

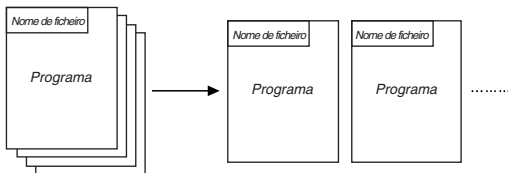


Programação

1. Antes de programar
2. Exemplos de programação
3. Como rectificar um programa
4. Como calcular o número de bytes usado por um programa
5. Função de segredo
6. Como procurar um ficheiro
7. Como editar o conteúdo de um programa
8. Como apagar um programa
9. Comandos úteis de programação
10. Referência dos comandos
11. Visualização de texto
12. Como utilizar as funções de calculadora em programas

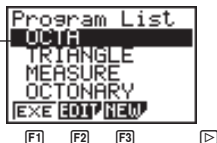
1. Antes de programar

A função de programação ajuda a tornar rápidos e fáceis os cálculos complexos e muito frequentes. Os comandos e os cálculos são executados sequencialmente, tal como as instruções múltiplas dos cálculos manuais. Os múltiplos programas podem ser guardados em ficheiros com nomes para se voltar a chamar e editar facilmente.



Selecione o ícone **PRGM** no menu principal e entre no modo PRGM. Assim que o fizer, aparecerá no visor uma lista de programas.

Área de memória seleccionada
(Utilize ▲ e ▼ para mover)



- F1** (EXE) Executar o programa
- F2** (EDIT) Edição do programa
- F3** (NEW) Novo programa



- F1** (DEL) Apagar programa específico
- F2** (DEL•A) Apagar tudo
- F3** (SRC) Buscar nome de ficheiro

Prima para voltar ao menu anterior.

- Se não houver programas guardados na memória quando entrar no modo PRGM, aparecerá no visor o aviso **"No Programs"** (Nenhum programa) e só aparecerá no menu de funções, o item NEW (NOVO) (**F3**).

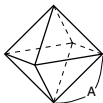


P.138
P.138
P.135

2. Exemplos de programação

Exemplo 1 Como calcular a área da superfície e o volume de três octaedros regulares com as dimensões abaixo indicadas.

Guarde a fórmula do cálculo no ficheiro de nome OCTA.



Comprimento do lado (A)	Área da superfície (S)	Volume (V)
7 cm	cm ²	cm ³
10 cm	cm ²	cm ³
15 cm	cm ²	cm ³

Seguem-se as fórmulas utilizados para o cálculo da área da superfície S e o volume V de um octaedro regular, dada a medida de um lado.

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

Quando registar a fórmula nova, registre primeiro o nome do ficheiro e depois introduza o programa actual.

• Como registar um nome de ficheiro

Exemplo Como registar o nome de ficheiro OCTA

- Note que um nome de ficheiro pode ter até 8 caracteres.

1. Enquanto visualiza a lista dos programas, prima **F3** (NEW).

F3 (NEW)

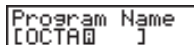


F3 (n0) Registo de senha

F4 (SYBL) Menu de símbolos

2. Introduza o nome do ficheiro.

O C T A



- O cursor muda de formato para indicar a introdução de carácter alfabético.
- Seguem-se os caracteres que poderá utilizar num nome de ficheiro:
De A a Z, espaços, [,] , { , } , " , ' , ~ , 0 a 9, . , + , - , x , +

- Ao premir **F4** (SYBL), visualizará um menu de símbolos de que poderá introduzir.

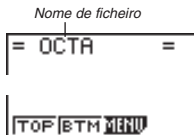
F4 (SYBL)



- Poderá apagar um carácter quando introduzir um nome de ficheiro, movendo o cursor para o carácter que quer apagar e premindo **DEL**.

3. Prima **EXE** para registar o nome do ficheiro e mudar para o ecrã de introdução de programas.

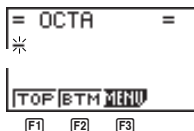
EXE



- O registo de um nome de ficheiro usa 17 bytes de memória.
- O ecrã de introdução de senha permanecerá no visor se tiver premido **EXE** sem introduzir um nome de ficheiro.
- Para sair do ecrã de introdução de nome de ficheiro e voltar à lista de programas, sem registar um nome de ficheiro, prima **QUIT**.

•Como introduzir um programa

Utilize o ecrã de introdução de programas para introduzir o conteúdo dum programa.



F1 (TOP) Parte superior do programa

F2 (BTM) Parte inferior do programa

F3 (MENU) Menu de modos

- Ao premir **▷**, visualizará um menu de símbolos a ser introduzidos num programa.

▷



▷



Prima **▷** para voltar ao menu anterior.



P.136

P.136



•Como mudar de modos num programa

- Ao premir **F3** (MENU), enquanto visualiza o ecrã de introdução de programas, fará aparecer um menu de mudança de modo. Poderá utilizar este menu para introduzir a mudança de modos nos seus programas. Para detalhes sobre cada um destes modos, veja “Utilização do menu principal”, assim como as secções deste manual que descrevem o que pode fazer em cada modo.

F3 (MENU)



- Ao premir **SHIFT** **SETUP**, visualizará um menu de comandos que poderá usar para alterar as programações do ecrã de preparação, num programa. Para detalhes sobre cada um destes comandos, veja “Como alterar a configuração de um modo”.

SHIFT **SETUP**



Os conteúdos reais dos programas são idênticos aos das calculadoras manuais. A seguir, mostra-se como seriam calculados a área da superfície e o volume de um octaedro regular, utilizando o cálculo manual.

Área da superfície S **2** **X** **SHIFT** **✓** **3** **X** <valor de A> **x²** **EXE**

Volume V **SHIFT** **✓** **2** **⇄** **3** **X** <valor de A> **∧** **3** **EXE**

Poderá também efectuar esse cálculo por meio da atribuição da medida dum lado à variável A.

Área da superfície S

..... <valor de A> **⇨** **ALPHA** **A** **EXE**

Área da superfície S **2** **X** **SHIFT** **✓** **3** **X** **ALPHA** **A** **X²** **EXE**

Volume V **SHIFT** **✓** **2** **÷** **3** **X** **ALPHA** **A** **^** **3** **EXE**

No entanto, se introduzir apenas os cálculos manuais acima indicados, a calculadora executá-los-á, de princípio ao fim, sem parar. Os comandos seguintes permitir-lhe-ão interromper um cálculo para a introdução de valores, e a visualização dos resultados intermédios.

?: Este comando interrompe a execução do programa e faz visualizar um ponto de interrogação como indicação para a introdução dum valor a atribuir a uma variável. Os passos deste comando são: ? → <nome da variável>.

▲: Este comando interrompe a execução do programa e faz visualizar o resultado obtido no último cálculo ou texto. É semelhante ao accionamento de **EXE** num cálculo manual.

- Para mais detalhes sobre como utilizar estes e outros comandos, veja "Comandos úteis de programação".

Seguem-se exemplos de como utilizar os comandos **?** e **▲**.

SHIFT **PRGM** **▶** **F1** **(?)** **←** **ALPHA** **A** **▶** **F3** **(:)**
2 **X** **SHIFT** **✓** **3** **X** **ALPHA** **A** **X²**
▶ **▶** **F2** **(▲)**

= OCTA =
 ?→A:2×√3×A²,
 _
 ? ▲ CLR/DISP
F1 **F2**

SHIFT **✓** **2** **÷** **3** **X** **ALPHA** **A** **^** **3**

= OCTA =
 ?→A:2×√3×A²,
 √2+3×A³_

QUIT **QUIT**

Program List
 OCTA

•Como executar um programa

1. Enquanto visualiza o programa, utilize **▲** e **▼** para destacar a especificação do programa que quer executar.
2. Prima **F1** (EXE) ou **EXE** para executar o programa.

Tentemos executar o programa que introduzimos acima.

Comprimento do lado (A)	Área da superfície (S)	Volume (V)
7 cm	169,7409791 cm ²	161,6917506 cm ³
10 cm	346,4101615 cm ²	471,4045208 cm ³
15 cm	779,4228634 cm ²	1590,990258 cm ³

```
Program List
UCTH
```

```
EXE EDIT NEW
```

F1

F1 (EXE) ou EXE

```
?
```

7 EXE

(Valor de A)

```
?
?
169.7409791
- DISP -
```

Resultado intermediário produzido por ▲

EXE

```
?
?
169.7409791
161.6917506
```

EXE

```
?
?
169.7409791
161.6917506
?
```

1 0 EXE

```
161.6917506
?
10
346.4101615
- DISP -
```

EXE

```
161.6917506
?
10
346.4101615
471.4045208
```

⋮

⋮



P.149

- Ao premir **EXE**, enquanto visualiza o resultado final do programa, executará de novo o programa.
- Poderá ainda executar um programa, enquanto no **modo RUN**, por meio da introdução de:
Prog "<nome do ficheiro>" **EXE**.
- Ocorre um erro (Go ERROR) se o programa especificado por Prog "<nome do ficheiro >" não puder ser encontrado.

3. Como rectificar um programa

Chama-se “bug” a um problema que impede a execução correcta dum programa, e ao processo de eliminar tais problemas chama-se “depurar” (debugging). Os sintomas seguintes são indicação de que o programa contém bugs e que é necessário depurá-lo.



- Aparecimento de avisos de erro, durante a execução do programa.
- Os resultados não condizem com os esperados.



• Como eliminar os bugs que causam avisos de erro.

Aparecerá um aviso de erro semelhante ao abaixo ilustrado, sempre que ocorrer algo não autorizado, durante a execução do programa.



Ma ERROR

Quando aparecer tal aviso, prima  e  para visualizar o local onde se originou o erro, assim como o cursor. Consulte o “Quadro de avisos de erro” para se informar dos passos a executar para corrigir a situação.

- Repare que premir  ou  não visualizará o local do erro se o programa estiver protegido com uma senha.

• Como eliminar os bugs que dão origem a resultados errados



Se o seu programa produzir resultados que não são os que seria de esperar, verifique o conteúdo do programa e faça as necessárias alterações. Veja “Como editar o conteúdo de um programa”, para mais detalhes sobre como alterar o conteúdo do programa.

4. Como calcular o número de bytes usado por um programa

Esta calculadora tem 20.000 bytes de memória. Um byte é a unidade memória que pode ser usada na memorização do dados.

Há 2 tipos de comandos: comandos de 1 byte e comandos de 2 bytes.

- Exemplos de comandos de 1 byte: sin, cos, tan, log, (,), A, B, C, 1, 2, etc.
- Exemplos de comandos de 2 bytes: Lbl 1, Goto 2, etc.

Enquanto o cursor estiver localizado no interior dum programa, cada accionamento de  e  fá-lo-á deslocar-se um byte.

- Poderá verificar quanta memória foi usada e quanta sobra, em qualquer altura, por meio da selecção do ícone **MEM** no menu principal e entrada no modo MEM. Veja “Situação da memória (MEM)” para mais detalhes.



P.200



P.133



P.135



P.37

5. Função de Segredo

Ao fazer um programa, você pode protegê-lo com uma senha que limita o acesso ao conteúdo do programa apenas para os que sabem a senha. Programas protegidos com senha não podem ser executados sem a introdução da senha.

•Para registar uma senha

Exemplo Para criar um ficheiro de programa com o nome AREA e protegê-lo com a senha CASIO

1. Com a lista de programas no ecrã, prima **F3** (NEW) e introduza o nome do novo ficheiro de programa.

F3 (NEW)
A **R** **E** **A**



F3

2. Prima **F3** (**π0**) e depois introduza a senha.

F3 (**π0**)
C **A** **S** **I** **O**




 P.127

- O procedimento de introdução da senha é idêntico ao usado para a introdução do nome de ficheiro.
3. Prima **EXE** para registar o nome do ficheiro e senha. Agora pode introduzir o conteúdo do ficheiro do programa.
 - O registo de uma senha utiliza 16 bytes de memória.
 - Premir **EXE** sem introduzir uma senha regista somente o nome do ficheiro, sem uma senha.
 4. Depois de fazer o programa, prima **QUIT** para sair do ficheiro do programa e retornar à lista de programas. Os ficheiros protegidos com uma senha são indicados por um asterisco à direita do nome do ficheiro.

QUIT



•Para chamar um programa

Exemplo Para chamar o ficheiro chamado AREA que está protegido com a senha CASIO

1. Na lista de programas, utilize \blacktriangle e \blacktriangledown para mover o realce para o nome do programa que deseja chamar.

2. Prima F_2 (EDIT).

F_2 (EDIT)

```

Program Name
[AREA  ]
Password?
[ ]

```

3. Introduza a senha e prima EXE para chamar o programa.

- A mensagem "Mismatch" aparecerá se introduzir a senha errada.

6. Como procurar um ficheiro

Poderá procurar um nome de ficheiro específico por meio de um destes três métodos.

- Busca por movimentação — desloque os nomes dos ficheiros na lista de programas.
- Busca pelo nome do ficheiro — introduza o nome do ficheiro.
- Busca pelos caracteres iniciais — introduza as primeiras letras do nome do ficheiro.

•Como encontrar um ficheiro por meio da busca por deslocamento

Exemplo Como usar a busca por deslocamento para chamar o programa com o nome OCTA.

1. Enquanto visualiza a lista de programas, utilize \blacktriangle e \blacktriangledown para deslocar a lista de nomes de programas até encontrar o que deseja.

```

Program List
OCTA
TRIANGLE
AREA *
MEASURE
[EXE EDIT NEW]

```

F_2

2. Quando o realce se encontrar sobre o nome do ficheiro que deseja, prima F_2 (EDIT) para o voltar a chamar.

F_2 (EDIT)

```

= OCTA =
2→A: 2×√3×A²,
√2+3×A^3

```

● **Como encontrar um ficheiro por meio da busca pelo nome do ficheiro**

Exemplo Como usar a busca pelo nome do ficheiro para chamar o programa com o nome OCTA.

1. Enquanto visualiza a lista de programas, prima **F3** (NEW) e introduza o nome do ficheiro que deseja encontrar.

F3 (NEW)
O **C** **T** **A**

2. Prima **EXE** para voltar a chamar o programa.

- Se não houver nenhum programa com o nome de ficheiro introduzido, será criado um ficheiro novo com esse nome.

● **Como encontrar um ficheiro por meio da busca dos caracteres iniciais**

Exemplo Como usar a busca dos caracteres iniciais para chamar o programa com o nome OCTA.

1. Enquanto visualiza a lista de programas, prima **F3** (SRC) e introduza os caracteres iniciais do ficheiro que deseja encontrar.

F3 (SRC)
O **C** **T**

2. Prima **EXE** para procurar.

EXE

- Serão chamados todos os ficheiros cujos nomes comecem pelos caracteres introduzidos.
 - Se não houver nenhum programa com o nome de ficheiro introduzido, aparecerá no visor o aviso **"Not Found"** (não encontrado). Se isso acontecer, prima **QUIT** para apagar o aviso de erro.
3. Utilize **▲** e **▼** para destacar o nome do ficheiro do programa que quer voltar a chamar e depois, prima **F2** (EDIT) para o fazer.

7. Como editar o conteúdo de um programa

● **Como editar o conteúdo de um programa**

1. Procure, na lista de programas, o nome do ficheiro do programa que deseja.



2. Chamar o programa.

- Os procedimentos a usar para editar os conteúdos de programas são idênticos aos usados para editar os cálculos manuais. Para mais detalhes, veja “Como fazer correções”.

- As teclas de função seguintes também são úteis quando se faz a edição dos conteúdos dos programas.

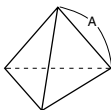
F1 (TOP) Move o cursor para a parte superior do programa.

```
= OCTA =
2→A: 2×√3×A²,
√2÷3×A³
```

F2 (BTM) Move o cursor para a parte inferior do programa.

```
= OCTA =
?→A: 2×√3×A²,
√2÷3×A³_
```

Exemplo 2 Como usar o programa OCTA para criar um programa que calcule a área da superfície e o volume de tetraedros regulares, dada a medida de um lado.



Comprimento do lado (A)	Área da superfície (S)	Volume (V)
7 cm	cm ²	cm ³
10 cm	cm ²	cm ³
15 cm	cm ²	cm ³

Seguem-se as fórmulas utilizados para o cálculo da área da superfície S e o volume V de um tetraedro regular, dada a medida de um lado.

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

Quando introduzir o programa, utilize as seguintes teclas de função.

Medida do lado A **SHIFT** **PRGM** **▷** **F1** (?) **←** **ALPHA** **A** **▷** **F3** (:)

Área da superfície S **SHIFT** **✓** **3** **×** **ALPHA** **A** **x²** **▷** **▷** **F2** (▲)

Volume V **SHIFT** **✓** **2** **÷** **1** **2** **×** **ALPHA** **A** **^** **3**

Compare isto com o programa para calcular a área da superfície e o volume dum octaedro regular.

Medida do lado A **SHIFT** **PRGM** **▷** **F1** (?) **←** **ALPHA** **A** **▷** **F3** (:)

Área da superfície S **2** **×** **SHIFT** **✓** **3** **×** **ALPHA** **A** **x²** **▷** **▷** **F2** (▲)

Volume V **SHIFT** **✓** **2** **÷** **3** **×** **ALPHA** **A** **^** **3**

Como vê, pode obter o programa TETRA, fazendo algumas alterações no programa OCTA.

- Apagar **2** **X** (sublinhado por meio de uma linha curva por cima)
- Alterar **3** para **1** **2** (sublinhado por meio de uma linha recta por cima)

Façamos a edição do programa.

F2(EDIT)

```
= OCTA =
2→A: 2×√3×A²,
√2÷3×A³
```

▶▶▶▶ DEL DEL

```
= OCTA =
?→A: √3×A²,
√2÷3×A³
```

▼◀ SHIFT INS 1 2

```
= OCTA =
?→A: √3×A²,
√2÷123×A³
```

DEL

```
= OCTA =
?→A: √3×A²,
√2÷12X A³
```

QUIT

Tentemos agora executá-lo.

Comprimento do lado (A)	Área da superfície (S)	Volume (V)
7 cm	84,87048957 cm ²	40,42293766 cm ³
10 cm	173,2050808 cm ²	117,8511302 cm ³
15 cm	389,7114317 cm ²	397,7475644 cm ³

```
Program List
OCTA
```

```
[EXE EDIT NEW]
```

F1

F1 (EXE) ou **EXE**

```
?
```

7 EXE

(Valor de A)

```
?
7
84.87048957
- DISP -
```

EXE

```

?
7
84.87048957
40.42293766
    
```

EXE

```

?
7
84.87048957
40.42293766
?
    
```

1 0 EXE

```

40.42293766
?
i0
173.2050808
- DISP -
    
```

EXE

```

40.42293766
?
i0
173.2050808
117.8511302
    
```

⋮

⋮

8. Como apagar um programa

Há duas maneiras diferentes de apagar um nome de ficheiro e respectivo programa.

- Apagar o programa específico
- Apagar todos os programas

•Como apagar um programa específico

1. Enquanto visualiza a lista de programas, utilize **▲** e **▼** para mover o realce para a especificação do programa que quer apagar.
2. Prima **▶ F1** (DEL).

▶ F1(DEL)

```

| YES | NO |
|-----|
| F1 | F4 |
    
```

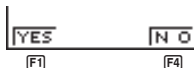
3. Prima **F1** (YES) para apagar o programa seleccionado ou **F4** (NO) para cancelar a operação sem apagar nada.

•Como apagar todos os programas

1. Enquanto visualiza a lista de programas, prima **▶ F2** (DEL•A).



▶ **F2** (DEL•A)



2. Prima **F1** (YES) para apagar todos os programas da lista ou **F4** (NO) para cancelar a operação sem apagar nada.

- Poderá ainda apagar todos os programas, por meio do **modo MEM**. Veja “Limpeza do conteúdo da memória”, para mais detalhes.

9. Comandos úteis de programação

Além dos comandos de cálculos, esta calculadora tem ainda uma variedade de comandos de relação e de salto, que poderão ser usados para criar programas que tornam rápidos e fáceis os cálculos repetidos.

Menu de programas

Prima **SHIFT** **PRGM** para visualizar o menu de programas.

SHIFT **PRGM**



F1 (COM) Menu de comandos dos programas

F2 (CTL) Menu dos comandos de controlo

F3 (JUMP) Menu dos comandos de salto

▶



F1 (?) Comando de entrada

F2 (▲) Comando de saída

F3 (CLR) Menu de comandos de apagar

F4 (DISP) Menu de comandos do visor

▶



F1 (REL) Menu dos operadores relacionais de saltos condicionais

F2 (I/O) Menu de comandos de entrada/saída

F3 (:) Comando de instrução múltipla

Prima **▶** para voltar ao menu anterior.

Menu de comandos dos programas (COM)

Enquanto visualiza o menu de programas, prima **F1** (COM) para visualizar o menu de comandos dos programas.

F1(COM)



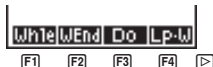
- F1** (If) Comando If
- F2** (Then) Comando Then
- F3** (Else) Comando Else
- F4** (If•End) Comando IfEnd

▶



- F1** (For) Comando For
- F2** (To) Comando To
- F3** (Step) Comando Step
- F4** (Next) Comando Next

▶



- F1** (While) Comando While
- F2** (WEnd) Comando WhileEnd
- F3** (Do) Comando Do
- F4** (Lp•W) Comando LpWhile

Prima ▶ para voltar ao menu anterior.

Menu de comandos dos controlos (CTL)

Enquanto visualiza o menu de programas, prima **F2** (CTL) para visualizar o menu de comandos dos controlos.

F2(CTL)

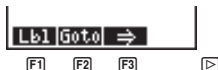


- F1** (Prog) Comando Prog
- F2** (Rtrn) Comando Return
- F3** (Brk) Comando Break
- F4** (Stop) Comando Stop

Menu de comandos de salto (JUMP)

Enquanto visualiza o menu de programas, prima **F3** (JUMP) para visualizar o menu de comandos de salto.

F3(JUMP)



- F1** (Lbl) Comando Lbl
- F2** (Goto) Comando Goto
- F3** (=>) Comando => (jump)

▶



- F1** (Lsz) Comando Lsz
- F2** (Dsz) Comando Dsz

Prima **▶** para voltar ao menu anterior.

Menu de comandos de apagar (CLR)

Enquanto visualiza o menu de programas, prima **▶ F3** (CLR) para visualizar o menu de comandos de apagar.

▶ F3(CLR)



- F1** (Text) Comando ClrText
- F2** (Grph) Comando ClrGraph
- F3** (List) Comando ClrList

Menu de comandos do visor (DISP)

Enquanto visualiza o menu de programas, prima **▶ F4** (DISP) para visualizar o menu de comandos do visor.

▶ F4(DISP)



- F1** (Stat) Comando DrawStat
- F2** (Grph) Comando DrawGraph
- F3** (TABL) Menu dos comandos de tabelas e gráficos

Ao premir **F3** (TABL) enquanto visualiza o menu de comandos do visor, fará aparecer o menu de comandos das tabelas e gráficos.

F3 (TABL)



F1 (Tabl) Comando DispTable

F2 (G-Con) Comando DrawTG-Con

F3 (G-Plt) Comando DrawTG-Plt

Menu dos operadores relacionais de saltos condicionais (REL)

Enquanto visualiza o menu de programas, prima $\rightarrow \rightarrow$ F1 (REL) para visualizar o menu dos operadores relacionais de saltos condicionais.

 $\rightarrow \rightarrow$ F1 (REL)

F1 (=) Operador relacional =

F2 (≠) Operador relacional ≠

F3 (>) Operador relacional >

F4 (<) Operador relacional <

 \rightarrow 

F1 (≥) Operador relacional ≥

F2 (≤) Operador relacional ≤

Prima \rightarrow para voltar ao menu anterior.**Menu dos comandos de entrada/saída (I/O)**

Enquanto visualiza o menu de programas, prima $\rightarrow \rightarrow$ F2 (I/O) para visualizar o menu de comandos de entrada/saída.

 $\rightarrow \rightarrow$ F2 (I/O)

F1 (Send) Comando Send (

F2 (Recv) Comando Receive (

10. Referência dos comandos

■ Índice de comandos

Break	149
ClrGraph	153
ClrList.....	153
ClrText.....	153
DispTable	154
Do~LpWhile	148
DrawTG-Con, DrawTG-Plt.....	154
DrawGraph	154
DrawStat	153
Dsz	151
For~To~Next	147
For~To~Step~Next	147
Goto~Lbl	151
If~Then	145
If~Then~Else	146
If~Then~Else~IfEnd	146
If~Then~IfEnd	145
Isz	152
Prog	149
Receive(.....	154
Return	150
Send(.....	155
Stop	150
While~WhileEnd	148
? (Comando de entrada)	144
▲ (Comando de saída)	144
: (Comando de instruções múltiplas)	144
↵ (Mudança de linha)	144
⇒ (Código de salto)	152
=, ≠, >, <, ≥, ≤ (Operadores relacionais)	155

Seguem-se as convenções utilizadas nesta secção para descrever os vários comandos.

Texto a **negrito**..... Os comandos e outros itens que têm sempre de ser introduzidos são mostrados a **negrito**.

{chaves} Servem para encerrar um número de itens, dos quais um tem de ser seleccionado no uso dum comando. Não introduza as chaves quando estiver a introduzir um comando.

- [Colchetes] Servem para encerrar os itens opcionais. Não introduza colchetes quando estiver a introduzir um comando.
- Expressões numéricas ... As expressões numéricas (tais como 10, 10+20, A) indicam constantes, cálculos, constantes numéricas, etc.
- Caracteres alfabéticos Indicam encadeamentos literais (Tal como AB).

■ Comandos de operações básicas

? (Comando de entrada)

Função: Indicação para a introdução de valores para atribuição a variáveis, durante a execução de programas.

Sintaxe: ? → <nome da variável>

Exemplo: ? → A ↵

Descrição:

1. Este comando interrompe momentaneamente a execução do programa e indica que se deve introduzir um valor ou expressão a atribuir a uma variável. Quando é executado o comando de entrada, aparecerá “?” no visor e a calculadora pára para que se introduza.
2. A introdução a fazer em resposta a este comando terá de ser de um valor ou de uma expressão, e esta não poderá ser de múltipla instrução.

▲ (Comando de saída)

Função: Faz visualizar um resultado intermédio durante a execução de programas.

Descrição:

1. Este comando interrompe momentaneamente a execução do programa e faz visualizar o texto em caracteres alfabéticos ou o resultado do último cálculo feito.
2. O comando de saída devia ser usado em situações em que premiria normalmente a tecla **EXE**, durante um cálculo manual.

: (Comando de instruções múltiplas)

Função: Une duas instruções para execução sequencial contínua.

Descrição:

1. Ao contrário do comando de saída (▲) as instruções ligadas pelo comando de instruções múltiplas são executadas de forma contínua.
2. Este comando poderá ser usado para unir duas expressões de cálculo ou dois comandos.
3. Poderá também usar um retorno de carro indicado por ↵ em lugar do comando de instruções múltiplas.

↵ (Mudança de linha)

Função: Une duas instruções para execução sequencial contínua.

Descrição:





1. A operação de mudança de linha é idêntica à do comando de instruções múltiplas.
2. O uso da mudança de linha em lugar do comando de instruções múltiplas facilita a leitura do programa visualizado.

■ Comandos de programação (COM)

If~Then

Função: A instrução Then (logo) é executada apenas quando a condição If (Se) for verdadeira (diferente de 0).

Sintaxe:


If <condição>   Then <instrução>   <instrução>

expressão numérica

Parâmetros: condição, expressão numérica

Descrição:



1. A instrução Then é executada apenas se a condição If for verdadeira (diferente de 0).
2. Se a condição for falsa (0), a instrução Then não será executada.
3. Uma condição If terá sempre de ser acompanhada por uma instrução Then. Ao omitir tal, provocará uma situação de erro (Syn ERROR).

Exemplo: If A = 0 
Then "A = 0"





If~Then~IfEnd

Função: A instrução Then (logo) é executada apenas quando a condição If (Se) for verdadeira (diferente de 0). A instrução IfEnd será sempre executada: depois de executada a instrução Then ou directamente depois da condição If, se esta for falsa (0).

Sintaxe:

If <condição>   Then <instrução>




expressão numérica

  <instrução>   IfEnd

Parâmetros: condição, expressão numérica

Descrição:

Este comando é quase idêntico ao If~Then. A única diferença está em que a instrução IfEnd é sempre executada, seja a condição If verdadeira (diferente de 0) ou falsa (0).

Exemplo: If A = 0 
Then "A = 0" 
IfEnd 
"END"

If~Then~Else

Função: A instrução Then (logo) é executada apenas quando a condição If (Se) for verdadeira (diferente de 0). A instrução Else será executada se a condição If for falsa (0).

Sintaxe:

$$\text{If } \begin{array}{l} \text{<condição>} \\ \text{expressão numérica} \end{array} \left\{ \begin{array}{l} \downarrow \\ \vdots \\ \uparrow \end{array} \right\} \text{ Then } \text{<instrução>} \left[\begin{array}{l} \downarrow \\ \vdots \\ \uparrow \end{array} \right] \text{<instrução>} \\ \left\{ \begin{array}{l} \downarrow \\ \vdots \\ \uparrow \end{array} \right\} \text{ Else } \text{<instrução>} \left[\begin{array}{l} \downarrow \\ \vdots \\ \uparrow \end{array} \right] \text{<instrução>} \left[\begin{array}{l} \downarrow \\ \vdots \\ \uparrow \end{array} \right]$$

Parâmetros: condição, expressão numérica

Descrição:

1. A instrução Then é executada quando a condição If for verdadeira (diferente de 0).
2. A instrução Else é executada quando a condição If for falsa (0).

Exemplo: If A = 0 \downarrow
 Then "TRUE" \downarrow
 Else "FALSE"

If~Then~Else~IfEnd

Função: A instrução Then (logo) é executada apenas quando a condição If (Se) for verdadeira (diferente de 0). A instrução Else é executada quando a condição If for falsa (0). A instrução IfEnd será sempre executada depois da instrução Then e da instrução Else.

Sintaxe:

$$\text{If } \begin{array}{l} \text{<condição>} \\ \text{expressão numérica} \end{array} \left\{ \begin{array}{l} \downarrow \\ \vdots \\ \uparrow \end{array} \right\} \text{ Then } \text{<instrução>} \left[\begin{array}{l} \downarrow \\ \vdots \\ \uparrow \end{array} \right] \text{<instrução>} \\ \left\{ \begin{array}{l} \downarrow \\ \vdots \\ \uparrow \end{array} \right\} \text{ Else } \text{<instrução>} \left[\begin{array}{l} \downarrow \\ \vdots \\ \uparrow \end{array} \right] \text{<instrução>} \left[\begin{array}{l} \downarrow \\ \vdots \\ \uparrow \end{array} \right] \text{ IfEnd}$$

Parâmetros: condição, expressão numérica

Descrição:

Este comando é quase idêntico ao If~Then~Else. A única diferença está em que a instrução IfEnd é sempre executada, seja a condição If verdadeira (diferente de 0) ou falsa (0).

Exemplo: ? \rightarrow A \downarrow
 If A = 0 \downarrow
 Then "TRUE" \downarrow
 Else "FALSE" \downarrow
 IfEnd \downarrow
 "END"

For~To~Next

Função: Este comando repete tudo que estiver entre a instrução For e a instrução Next. O valor inicial é atribuído à variável de controle na primeira execução e este valor será aumentado de um por cada execução. A execução prosseguirá até que o valor da variável de controle exceda a variável final.

Sintaxe:

For <valor inicial> → <nome da variável de controle> To <valor final> {
 ↓
 :
 ↙

[<instrução> {
 ↓
 :
 ↙ }] Next

Parâmetros:

- nome da variável de controle: A a Z
- valor inicial: valor ou expressão que produz um valor (i.e. sin x , A, etc.)
- valor final: valor ou expressão que produz um valor (i.e. sin x , A, etc.)

Descrição:

1. Quando o valor inicial da variável de controle for maior do que o valor final, a execução prosseguirá a partir da instrução a seguir a Next, sem executar as instruções compreendidas entre For e Next.
2. Uma instrução For tem de ter sempre uma correspondente instrução Next, e esta terá sempre de vir a seguir à instrução For sua correspondente.
3. A instrução Next define o fim duma sequência criada por For~Next, e tem, por isso, que ser incluída. Se assim não for, provocará um erro (Syn ERROR).

Exemplo: For 1 → A To 10 ↓

A × 3 → B ↓

B ↙

Next

For~To~Step~Next

Função: Este comando repete tudo que estiver entre a instrução For e a instrução Next. O valor inicial é atribuído à variável de controle na primeira execução e este valor será aumentado de um por cada execução. A execução prosseguirá até que o valor da variável de controle exceda a variável final.

Sintaxe:

For <valor inicial> → <nome da variável de controle> To <valor final> Step <valor do incremento> {
 ↓
 :
 ↙

Next

Parâmetros:

- nome da variável de controle: A a Z
- valor inicial: valor ou expressão que produz um valor (i.e. sin x , A, etc.)
- valor final: valor ou expressão que produz um valor (i.e. sin x , A, etc.)
- valor do incremento: valor numérico (ao omitir este valor, passará a ser 1)

Descrição:

1. Este comando é basicamente idêntico ao For~To~Next. A única diferença está em que poderá especificar o incremento.
2. Ao omitir o valor do incremento este passará automaticamente a ser 1.
3. Ao especificar o valor inicial menor do que o valor final e um valor de incremento positivo, fará com que a variável de controlo seja aumentada em cada execução. Se, pelo contrário, especificar um valor inicial maior do que o valor final e um valor de incremento negativo, fará com que a variável de controlo seja diminuída em cada execução.

Exemplo: For 1 → A To 10 Step 0.1 ↵

A × 3 → B ↵

B ▲

Next

Do~LpWhile

Função: Este comando repete comandos específicos, enquanto a respectiva condição for verdadeira (diferente de 0).

Sintaxe:

Do { ↵
: ↵
▲ } ~ LpWhile <expressão>

Parâmetros: expressão

Descrição:

1. Este comando repete os comandos contidos no círculo, enquanto a respectiva condição for verdadeira (diferente de 0). Quando essa condição se torna falsa (0), a execução prossegue a partir da instrução que se segue à instrução LpWhile.
2. Como a condição vem depois da instrução LpWhile, é testada (verificada) depois de terem sido executados todos os comandos compreendidos no círculo.

Exemplo: Do ↵

? → A ↵

A × 2 → B ↵

B ▲

LpWhile B >10

While~WhileEnd

Função: Este comando repete comandos específicos, enquanto a respectiva condição for verdadeira (diferente de 0).

Sintaxe:

While <expressão> { ↵
: ↵
▲ } ~ WhileEnd

Parâmetros: expressão

Descrição:

1. Este comando repete os comandos contidos no círculo, enquanto a respectiva condição for verdadeira (diferente de 0). Quando essa condição se torna falsa (0), a execução prossegue a partir da instrução que se segue à instrução WhileEnd.
2. Como a condição vem depois da instrução While, é testada (verificada) depois de terem sido executados todos os comandos compreendidos no círculo.

Exemplo: 10 → A ↵
 While A > 0 ↵
 A - 1 → A ↵
 "GOOD" ↵
 WhileEnd

■ **Comandos de controlo dos programas (CTL)**

Break

Função: Este comando suspende a execução de um círculo e continua a partir do comando que se segue a esse círculo.

Sintaxe: Break ↵

Descrição:

1. Este comando suspende a execução de um círculo e continua a partir do comando que se segue a esse círculo.
2. Este comando poderá ser utilizado para suspender a execução de uma instrução For, uma instrução Do e uma instrução While.

Exemplo: While A>0 ↵
 If A > 2 ↵
 Then Break ↵
 IfEnd ↵
 WhileEnd ↵
 A ▲ ←————— Executada depois de Break

Prog

Função: Este comando especifica a execução de outro programa como sub-rotina. No modo RUN, este comando executará um programa novo.

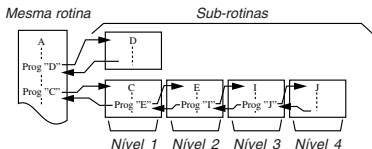
Sintaxe: Prog "nome do ficheiro" ↵

Exemplo: Prog "ABC" ↵

Descrição:

1. Mesmo quando este comando está localizado dentro dum círculo, a sua execução suspende imediatamente o círculo e inicia a sub-rotina.
2. Este comando poderá ser utilizado tantas vezes quantas necessárias, numa rotina principal, para chamar sub-rotinas independentes a executarem tarefas específicas.

3. Uma sub-rotina poderá ser utilizada em múltiplas posições dentro da mesma rotina principal, ou então, ser chamada por um número qualquer de rotinas principais.



4. Ao chamar uma sub-rotina fará com que esta seja executada desde o início. Assim que a sua execução esteja terminada, esta voltará à rotina principal, continuando a partir da instrução que se segue ao comando Prog.
5. Um comando Goto~Lbl dentro de uma sub-rotina será apenas válido dentro dessa rotina. Não poderá ser utilizado para saltar para uma etiqueta fora da sub-rotina.
6. Se não existir uma sub-rotina com o nome de ficheiro especificado pelo comando Prog, ocorrerá um erro (Go ERROR).
7. No **Modo RUN**, ao introduzir o comando Prog e premir **[EXE]**, inicia o programa especificado pelo comando.

Return

Função: Este comando faz voltar de uma subrotina.

Sintaxe: Return ↵

Descrição:

A execução do comando Return dentro duma rotina principal faz parar a execução do programa.

Exemplo:

Prog "A"	Prog "B"
1 → A ↵	For A → B To 10 ↵
Prog "B" ↵	B + 1 → C ↵
C ▲	Next ↵
	Return

Ao executar o programa no Ficheiro A, visualizará o resultado da operação (11)

Stop

Função: Este comando termina a execução dum programa.

Sintaxe: Stop ↵

Descrição:

- Este comando termina a execução do programa.
- A execução deste comando dentro dum círculo termina a execução do programa sem que seja originado um erro.

```
Exemplo: For 2 → 1 To 10 ↵
          If I = 5 ↵
          Then "STOP" : Stop ↵
          IfEnd ↵
          Next
```

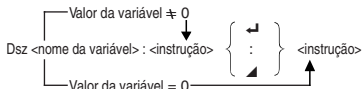
Este programa conta de 2 a 10. Quando chega a 5, contudo, termina a execução e fará aparecer o aviso "STOP".

■ Comandos de salto (JUMP)

Dsz

Função: Este comando é um salto de contagem que diminui de 1, o valor de uma variável de controlo e depois, salta, se o valor actual da variável for zero.

Sintaxe:



Parâmetros:

Nome da variável: A a Z

[Exemplo] Dsz B: Diminui de 1 o valor atribuído à variável B.

Descrição:

Este comando diminui de 1, o valor da variável de controlo, e depois testa-a (verifica-a). Se o valor actual for diferente de zero, a execução prossegue com a instrução seguinte. Se o valor actual for zero, a execução salta para a instrução que se segue ao comando de instruções múltiplas (:), comando do visor (↵) ou retorno do carro (↵).

Exemplo: 10 → A : 0 → C :

Lbl 1 : ? → B : B+C → C :

Dsz A : Goto 1 : C + 10

Este programa indica a introdução de 10 valores e depois calcula a média dos valores introduzidos.

Goto~Lbl

Função: Este comando efectua um salto incondicional para um local específico.

Sintaxe: Goto <valor ou variável> ~ Lbl <valor ou variável>

Parâmetros: Valor (de 0 a 9), variável (A a Z)

Descrição:

1. Este comando consiste de duas partes: Goto *n* (em que *n* é um valor de 0 a 9) e Lbl *n* (em que *n* é um valor especificado para Goto). Este comando faz com que a execução do programa dê um salto para a instrução Lbl cujo valor coincida com o especificado para a instrução Goto.
2. Este comando poderá ser usado para voltar ao princípio de um programa ou para saltar para qualquer local dentro do programa.

- Este comando poderá ser usado em combinação com saltos condicionais e saltos de contagem.
- Se não houver uma instrução Lbl cujo valor seja adequado ao da instrução Goto, ocorrerá um erro (Go ERROR).

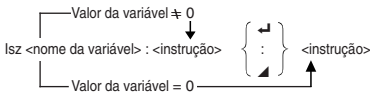
Exemplo: ? → A : ? → B : Lbl 1 :
 ? → X : A × X + B ▲
 Goto 1

Este programa calcula $y = AX + B$ para tantos valores para cada variável quantos queira introduzir. Para sair da execução deste programa, prima **AC**.

Isz

Função: Este comando é um salto de contagem que incrementa de 1, o valor de uma variável de controlo e depois, salta, se o valor actual da variável for zero.

Sintaxe:



Parâmetros:

Especificação da variável: A a Z

[Exemplo] Isz A: Incrementa em 1 o valor atribuído à variável A.

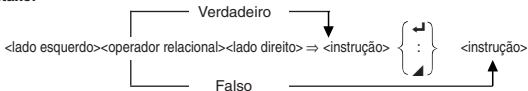
Descrição:

Este comando incrementa de 1, o valor da variável de controlo, e depois testa-a (verifica-a). Se o valor actual for diferente de zero, a execução prossegue com a instrução seguinte. Se o valor actual for zero, a execução salta para a instrução que se segue ao comando de instruções múltiplas (:), comando do visor (▲) ou retorno do carro (↵).

⇒ (Código de salto)

Função: Este código é utilizado para preparar as condições para um salto condicional. O salto é executado sempre que as condições forem falsas.

Sintaxe:



Parâmetros:

lado esquerdo/lado direito: variável (A a Z); constante numérica, expressão com variável (tal como: $A \times 2$)

operador relacional: =, ≠, >, <, ≥, ≤

Descrição:

1. O salto condicional compara o conteúdo de duas variáveis ou os resultados de duas expressões, e toma uma decisão quanto a executar ou não o salto com base nos resultados da comparação.
2. Se a comparação resultar num resultado verdadeiro, a execução prossegue com a instrução que segue o comando \Rightarrow . Caso contrário, a execução salta para as instruções que se seguem ao comando de instruções múltiplas (:), comando do visor (\blacktriangleleft) ou mudança de linha (\blacktriangledown).

Exemplo: Lbl 1 : ? \rightarrow A :

$A \geq 0 \Rightarrow \sqrt{A} \blacktriangleleft$

Goto 1

Com este programa, a introdução dum valor de 0 ou superior, calcula e faz visualizar a raiz quadrada do valor introduzido. A introdução dum valor inferior a 0, faz voltar no prompt, sem fazer cálculo algum.

■ Comandos de apagar (CLR)

ClrGraph

Função: Este comando apaga o ecrã de gráfico.

Sintaxe: ClrGraph \blacktriangledown

Descrição: Este comando apaga o ecrã de gráfico, durante a execução de programas.

ClrList

Função: Este comando apaga os dados de lista.

Sintaxe: ClrList \blacktriangledown

Descrição: Este comando apaga o conteúdo da lista actualmente seleccionada (da List 1 à List 6), durante a execução de programas.

ClrText

Função: Este comando apaga o ecrã de texto.

Sintaxe: ClrText \blacktriangledown

Descrição: Este comando apaga o texto do ecrã, durante a execução de programas.

■ Comandos de visualização (DISP)

DrawStat

Função: Este comando desenha um gráfico estatístico.

Sintaxe:

DrawStat \blacktriangledown

Descrição:

Este comando desenha um gráfico estatístico de acordo com as condições definidas no programa.

DrawGraph

Função: Este comando desenha um gráfico.

Sintaxe: DrawGraph ↵

Descrição: Este comando desenha um gráfico de acordo com as condições de desenho definidas no programa.

DispTable

Função: Estes comandos fazem visualizar tabelas numéricas.

Sintaxe:

DispTable ↵

Descrição:

Estes comandos criam tabelas numéricas durante a execução de programas de acordo com as condições definidas no programa.

DrawTG-Con, DrawTG-Plt

Função: Estes comandos fazem gráficos de funções.

Sintaxe:

DrawTG-Con ↵

DrawTG-Plt ↵

Descrição:

1. Estes comandos fazem gráficos de funções de acordo com as condições definidas no programa.
2. O comando DrawTG-Con produz um gráfico de tipo de ligação, enquanto que o comando DrawTG-Plt produz um gráfico do tipo plotagem.

■ Comandos de entrada/saída (I/O)

Receive (

Função: Este comando recebe dados de um dispositivo externo.

Sintaxe: Receive (<dados>) (...ex. Receive (List 1))

Descrição:

1. Este comando recebe dados de um dispositivo externo.
2. Os seguintes tipos de dados podem ser recebidos por este comando.
 - Valores individuais designados para variáveis
 - Dados de lista (todos os valores - valores individuais não podem ser especificados)

Send (

Função: Este comando envia dados para um dispositivo externo.

Sintaxe: Send (<dados>) (...ex. Send (List 1))

Descrição:


1. Este comando envia dados para um dispositivo externo.
2. Os seguintes tipos de dados podem ser enviados por este comando.
 - Valores individuais designados para variáveis
 - Dados de lista (todos os valores - valores individuais não podem ser especificados)

■ Operadores relacionais de saltos condicionais (REL)

=, ≠, >, <, ≥, ≤

Função: Estes operadores relacionais são utilizados em combinação com o comando de salto condicional.

Sintaxe:

<lado esquerdo><operador relacional><lado direito> ⇒ <instrução>  <instrução>

(Com código de salto)

Parâmetros:

lado esquerdo/lado direito: variável (A a Z), constante numérica, expressão com variável (tal como: A × 2)

operadores relacionais: =, ≠, >, <, ≥, ≤

Descrição:

1. Os seguintes seis operadores relacionais poderão ser usados no comando de salto condicional.

<lado esquerdo> = <lado direito> : verdadeiro quando <lado esquerdo> igual ao <lado direito>

<lado esquerdo> ≠ <lado direito> : verdadeiro quando <lado esquerdo> não é igual ao <lado direito>

<lado esquerdo> > <lado direito>: verdadeiro quando <lado esquerdo> é maior que <lado direito>

<lado esquerdo> < <lado direito>: verdadeiro quando <lado esquerdo> é menor que <lado direito>

<lado esquerdo> ≥ <lado direito>: verdadeiro quando <lado esquerdo> é maior ou igual ao <lado direito>

<lado esquerdo> ≤ <lado direito>: verdadeiro quando <lado esquerdo> é menor ou igual ao <lado direito>

2. Veja “⇒ (Código de salto)”, para mais detalhes sobre o uso do salto condicional.

11. Visualização de texto

Poderá incluir texto num programa, com a simples inclusão deste entre duas aspas. Tal texto aparecerá no visor durante a execução do programa, o que quer dizer que poderá acrescentar etiquetas às indicações de introdução e aos resultados.

Programa	Visualização
? → X	?
"X = " ? → X	X = ?

- Se o texto for seguido de uma fórmula de cálculo, não deixe de inserir um comando de visor (↵) ou um comando de instrução múltipla (:) entre o texto e o cálculo.
- Ao introduzir mais de 13 caracteres, fará com que o texto se desloque para a linha seguinte. O ecrã subirá automaticamente se o texto encher o ecrã.

12. Como utilizar as funções de calculadora em programas

■ Como utilizar funções de gráfico num programa

Poderá incorporar funções de gráfico num programa, a fim de desenhar gráficos complexos e sobrepor gráficos. A seguir, mostra-se vários tipos de passos que terá de usar quando estiver a programar com funções de gráfico.

- Janela de visualização
View Window -5, 5, 1, -5, 5, 1 ↵
- Introdução da função de gráfico
Y = Type ↵ Especifica o tipo de gráfico
"X² - 3" → Y1 ↵
- Operação de desenho de gráfico
DrawGraph ↵

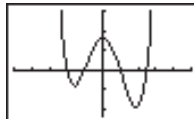
Exemplo de programa

- ① ClrGraph ↵
- ② View Window -10, 10, 2, -120, 150, 50 ↵
- ③ Y = Type ↵
"X ^ 4 - X ^ 3 - 24X² + 4X + 80" → Y1 ↵
- ④ G SelOn 1 ↵
- ⑤ DrawGraph

- ① SHIFT PROG > F3 F2
- ② SHIFT F3 F1 QUIT
- ③ F3 F3 F2 F1 QUIT
- ④ VARS > F2 F1 QUIT
- ⑤ F3 F3 F1 F1
- ⑥ SHIFT PROG > F4 F2



Ao executar este programa produzirá o resultado aqui ilustrado.



■ Como utilizar funções de tabelas e gráficos num programa

As funções de tabelas e gráficos num programa podem criar tabelas numéricas e efectuar operações de gráfico. Seguem-se vários tipos de sintaxe que terá de seguir para programar funções de tabelas e gráficos.

- Fixação dos limites da tabela
 - 1 → F Start ↵
 - 5 → F End ↵
 - 1 → F pitch ↵
- Criação de tabelas numéricas
 - DispTable ↵
- Operação de desenho de gráfico
 - Tipo ligação: DrawTG-Con ↵
 - Tipo plotagem: DrawTG-Plt ↵

Exemplo de programa

```
ClrGraph ↵
ClrText ↵
View Window 0, 6, 1, -2, 106, 20 ↵
Y = Type ↵
"3X2 - 2" → Y1 ↵
```

- ① T SelOn 1 ↵
- 0 → ② F Start ↵
- 6 → ③ F End ↵
- 1 → ④ F pitch ↵
- ⑤ DispTable ↵
- ⑥ DrawTG-Con

- ① **F3** **F4** **F1** **QUIT**
- ② **VARS** **▶** **F3** **F1**
- ③ **F2**
- ④ **F3** **QUIT**
- ⑤ **SHIFT** **PRGM** **▶** **F4** **F3** **F1** **QUIT**
- ⑥ **SHIFT** **PRGM** **▶** **F4** **F3** **F2** **QUIT**

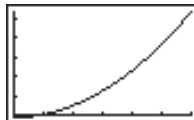
Ao executar este programa produzirá os resultados aqui ilustrados.

Tabela numérica

X	Y1
0	-2
1	1

Gráfico

EXE



P.82

■ Como utilizar funções de ordenamento de listas num programa

Estas funções permitem-lhe ordenar os dados nas listas, por ordem crescente ou decrescente.

- Ordem crescente

① SortA (② List 1, List 2, List 3)

— Listas a ser ordenadas (Pode-se especificar até 6)

① F3 F2 F1 QUIT

② OPTN F1 F1

- Ordem decrescente

SortD (List 1, List 2, List 3)

— Listas a ser ordenadas (Pode-se especificar até 6)



P.96

■ Como utilizar cálculos estatísticos e gráficos num programa

A inclusão de cálculos estatísticos e operações de criação de gráficos nos programas permite-lhe calcular e desenhar gráficos de dados estatísticos.

• Como programar as condições e desenhar um gráfico estatístico

A seguir a "StatGrph", tem de especificar as seguintes condições para os gráficos:

- Situação de desenhar/não desenhar gráfico (DrawOn/DrawOff)
- Tipo de gráfico
- Localização dos dados do eixo x (nome da lista)
- Localização dos dados do eixo y (nome da lista)
- Localização dos dados de frequência (nome da lista)
- Tipo de marca

As condições de gráfico requeridas dependem do tipo de gráfico. Veja “Como alterar os parâmetros dos gráficos”.

- A seguir indicamos uma especificação de condição de gráfico típica para um diagrama de dispersão ou gráfico de linhas xy .

S-Gph1 DrawOn, Scatter, List1, List2, 1, Square ↵

No caso do gráfico de linhas xy , substitua “Scatter” da especificação acima por “xyLine”.

- A seguir indicamos uma especificação de condição de gráfico típica para um gráfico de maçã.

S-Gph1 DrawOn, Pie, List1, % (formato de visualização dos dados) ↵

- A seguir indicamos uma especificação de condição de gráfico típica para um gráfico de barras empilhadas, um gráfico de barras e um gráfico de linhas.

Gráfico de barras empilhadas: S-Gph1 DrawOn, StackedBar, List1 ↵

Gráfico de barras: S-Gph1 DrawOn, Bar, List1 ↵

Gráfico de linhas: S-Gph1 DrawOn, LineG, List1 ↵

- A seguir indicamos uma especificação de condição de gráfico típica para um gráfico de barras e um gráfico de linhas sobrepostos.

S-Gph1 DrawOn, Both, List1 (lista de gráfico de barras), List2 (lista de gráfico de linha), Sep. G(definição AutoWin) ↵

- A seguir indicamos uma especificação de condição de gráfico típica para um gráfico de variável simples.

S-Gph1 DrawOn, Hist, List1, List2 ↵

O mesmo formato poderá ser usado para os seguintes tipos de gráficos, bastando substituir “Hist” da especificação acima pelo tipo de gráfico adequado.

Histograma: Hist

Caixa-média: MedBox

Distribuição normal: N-Dist

- A seguir indicamos uma especificação de condição de gráfico típica para um gráfico de regressão.

S-Gph1 DrawOn, Linear, List1, List2, List3 ↵

O mesmo formato poderá ser usado para os seguintes tipos de gráficos, bastando substituir “Linear” da especificação acima pelo tipo de gráfico adequado.

Regressão linear: Linear

Média-média: Med-Med

Regressão quadrática: .. Quad

Regressão logarítmica: ... Log

Regressão exponencial: Exp

Regressão potencial: Power

Exemplo de programa

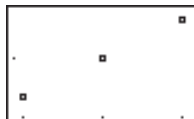
```

ClrGraph ↵
① S-WindAuto ↵
  {1, 2, 3} → ② List 1 ↵
  {1, 2, 3} → ③ List 2 ↵
④ S-Gph1 ⑤ DrawOn,
⑥ Scatter, List1, List2, 1, ⑦ Square ↵
⑧ DrawStat
    
```

```

① [SHIFT] [SETUP] [D] [D] [F1] [QUIT]
② [OPTN] [F1] [F1]
③ [F1] [QUIT]
④ [F3] [F1] [F2] [F1] [QUIT]
⑤ [F3] [F1] [F1] [F1] [QUIT]
⑥ [F3] [F1] [F2] [D] [F1] [QUIT]
⑦ [F3] [F1] [F4] [F1] [QUIT]
⑧ [SHIFT] [PRGM] [D] [F4] [F1] [QUIT]
    
```

Ao executar este programa produzirá o diagrama de dispersão aqui ilustrado.



■ Como efectuar cálculos estatísticos

- Cálculo estatístico de variável simples

① 1-Variable List 1, List 2

— Dados de frequência (Frequency)

— Dados do eixo x (XList)

① [F3] [F1] [D] [F1] [F1] [QUIT]

```

1-Variable
x̄ = 2.33333
Σx = 14
Σx² = 36
xón = 0.74535
    
```

- Cálculo estatístico de variáveis binárias

2-Variable List 1, List 2, List 3

— Dados de frequência (Frequency)

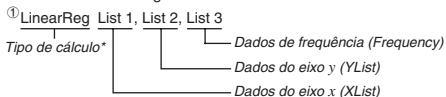
— Dados do eixo y (YList)

— Dados do eixo x (XList)

```

2-Variable
x̄ = 2.33333
Σx = 14
Σx² = 36
xón = 0.74535
    
```

• Cálculo estatístico da regressão



① **F3** **F1** **▷** **F1** **▷** **F1** **QUIT**

```
LinearReg
a= 0.64641
b=-0.71186
r= 0.87959
y=ax+b
```

* Pode ser especificado como tipo de cálculo, qualquer dos seguintes

- LinearReg regressão linear
- Med-MedLine .. cálculo de média-média
- QuadReg regressão quadrática
- LogReg regressão logarítmica
- ExpReg regressão exponencial
- PowerReg regressão potencial

