean(** |
| | Median | | **Median(** |
| | Sum | | **Sum_** |
| | Prod | | **Prod_** |
| | Cuml | | **Cuml_** |
| | % | | **Percent_** |
| | ⊿List | | **⊿List_** |
| | Augmnt | | **Augment(** |
| | Fill | | **Fill(** |
| | L→Mat | | **List→Mat(** |
| MAT | Mat | | **Mat_** |
| | Dim | | **Dim_** |
| | Det | | **Det_** |
| | Trn | | **Trn_** |
| | Augmnt | | **Augment(** |
| | Ident | | **Identity_** |
| | Fill | | **Fill(** |
| | M→List | | **Mat→List(** |
| CPLX | Abs | | **Abs_** |
| | Arg | | **Arg_** |
| | Conjg | | **Conjg_** |
| | ReP | | **ReP_** |
| | ImP | | **ImP_** |
| | ►re^θi | | **►re^θi** |
| | ►a+bi | | **►a+bi** |
| CALC | d/dx | | **d/dx(** |
| | $d^2/dx^2$ | | **$d^2/dx^2$(** |
| | ∫dx | | **∫(** |
| | Σ | | **Σ(** |
| | FMin | | **FMin(** |
| | FMax | | **FMax(** |
| | Solve | | **Solve(** |
| NUM | Abs | | **Abs_** |
| | Int | | **Int_** |
| | Frac | | **Frac_** |
| | Rnd | | **Rnd** |
| | Intg | | **Intg_** |
| | E-SYM | m | **m** |
| | | µ | **µ** |
| | | n | **n** |
| | | p | **p** |
| | | f | **f** |
| | | k | **k** |
| | | M | **M** |
| | | G | **G** |
| | | T | **T** |
| | | P | **P** |
| | | E | **E** |
| PROB | x! | | **!** |
| | nPr | | **P** |
| | nCr | | **C** |
| | Ran# | | **Ran#_** |
| | P( | | **P(** |
| | Q( | | **Q(** |
| | R( | | **R(** |
| | t( | | **t(** |
| HYP | sinh | | **sinh_** |
| | cosh | | **cosh_** |
| | tanh | | **tanh_** |
| | $\sinh^{-1}$ | | **$\sinh^{-1}$_** |
| | $\cosh^{-1}$ | | **$\cosh^{-1}$_** |
| | $\tanh^{-1}$ | | **$\tanh^{-1}$_** |
| ANGL | ° | | ° |
| | r | | **r** |
| | g | | **g** |
| | ° ' " | | ° ' " |
| | ►DMS | | **►DMS** |
| | Pol( | | **Pol(** |
| | Rec( | | **Rec(** |
| STAT | $\hat{x}$ | | **$\hat{x}$** |
| | $\hat{y}$ | | **$\hat{y}$** |
| FMEM | fn | | **fn** |
| ZOOM | Factor | | **Factor_** |
| | Auto | | **ZoomAuto** |
| SKTCH | Cls | | **Cls** |
| | PLOT | On | **PlotOn_** |
| | | Off | **PlotOff_** |
| | | Change | **PlotChg_** |
| | | Plot | **Plot_** |
| | LINE | F-Line | **F-Line_** |
| | | Line | **Line** |
| | GRAPH | Y= | **Graph_Y=** |
| | | ∫dx | **Graph_∫** |
| | Text | | **Text_** |
| | PIXEL | On | **PxlOn_** |
| | | Off | **PxlOff_** |
| | | Change | **PxlChg_** |
| | | Test | **PxlTest(** |
| | Tangnt | | **Tangent_** |
| | Normal | | **Normal_** |
| | Invrse | | **Inverse_** |
| | Circle | | **Circle_** |
| | Vert | | **Vertical_** |
| | Horz | | **Horizontal_** |
| PICT | Store | | **StoPict_** |
| | Recall | | **RclPict_** |
| SYBL | ' | | **'** |
| | " | | **"** |
| | ~ | | **~** |
| | * | | ***** |
| | # | | **#** |
| ° ' " | | | □ |

**[VARS] key**

| Level 1 | Level 2 | Level 3 | Command |
|---|---|---|---|
| V-WIN | Xmin | | **Xmin** |
| | Xmax | | **Xmax** |
| | Xscale | | **Xscl** |
| | Xdot | | **Xdot** |
| | Ymin | | **Ymin** |
| | Ymax | | **Ymax** |
| | Yscale | | **Yscl** |
| | Tθmin | | **Tθmin** |
| | Tθmax | | **Tθmax** |
| | Tθptch | | **Tθptch** |
| | R-Xmin | | **RightXmin** |
| | R-Xmax | | **RightXmax** |
| | R-Xscl | | **RightXscl** |
| | R-Xdot | | **RightXdot** |
| | R-Ymin | | **RightYmin** |
| | R-Ymax | | **RightYmax** |
| | R-Yscl | | **RightYscl** |
| | R-Tmin | | **RightT θmin** |
| | R-Tmax | | **RightT θmax** |
| | R-Tpch | | **RightT θptch** |
| FACT | Xfact | | **Xfct** |
| | Yfact | | **Yfct** |
| STAT | n | | **n** |
| | X | $\bar{x}$ | $\bar{x}$ |
| | | $\Sigma x$ | **$\Sigma x$** |
| | | $\Sigma x^2$ | **$\Sigma x^2$** |
| | | xσn | **xσn** |
| | | xσn–1 | **xσn–1** |
| | | minX | **minX** |
| | | maxX | **maxX** |
| | Y | $\bar{y}$ | $\bar{y}$ |
| | | $\Sigma y$ | **$\Sigma y$** |
| | | $\Sigma y^2$ | **$\Sigma y^2$** |
| | | $\Sigma xy$ | **$\Sigma xy$** |
| | | yσn | **yσn** |
| | | yσn–1 | **yσn–1** |
| | | minY | **minY** |
| | | maxY | **maxY** |
| | GRAPH | a | **a** |
| | | b | **b** |
| | | c | **c** |
| | | d | **d** |
| | | e | **e** |
| | | r | **r** |
| | | $r^2$ | **$r^2$** |
| | | Q1 | **Q1** |
| | | Med | **Med** |
| | | Q3 | **Q3** |
| | | Mod | **Mod** |
| | | H-Strt | **H_Start** |
| | | H-ptch | **H_pitch** |
| | PTS | x1 | **x1** |
| | | y1 | **y1** |
| | | x2 | **x2** |
| | | y2 | **y2** |
| | | x3 | **x3** |
| | | y3 | **y3** |
| GRPH | Yn | | **Y** |
| | rn | | **r** |
| | Xtn | | **Xt** |
| | Ytn | | **Yt** |
| | Xn | | **X** |
| DYNA | Start | | **D_Start** |
| | End | | **D_End** |
| | Pitch | | **D_pitch** |
| TABL | Start | | **F_Start** |
| | End | | **F_End** |
| | Pitch | | **F_pitch** |
| | Result | | **F_Result** |
| RECR | FORM | an | **an** |
| | | an+1 | **an+1** |
| | | an+2 | **an+2** |
| | | bn | **bn** |
| | | bn+1 | **bn+1** |
| | | bn+2 | **bn+2** |
| | | cn | **cn** |
| | | cn+1 | **cn+1** |
| | | cn+2 | **cn+2** |
| | RANGE | R-Strt | **R_Start** |
| | | R-End | **R_End** |
| | | a0 | **a0** |
| | | a1 | **a1** |
| | | a2 | **a2** |
| | | b0 | **b0** |
| | | b1 | **b1** |
| | | b2 | **b2** |
| | | c0 | **c0** |
| | | c1 | **c1** |
| | | c2 | **c2** |
| | | anStrt | **anStart** |
| | | bnstrt | **bnStart** |
| | | cnStrt | **cnStart** |
| | Result | | **R_Result** |
| EQUA | S-Rslt | | **Sim_Result** |
| | S-Coef | | **Sim_Coef** |
| | P-Rslt | | **Ply_Result** |
| | P-Coef | | **Ply_Coef** |

**[SHIFT][VARS](PRGM) key**

| Level 1 | Level 2 | Level 3 | Command |
|---|---|---|---|
| Prog | | | **Prog_** |
| JUMP | Lbl | | **Lbl_** |
| | Goto | | **Goto_** |
| | Isz | | **Isz_** |
| | Dsz | | **Dsz_** |
| ? | | | **?** |
| ◢ | | | **◢** |
| I/O | Locate | | **Locate_** |
| | Getkey | | **Getkey** |
| | Send | | **Send(** |
| | Receiv | | **Receive(** |
| IF | If | | **If_** |
| | Then | | **Then_** |
| | Else | | **Else_** |
| | IfEnd | | **IfEnd** |
| FOR | For | | **For_** |
| | To | | **_To_** |
| | Step | | **_Step_** |
| | Next | | **Next** |
| WHLE | While | | **While_** |
| | WhlEnd | | **WhileEnd** |
| | Do | | **Do** |
| | LpWhle | | **LpWhile_** |
| CTRL | Prog | | **Prog_** |
| | Return | | **Return** |
| | Break | | **Break** |
| | Stop | | **Stop** |
| LOGIC | = ≠ < | = | **=** |
| | | ≠ | **≠** |
| | | > | **>** |
| | | < | **<** |
| | | ≧ | **≧** |
| | | ≦ | **≦** |
| | And | | **_And_** |
| | Or | | **_Or_** |
| | Not | | **Not_** |
| CLR | Text | | **ClrText** |
| | Graph | | **ClrGraph** |
| | List | | **ClrList_** |
| | Matrix | | **ClrMat_** |
| DISP | Stat | | **DrawStat** |
| | Graph | | **DrawGraph** |
| | Dyna | | **DrawDyna** |
| | F-TBL | Table | **DispF-Tbl** |
| | | G-Con | **DrawFTG-Con** |
| | | G-Plot | **DrawFTG-Plt** |
| | R-TBL | Table | **DispR-Tbl** |
| | | Web | **DrawWeb_** |
| | | R-Con | **DrawR-Con** |
| | | RΣ-Con | **DrawRΣ-Con** |
| | | R-Plot | **DrawR-Plt** |
| | | RΣ-Plt | **DrawRΣ-Plt** |
| : | | | **:** |

**[CTRL][F3](SET UP) key**

| Level 1 | Level 2 | Level 3 | Command |
|---|---|---|---|
| ANGL | Deg | | **Deg** |
| | Rad | | **Rad** |
| | Gra | | **Gra** |
| DISP | Fix | | **Fix_** |
| | Sci | | **Sci_** |
| | Norm | | **Norm** |
| | EngOn | | **EngOn** |
| | EngOff | | **EngOff** |
| CPLX | Real | | **Real** |
| | a+bi | | **a+bi** |
| | re^θi | | **re^θi** |
| GRPH | G-FUNC | On | **FuncOn** |
| | | Off | **FuncOff** |
| | D-TYPE | G-Con | **G-Connect** |
| | | G-Plot | **G-Plot** |
| | BG | None | **BG-None** |
| | | Pict | **BG-Pict_** |
| | SIMUL | On | **SimulOn** |
| | | Off | **SimulOff** |
| | COORD | On | **CoordOn** |
| | | Off | **CoordOff** |
| | GRID | On | **GridOn** |
| | | Off | **GridOff** |
| | AXES | On | **AxesOn** |
| | | Off | **AxesOff** |
| | LABEL | On | **LabelOn** |
| | | Off | **LabelOff** |
| STAT | S-WIN | Auto | **S-WindAuto** |
| | | Manual | **S-WindMan** |
| | File | | **File_** |
| | RESID | None | **Resid-None** |
| | | List | **Resid-List_** |
| DERIV | On | | **DerivOn** |
| | Off | | **DerivOff** |
| T-VAR | Range | | **VarRange** |
| | List | | **VarList_** |
| Σ•DSP | On | | **ΣdispOn** |
| | Off | | **ΣdispOff** |

| [SHIFT][OPTN](V-Window)key | | | |
|---|---|---|---|
| **Level 1** | **Level 2** | **Level 3** | **Command** |
| V-Win | | | **ViewWindow_** |
| Sto | | | **StoV-Win_** |
| Rcl | | | **RclV-Win_** |

## BASE Program

| **Level 1** | **Level 2** | **Level 3** | **Command** |
|---|---|---|---|
| d~o | d | | **d** |
| | h | | **h** |
| | b | | **b** |
| | o | | **o** |
| LOG | Neg | | **Neg_** |
| | Not | | **Not_** |
| | and | | **and** |
| | or | | **or** |
| | xor | | **xor** |
| | xnor | | **xnor** |
| DISP | ▸Dec | | **▸Dec** |
| | ▸Hex | | **▸Hex** |
| | ▸Bin | | **▸Bin** |
| | ▸Oct | | **▸Oct** |

| [SHIFT][VARS](PRGM) key | | | |
|---|---|---|---|
| **Level 1** | **Level 2** | **Level 3** | **Command** |
| Prog | | | **Prog_** |
| JUMP | Lbl | | **Lbl_** |
| | Goto | | **Goto_** |
| | Isz | | **Isz_** |
| | Dsz | | **Dsz_** |
| ? | | | **?** |
| ◢ | | | **◢** |
| = ≠ < | = | | **=** |
| | ≠ | | **≠** |
| | > | | **>** |
| | < | | **<** |
| | ≧ | | **≧** |
| | ≦ | | **≦** |
| : | | | **:** |

| [CTRL][F3](SETUP) key | | | |
|---|---|---|---|
| **Level 1** | **Level 2** | **Level 3** | **Command** |
| Dec | | | **Dec** |
| Hex | | | **Hex** |
| Bin | | | **Bin** |
| Oct | | | **Oct** |

# 8-8 Program Library

- Be sure to check how many bytes of unused memory are remaining before attempting to perform any programming.

## Program Name Prime Factorization

**Description**

This program continually divides a natural number by factors until all its prime factors are produced.

**Purpose**

This program accepts input of natural number A, and divides it by B (2, 3, 5, 7....) to find the prime factors of A.

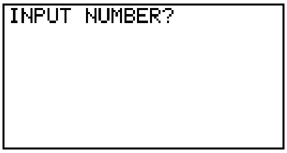
- If a division operation does not produce a remainder, the result of the operation is assigned to A.
- The above procedure is repeated until B > A.

● ● ● ● ●

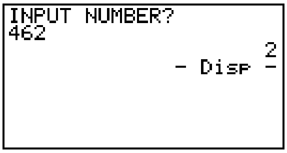
**Example**

440730 = 2 × 3 × 3 × 5 × 59 × 83

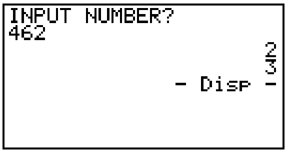
```
ClrText↵
"INPUT NUMBER"?→A↵
2→B↵
Do↵
While Frac (A/B)=0↵
B◢
A/B→A↵
WhileEnd↵
If B=2↵
Then 3→B↵
Else B+2→B↵
IfEnd↵
LpWhile B≤A↵
"END"
```


```
INPUT NUMBER?
```

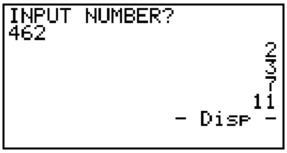
4 6 2 EXE


```
INPUT NUMBER?
462
                  2
           - Disp -
```

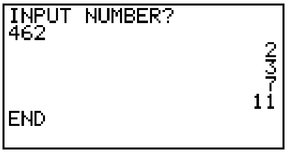
EXE


```
INPUT NUMBER?
462
                  2
                  3
           - Disp -
```

EXE EXE


```
INPUT NUMBER?
462
                  2
                  3
                  7
                 11
           - Disp -
```

EXE


```
INPUT NUMBER?
462
                  2
                  3
                  7
                 11
END
```

## Program Name Arithmetic-Geometric Sequence Differentiation

### Description

After inputting sequence terms 1, 2, and 3, this program determines whether it is an arithmetic sequence or geometric sequence based on the differences and ratios of the terms.

### Purpose

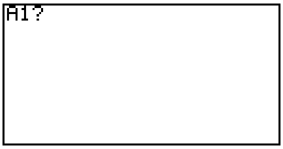
This program determines whether a specific sequence is an arithmetic sequence or geometric sequence.

● ● ● ● ●
**Example 1** 5, 10, 15, ... Arithmetic sequence

● ● ● ● ●
**Example 2** 5, 10, 20, ... Geometric sequence

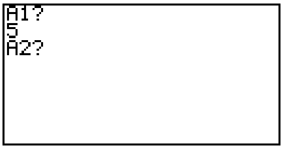
```
ClrText↵
"A1"?→A↵
"A2"?→B↵
"A3"?→C↵
B-A→D↵
C-B→E↵
If D=E↵
Then ClrText↵
"AN = A1 + (N-1)D"↵
" "↵
"A1 ="↵
"D  ="↵
Locate 6,3,A↵
Locate 6,4,D◢
IfEnd↵
B/A→F↵
C/B→G↵
If F=G↵
Then ClrText↵
"AN = A1×r^(N-1)"↵
" "↵
"A1 ="↵
"r  ="↵
Locate 6,3,A↵
Locate 6,4,F↵
IfEnd↵
"END"
```

**Example 1**



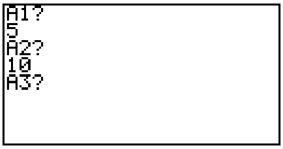
```
A1?
```

5 EXE



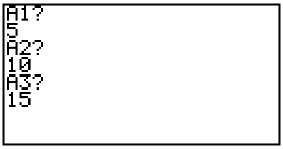
```
A1?
5
A2?
```

1 0 EXE



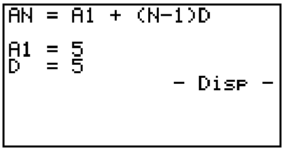
```
A1?
5
A2?
10
A3?
```

1 5



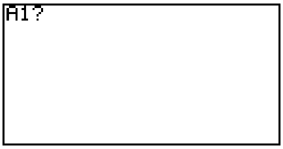
```
A1?
5
A2?
10
A3?
15
```

EXE



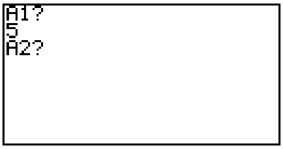
```
AN = A1 + (N-1)D

A1 = 5
D  = 5
             - Disp -
```

**Example 2**



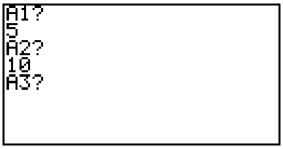
```
A1?
```

5 EXE



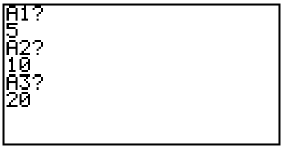
```
A1?
5
A2?
```

1 0 EXE



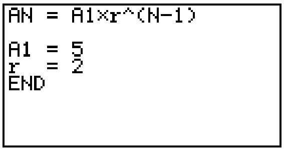
```
A1?
5
A2?
10
A3?
```

2 0



```
A1?
5
A2?
10
A3?
20
```

EXE



```
AN = A1×r^(N-1)

A1 = 5
r  = 2
END
```

## Program Name Ellipse

### Description

This program displays a number table of the following values based on input of the foci of an ellipse, the sum of the distance between the loci and foci, and the pitch (step size) of X.

Y1: Coordinate values of upper half of ellipse

Y2: Coordinate values of lower half of ellipse

Y3: Distances between right focus and loci

Y4: Distances between left focus and loci

Y5: Sum of Y3 and Y4

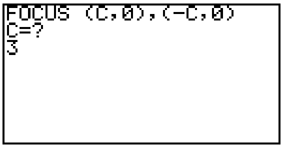
Next, the program plots the foci and values in Y1 and Y2.

### Purpose

This program shows that the sums of the distances between the loci and two foci of an ellipse are equal.

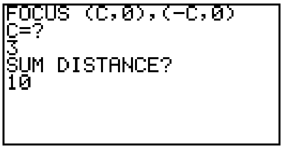
```
Do↵
ClrText↵
"FOCUS (C,0),(-C,0)"↵
"C="?→C↵
"SUM DISTANCE"?→D↵
LpWhile 2Abs C≥D Or D<0↵
D/2→A↵
√(A²-C²)→B↵
Y=Type↵
"B√(1-X²/A²)"→Y1↵
"-Y1"→Y2↵
"√((X-C)²+Y1²)"→Y3↵
"√((X+C)²+Y1²)"→Y4↵
"Y3+Y4"→Y5↵
For 1→E To 20↵
If E≤5↵
Then G SelOn E↵
Else G SelOff E↵
IfEnd↵
Next↵
-Int A→F Start↵
Int A→F End↵
"F pitch"?→F pitch↵
DispF-Tbl◢
ClrGraph↵
1.2A→Xmax↵
-1.2A→Xmin↵
1.2B→Ymax↵
-1.2B→Ymin↵
G SelOff 3↵
G SelOff 4↵
G SelOff 5↵
DispF-Tbl↵
DrawFTG-Plt↵
PlotOn C,0↵
PlotOn -C,0◢
"END"
```

3



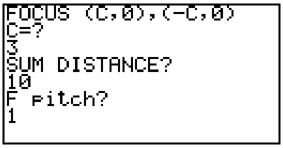
```
FOCUS (C,0),(-C,0)
C=?
3
```

EXE 1 0



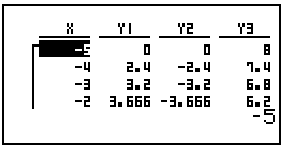
```
FOCUS (C,0),(-C,0)
C=?
3
SUM DISTANCE?
10
```

EXE 1



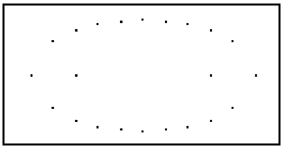
```
FOCUS (C,0),(-C,0)
C=?
3
SUM DISTANCE?
10
F pitch?
1
```

EXE



```
        X      Y1      Y2      Y3
       -5       0       0       8
       -4     2.4    -2.4     7.4
       -3     3.2    -3.2     6.8
       -2   3.666  -3.666     6.2
                               -5
```

EXE CTRL 0



## Program Name Rotation

### Description

This program draws an angle at the coordinate defined by an input vertex, and then rotates it to a specified angle around the vertex.

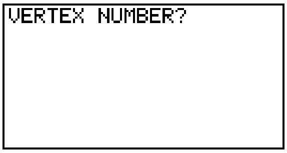
### Purpose

This program demonstrates coordinate transformation using a matrix.

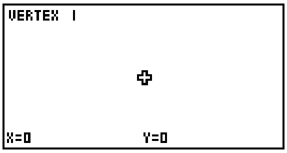
### Important!

Deg must be set as the angle unit for this program.

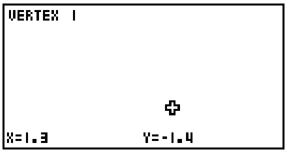
```
Do↵
ClrText↵
"VERTEX NUMBER"?→A↵
LpWhile A≤0 Or Frac A≠0↵
{2,A}→Dim Mat A↵
ClrGraph↵
For 1→B To A↵
Text 1,1,"VERTEX"↵
Text 1,30,B↵
If B=1↵
Then Plot ↵
PlotOn X,Y↵
X→Mat A[1,B]↵
Y→Mat A[2,B]↵
Else Plot C,D↵
F-Line C,D,X,Y↵
X→Mat A[1,B]↵
Y→Mat A[2,B]↵
IfEnd↵
Mat A[1,B]→C↵
Mat A[2,B]→D↵
Next↵
Mat A[1,1]→E↵
Mat A[2,1]→F↵
F-Line C,D,E,F↵
Text 1,1,"--AXIS--"↵
Plot ↵
PlotOn X,Y↵
X→C↵
Y→D↵
A→Dim List 1↵
A→Dim List 2↵
Fill(C,List 1)↵
Fill(D,List 2)↵
List→Mat(List 1,List 2)↵
Trn Mat Ans→Mat C↵
Mat A-Mat C→Mat A↵
ClrText↵
"ANGLE"?→E↵
[[cos E,-sin E][sin E,cos E]]→Mat B↵
Mat B×Mat A→Mat D↵
Mat D+Mat C→Mat D↵
If A=1↵
Then PlotOn Mat D[1,1],Mat D[2,1]↵
Else For 1→B To A-1↵
Mat D[1,B]→F↵
Mat D[2,B]→G↵
Mat D[1,B+1]→H↵
Mat D[2,B+1]→I↵
F-Line F,G,H,I↵
Next↵
If A>2↵
Then Mat D[1,1]→F↵
Mat D[2,1]→G↵
F-Line H,I,F,G↵
IfEnd↵
IfEnd↵
Text 1,1,"--END--"
```


VERTEX NUMBER?

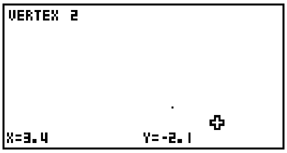
3 EXE


VERTEX 1
X=0 Y=0

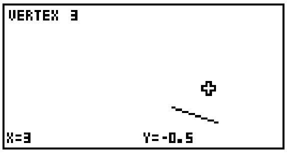
▲ ▼ ◀ ▶


VERTEX 1
X=1.3 Y=-1.4

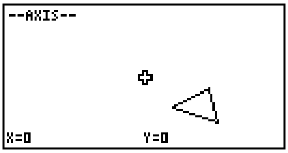
EXE EXE ▲ ▼ ◀ ▶


VERTEX 2
X=3.4 Y=-2.1

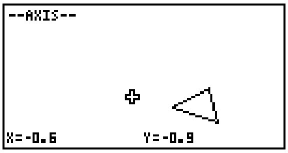
EXE EXE ▲ ▼ ◀ ▶


VERTEX 3
X=3 Y=-0.5

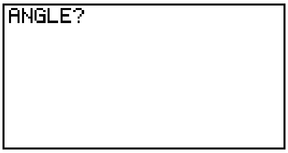
EXE EXE


--AXIS--
X=0 Y=0

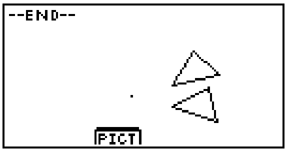
▲ ▼ ◀ ▶


--AXIS--
X=-0.6 Y=-0.9

EXE EXE


ANGLE?

3 0 EXE


--END--

## Program Name Interior Angles and Surface Area of a Triangle

### Description

This program calculates the interior angles and surface area of a triangle defined by input coordinates for angles A, B, and C.

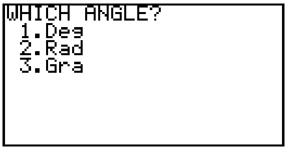
### Purpose

This program calculates the interior angles and surface area of a triangle defined by coordinates for angles A, B, and C.

### Important!

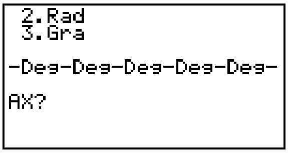
Inputting the same coordinates for any two angles (A, B, C) causes an error.

```
C rText↵
"WHICH ANGLE?"↵
" 1.Deg"↵
" 2.Rad"↵
" 3.Gra"↵
Do↵
Getkey↵
LpWhile ((Ans=72) Or (Ans=62) Or (Ans=52))=0↵
If Ans=72↵
Then 1→θ↵
Deg↵
" "↵
"-Deg Deg-Deg-Deg-Deg-"↵
IfEnd↵
If Ans=62↵
Then 2→θ↵
Rad↵
" "↵
"-Rad-Rad-Rad-Rad-Rad-"↵
IfEnd↵
If Ans=52↵
Then 3→θ↵
Gra↵
" "↵
"-Gra-Gra-Gra-Gra-Gra-"↵
IfEnd↵
"AX"?→A↵
"AY"?→B↵
"BX"?→C↵
"BY"?→D↵
"CX"?→E↵
"CY"?→F↵
A-C→G↵
B-D→H↵
C-E→I↵
D-F→J↵
E-A→K↵
F B→L↵
-GI-HJ→M↵
-IK-JL→N↵
-KG-LH→O↵
√(G²+H²)→P↵
√(I²+J²)→Q↵
√(K²+L²)→R↵
M/PQ→S↵
N/QR→T↵
O/PR→U↵
cos⁻¹ S→V↵
cos⁻¹ T→W↵
cos⁻¹ U→X↵
PQ√(1-S²)→Y↵
ClrText↵
" <ABC ="↵
Locate 9,1,V↵
" <ACB ="↵
Locate 9,2,W↵
" <BAC ="↵
Locate 9,3,X↵
If θ=1↵
Then "              (Deg)"↵
IfEnd↵
If θ=2↵
Then "              (Rad)"↵
IfEnd↵
If θ=3↵
Then "              (Gra)"↵
IfEnd↵
" AREA ="↵
Locate 9,5,Y/2↵
" "↵
"END"
```



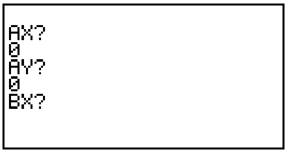
```
WHICH ANGLE?
 1.Deg
 2.Rad
 3.Gra
```

1



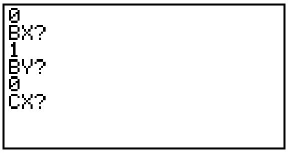
```
 2.Rad
 3.Gra

-Deg-Deg-Deg-Deg-Deg-

AX?
```

0 EXE 0 EXE



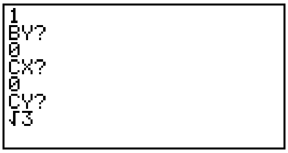
```

AX?
0
AY?
0
BX?
```

1 EXE 0 EXE



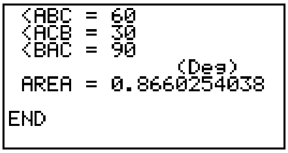
```
0
BX?
1
BY?
0
CX?
```

0 EXE √ 3



```
1
BY?
0
CX?
0
CY?
√3
```

EXE



```
 <ABC = 60
 <ACB = 30
 <BAC = 90
             (Deg)
 AREA = 0.8660254038

END
```