CURSO PRÁTICO 25 DE PROGRAMAÇÃO DE COMPUTADORES

INFO

PROGRAMAÇÃO BASIC - PROGRAMAÇÃO DE JOGOS - CÓDIGO DE MÁQUINA

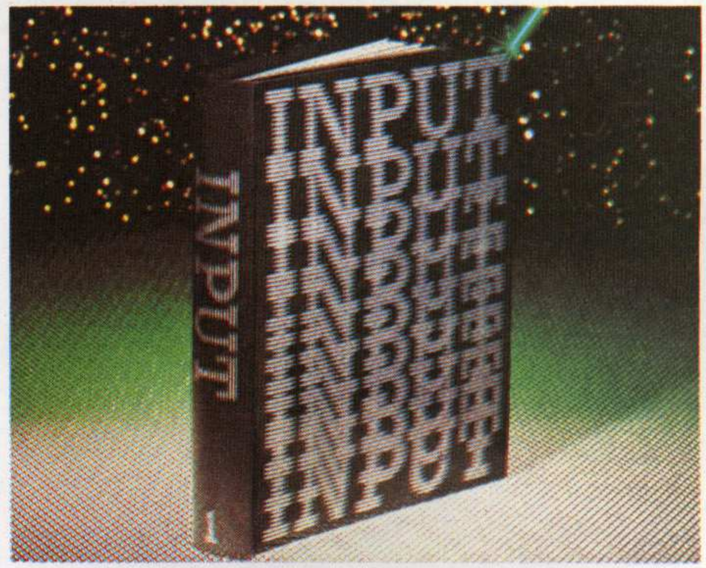
NOVA CULTURAL

Cz$ 39,00

INPUT

Vol. 2 Nº 25

NESTE NÚMERO



PLANO DA OBRA
INPUT é uma obra editada em fascículos semanais, e cada conjunto de 15 fascículos compõe um volume. A capa para encadernação de cada volume estará à venda oportunamente.

FÉRIAS, VIAGENS, MUDANÇAS...
NÃO FIQUE COM A COLEÇÃO INCOMPLETA
Se você está saindo de férias, pretende viajar ou vai se ausentar por algum tempo, avise antecipadamente seu jornaleiro. Ele pode guardar os seus fascículos enquanto você estiver fora. Se, por qualquer motivo, você perdeu alguns números, peça-os também a seu jornaleiro, ou entre em contato com nossa Distribuidora:

1. **Pessoalmente** — Em *São Paulo,* os endereços são: rua Brigadeiro Tobias, 773, Centro; av. Industrial, 117, Santo André. No *Rio de Janeiro,* av. Mem de Sá, 191/193, Centro.
2. **Por carta** — Envie para:
   DINAP — Distribuidora Nacional de Publicações
   Números Atrasados
   Estrada Velha de Osasco, 132 — Jardim Teresa
   CEP 06040 — Osasco — SP
3. **Por telex** — Utilize o nº (011) 33 670 DNAP.

Em **Portugal**, os pedidos devem ser feitos à Distribuidora Jardim de Publicações Lda. — Qta. Pau Varais, Azinhaga de Fetais, 2685, Camarate, Lisboa; Apartado 57; Telex 43 069 JARLIS P.
Atenção: Após seis meses do encerramento da coleção, o atendimento dos pedidos dependerá da disponibilidade do estoque.
Obs.: Quando pedir livros, mencione sempre o título e/ou autor da obra, além do número da edição.

COLABORE CONOSCO
Encaminhe seus comentários, críticas, sugestões ou reclamações ao
SERVIÇO DE ATENDIMENTO AO LEITOR
Caixa Postal 9 442, São Paulo — SP.

EDITOR
RICHARD CIVITA
NOVA CULTURAL

Presidente
Flávio Barros Pinto
Diretoria
Carmo Chagas, Iara Rodrigues, Pierluigi Bracco, Plácido Nicoletto, Roberto Silveira, Shozi Ikeda, Sônia Carvalho

REDAÇÃO
Diretor Editorial: Carmo Chagas
Editores Executivos: Antonio José Filho, Berta Sztark Amar
Editor Chefe: Paulo de Almeida
Editoras Assistentes: Ana Lúcia B. de Lucena, Marisa Soares de Andrade
Chefe de Arte: Carlos Luiz Batista
Assistentes de Arte: Dagmar Bastos Sampaio, Grace Alonso Arruda, Monica Lenardon Corradi

Secretária de Redação/Coordenadora: Stefania Crema
Secretário de Redação: Mauro de Queiroz

Colaboradores
Consultor Editorial Responsável: Dr. Renato M. E. Sabbatini (Diretor do Núcleo de Informática Biomédica da Universidade Estadual de Campinas-SP)
Execução Editorial: DATAQUEST Assessoria em Informática Ltda., Campinas
Tradução, adaptação, programação e redação: Abílio Pedro Neto, Aluísio J. Dornellas de Barros, Marcelo R. Pires Therezo, Marcos Huascar Velasco, Raul Neder Porrelli, Ricardo J. P. de Aquino Pereira
Coordenação Geral: Rejane Felizatti Sabbatini

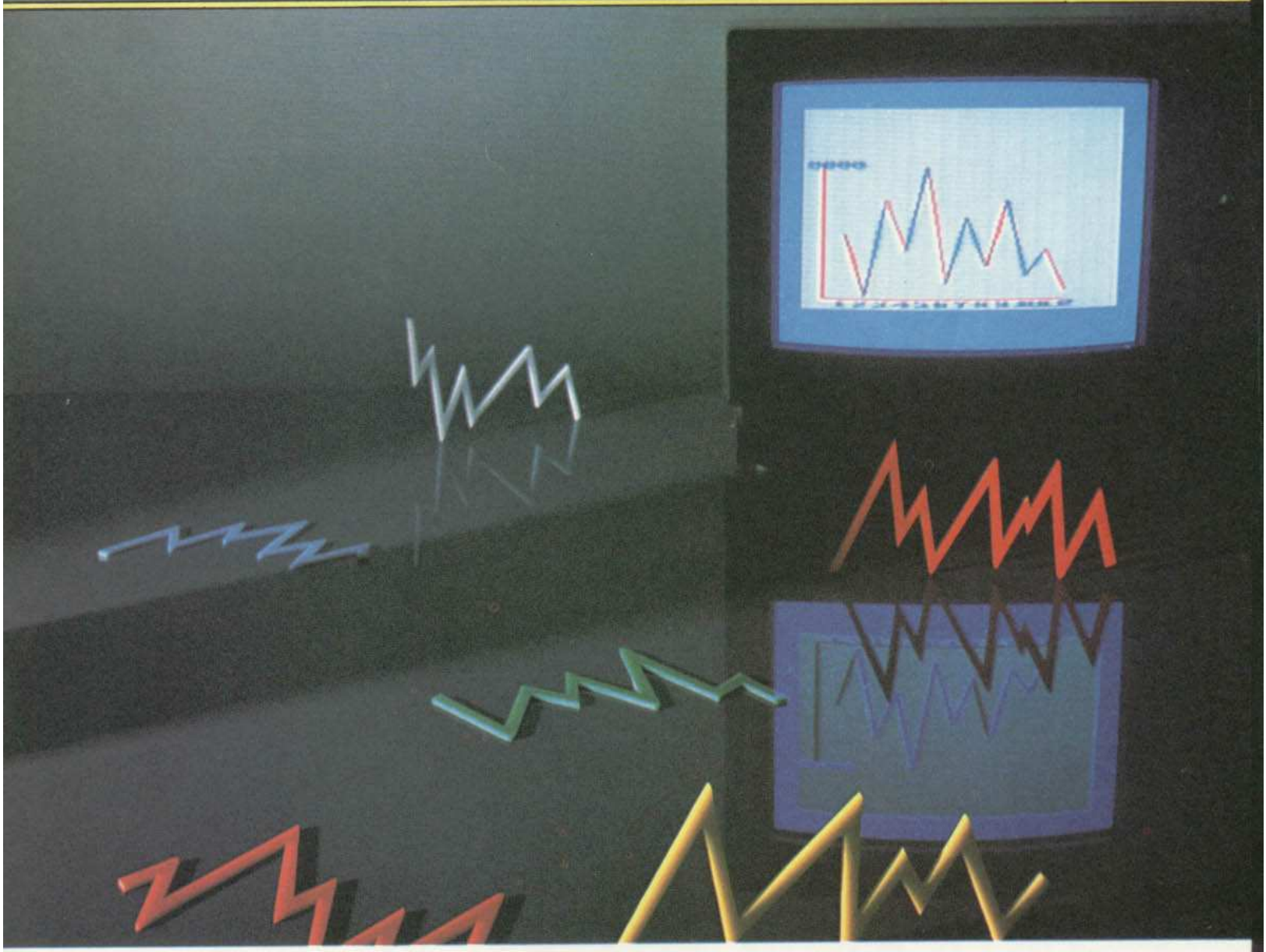
COMERCIAL
Diretor Comercial: Roberto Silveira
Gerente Comercial: Joaquim Celestino da Silva
Gerente de Circulação: Denise Mozol
Gerente de Propaganda e Publicidade: José Carlos Madio
Gerente de Pesquisa e Análise de Mercado: Wagner M. P. Nabuco de Araújo

(CLC)
A Editora Nova Cultural Ltda. é uma empresa do Grupo CLC — Comunicações, Lazer e Cultura
Presidente: Richard Civita
Diretoria: Flávio Barros Pinto, João Gomez, Menahen M. Politi, Renê C. X. Santos, Stélio Alves Campos

Edição organizada pela Editora Nova Cultural Ltda.
Av. Brigadeiro Faria Lima, 2000 - 3º andar
CEP 01452 - São Paulo - SP - Brasil
(Artigo 15 da Lei 5 988, de 14/12/1973).
Esta obra foi composta pela AM Produções Gráficas Ltda. e impressa pela Companhia Lithographica Ypiranga.

COMO TRAÇAR GRÁFICOS

- COMO TRAÇAR GRÁFICOS DE FUNÇÕES MATEMÁTICAS
- CURVAS SERRILHADAS
- OS EIXOS CARTESIANOS
- COMO DETERMINAR ESCALAS



Junte a capacidade de manusear dados com as funções gráficas do seu micro e desenvolva programas capazes de mostrar em um gráfico qualquer tipo de informação numérica.

Armazenamento e processamento de dados são atributos típicos de qualquer computador. Contudo, não é fácil para o usuário assimilar grandes quantidades de informações. Se nos apresentarem, por exemplo, uma lista contendo milhares de números, é muito difícil interpretar seu significado.

Para entender melhor, dê uma olhada na seguinte lista: 132.9, 146.2, 132.89, 123.92, 147.01, 153.47, 132.09, 138.79, 147.57, 153.9, 140.04, 142.76, 152.76, 132.6, 135.09, 146.98. Agora responda rapidamente às seguintes perguntas: qual é o maior número da lista? Qual a sua posição na lista? Quantos números menores que 135.5 há nela?

Como você vê, não é muito fácil dar respostas rápidas a essas questões. Imagine se a quantidade de números fosse cinco vezes maior!

A maneira tradicional de tratar esse tipo de problema consiste em colocar a informação em um gráfico. Quando isto é feito, torna-se fácil perceber qual é o ponto mais alto. Assim, com o auxílio de uma régua na posição horizontal, podemos verificar quais os valores que estão acima ou abaixo de um determinado valor.

No artigo *Reúna Seus Dados em Gráficos* (página 181), apresentamos um programa que mostra, sob a forma de um gráfico de barras, dados introduzidos no computador. O presente artigo trata das técnicas necessárias à confecção de gráficos lineares. Em lições posteriores, abordaremos os histogramas e outros tipos de gráfico.

Como as rotinas deste artigo vão além das capacidades gráficas do ZX-81 e do TRS-80, não apresentaremos programas para tais micros.

GRÁFICOS DE FUNÇÕES MATEMÁTICAS

A estrutura de um programa depende do tipo de informação que queremos mostrar na tela. Quando há uma relação matemática ligando os dados, podemos usar uma função matemática. Os microcomputadores são dotados, por exemplo, de funções trigonométricas — seno, cosseno e tangente — que tornam fácil o traçado de curvas cíclicas, como já foi explicado em artigos anteriores, ou como demonstra o seguinte programa:

```
30 PLOT 0,76
40 FOR t=0 TO PI*10 STEP .3
50 DRAW 2,COS t*15
60 NEXT t
```

```
10 COLOR 1,15,1
20 SCREEN 2
30 DRAW "BM0,95"
40 FOR T=0 TO 40*ATN(1) STEP .0
6
50 LINE -(8*T,95-40*SIN(T))
60 NEXT
150 GOTO 150
```

```
10  HOME : HGR : HCOLOR= 3
30  HPLOT 0,145
40  FOR T = 0 TO 279 STEP 4
50  HPLOT  TO T,95 + 50 *  COS
(T / 10)
60  NEXT T
150  END
```

```
15 PMODE 3,1
20 PCLS
25 SCREEN 1,1
30 DRAW"BM0,95"
40 FOR T=0 TO 40*ATN(1) STEP.06
50 LINE-(8*T,95-40*SIN(T)),PSET
60 NEXT
150 GOTO 150
```

Esse programa traça uma curva senóide, que tem muitas aplicações em ciência, tecnologia, música e traçados gráficos. A linha 40 estabelece que a curva terá cinco ciclos (menos no Apple). Cada ciclo é duas vezes PI. A linha 50 determina como é desenhada a curva (para conhecer maiores detalhes, veja o artigo *Mais Requinte em Seus Desenhos*,

à página 354). Alterando os valores nessas linhas (linha 50 no Apple), podemos variar consideravelmente a forma da curva.

O micro tem um número limitado de funções embutidas, prontas para serem usadas, mas qualquer função matemática pode ser desenhada, desde que a definamos dentro de um programa.

Acrescente as próximas linhas ao programa e execute-o novamente.

```
 70 PLOT 0,60
 80 FOR n=1 TO 220 STEP 5
 90 DRAW 5,((110-n)*15)/200
100 NEXT n
110 PLOT 0,100
120 FOR n=1 TO 220 STEP 5
130 DRAW 5,-((110-n)*15)/200
140 NEXT n
```

```
15 D=1:E=64
70 COLOR 4
80 DRAW"BM0,"+STR$(INT(109*D+E)
)
90 FOR T=-127 TO 128
100 LINE -(127+T,T*T/150*D+E)
110 NEXT
120 IF D=-1 THEN 150
130 D=-1:E=E+64:COLOR 6
140 GOTO 80
```

```
5 D = 1:E = 64
70  HCOLOR= 5
80  HPLOT 0,64 * D + E
90  FOR T =  - 139 TO 140
100  HPLOT  TO 139 + T,T * T /
160 * D + E
110  NEXT
120  IF D =  - 1 THEN 150
130 D =  - 1:E = E + 64: HCOLOR
= 6
140  GOTO 80
```

```
10 D=1:E=64
70 COLOR 3
80 DRAW"BM0,"+STR$(INT(109*D+E)
)
90 FOR T=-127 TO 128
100 LINE -(127+T,T*T/150*D+E),P
SET
110 NEXT
120 IF D=-1 THEN 150
130 D=-1:E=E+64:COLOR 2
140 GOTO 80
```

Temos agora duas parábolas, além da senóide. Os números necessários para sua elaboração são criados em um laço **FOR...NEXT**. O Spectrum, por exemplo, usa um laço para as parábolas e outro para a senóide.

GRÁFICOS IRREGULARES

A maioria dos gráficos, porém, está vinculada a números sem qualquer relação matemática entre si. Esses números podem representar, por exemplo, os valores da precipitação pluviométrica ao longo do ano ou a flutuação dos lucros e perdas de uma firma: esse tipo de dado varia imprevisivelmente. Se fôssemos traçar o gráfico a mão, começaríamos pelos eixos cartesianos. Assim, façamos com que esta seja a primeira parte do programa:

```
140 DRAW 0,175
150 PLOT 0,0
160 DRAW 255,0
```

```
5 COLOR 1,15,15
```

```
10 SCREEN 2
140 LINE (8,0)-(8,191)
160 LINE (8,191)-(255,191)
200 GOTO 200
```

```
10  HOME : HGR : HCOLOR= 3
140  HPLOT 0,0 TO 0,159
160  HPLOT 0,159 TO 279,159
200  END
```

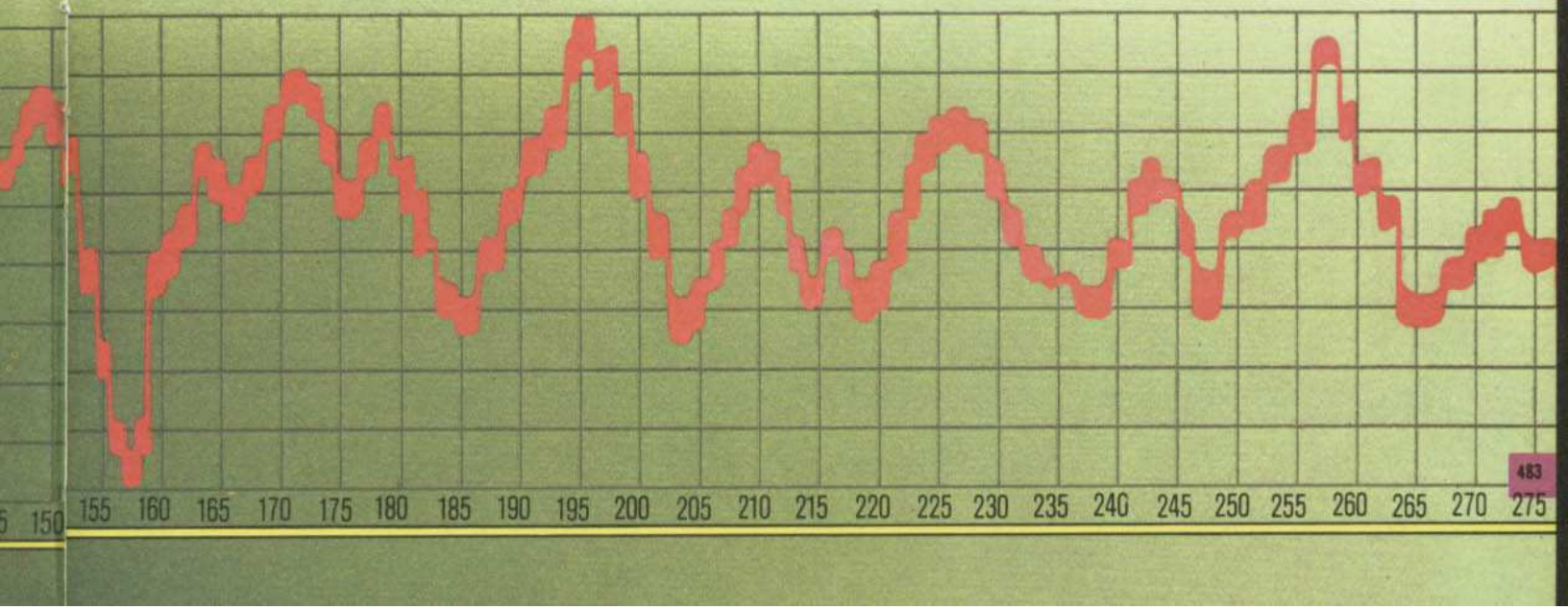
```
5 PMODE 3,1
10 PCLS
15 SCREEN 1,1
140 LINE (0,0)-(0,191),PSET
160 LINE (0,191)-(255,191),PSET
200 GOTO 200
```

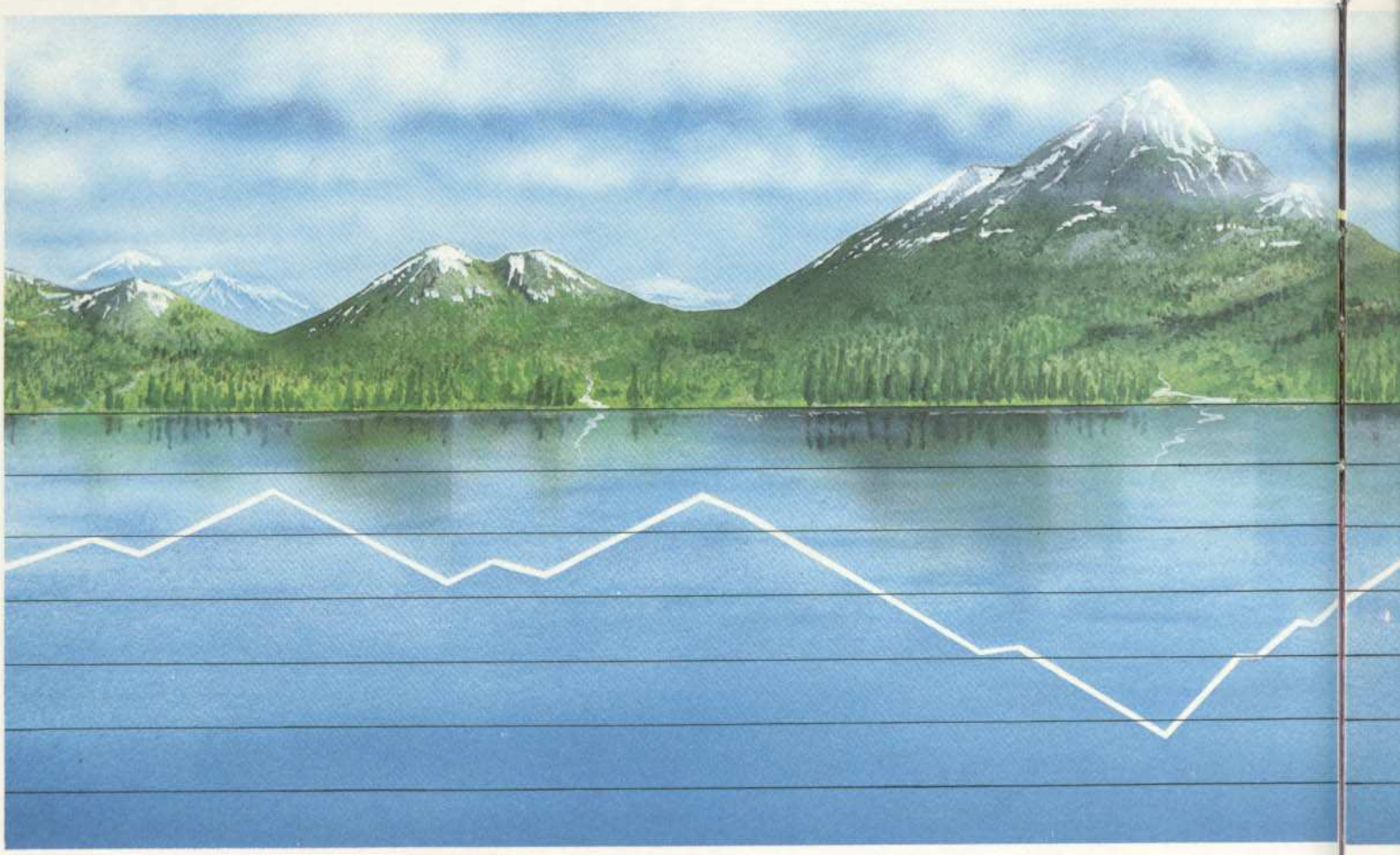
Nessas linhas, o programa seleciona um modo gráfico (menos no Spectrum, onde isso não é necessário), colocando o eixo Y no lado esquerdo da tela, e o eixo X, embaixo.

Esta é a disposição usual dos eixos, mas podemos modificá-la conforme o intervalo a que pertencem os números em questão. Por exemplo, se traçarmos curvas a respeito da variação de temperaturas ou sobre ganhos e perdas, teremos (em alguns casos) a ocorrência de números negativos; dessa forma, o eixo X terá que se deslocar para uma posição mais alta na tela.

Outra razão para mudar a disposição dos eixos é deixar espaço para números e legendas, sempre úteis quando se trata de gráficos. Faça as próximas modificações para ter um exemplo.

```
130 PLOT 20,10
140 DRAW 0,155
```





P&R

Posso modificar os programas de modo a misturar dados positivos e negativos em um só gráfico?

A rotina de entrada de dados não precisa ser modificada para aceitar números negativos; em contrapartida, as rotinas que cuidam dos eixos e dos gráficos exigem essa alteração. A maior dificuldade não é descobrir os valores máximo e mínimo, mas sim decidir em que ponto da tela colocar o eixo X.

Se você se dispuser a mudar a posição desse eixo a cada novo grupo de dados, as modificações serão bastante simples. Para isso, observe atentamente os dados, escolha fatores de escala adequados e comece a desenhar a partir da origem dos eixos.

```
150 PLOT 20,10
160 DRAW 235,0
```

```
5 COLOR 6,15,15
135 COLOR 1
140 LINE (20,0)-(20,160)
160 LINE (20,80)-(255,80)
```

```
10  HOME : HGR
135  HCOLOR= 3
140  HPLOT 20,0 TO 20,140
160  HPLOT 20,70 TO 279,70
200  END
```

```
5 PMODE 3,1
10 PCLS
15 SCREEN 1,1
135 COLOR 2
140 LINE (20,0)-(20,160),PSET
160 LINE (20,80)-(255,80),PSET
200 GOTO 200
```

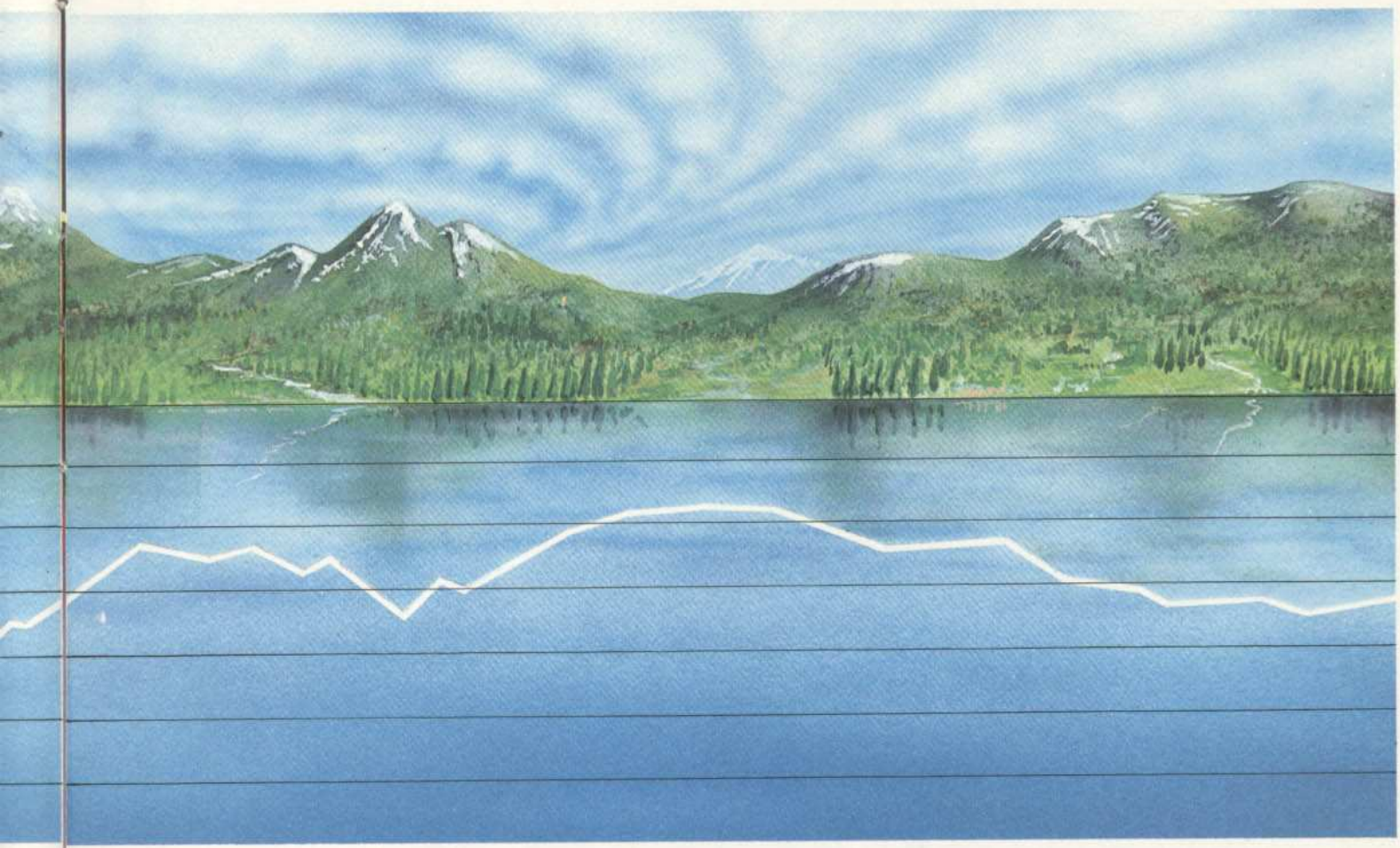
Ao executar o programa, notaremos que há margens à direita e abaixo do gráfico.

Para modificar o tamanho dessas margens, altere os valores nas linhas 130 a 160, com o cuidado de que, no Spectrum, as linhas 130 e 150 sejam idênticas.

Agora, acrescente as próximas linhas para continuar e acompanhe o desenvolvimento do processo.

```
  60 LET n=10
  70 PLOT 20,n
  80 FOR x=1 TO 12
  90 READ y
 100 DRAW 18,y-n
 105 LET n=y
 110 NEXT x
1000 DATA 50,70,60,100,80,120,1
00,130,70,140,90,110
```

```
70 DRAW "BM20,160"
80 FOR X=20 TO 20+11*20 STEP 20
90 READ Y
100 LINE-(X,161-Y)
110 NEXT
160 LINE (20,160)-(255,160)
1000 DATA 140,123,148,45,100,10
,20,8,45,8,45,30
```



```
15  HCOLOR= 5
70  HPLOT 20,160
80  FOR X = 20 TO 20 + 11 * 20
STEP 20
90  READ Y
100  HPLOT  TO X,141 - Y
110  NEXT
160  HPLOT 20,140 TO 279,140
1000  DATA  130,113,138,35,90,5
,20,8,45,8,45,30
```

```
70 DRAW"BM20,160"
80 FOR X=20 TO 20+11*20 STEP 20
90 READ Y
100 LINE -(X,Y),PSET
110 NEXT
160 LINE (20,160)-(255,160),PSE
T
1000 DATA 140,123,148,45,100,10
,20,8,45,8,45,30
```

O programa desenha doze pontos na tela, tomando os valores de X da linha 80, e os de Y da linha 1000; em seguida, ele une esses valores para formar um gráfico serrilhado.

O primeiro valor da linha **DATA** é desenhado na extrema esquerda, em cima do eixo dos Y, mas poderíamos tê-lo colocado na primeira posição dos X, modificando, assim, a localização inicial do cursor.

Em todos os programas, menos no do Spectrum, essa localização é especificada na linha 70; no Spectrum, isso é feito invariavelmente na linha 60.

Se os valores vão ser usados várias vezes, convém distribuí-los em uma linha **DATA**. Mas, se quisermos traçar gráficos com valores diferentes, é melhor usarmos o comando **INPUT**.

Acrescente então a próxima linha. Isso funciona em todos os gêneros de microcomputadores, à exceção dos que são compatíveis com o MSX, que não aceita o comando **INPUT** quando está no modo gráfico.

```
90 INPUT "Valor de Y ?",y:
LET y=y+10
```

```
90  VTAB 22: INPUT "VALOR DE Y
?";Y
```

```
90 INPUT "VALOR DE Y ";Y
130 SCREEN 1,1
```

Ao ser executado, o programa pede o primeiro valor de Y. Em seguida, ele realiza a primeira volta do laço da linha 80. Feito isso, ele torna a pedir um valor de Y, prosseguindo dessa maneira até que os doze valores sejam colocados na tela.

Muitos usuários não gostam da interrupção que a rotina causa no traçado do gráfico. As seguintes modificações resolvem o problema:

```
 20 DIM y(12)
 40 FOR n=1 TO 12
 50 INPUT "Valor de Y?",y(n):
 LET y(n)=y(n)+10
 60 NEXT n: CLS
 70 LET n=10
 80 PLOT 20,n
 90 FOR x=1 TO 12
100 DRAW 18,y(x)-n
105 LET n=y(x)
110 NEXT x
```

O que é escalamento?

Escalamento é uma transformação matemática dos eixos de um gráfico, de modo a fazer com que os seus extremos (pontos máximo e mínimo) caibam nos limites de uma "janela" no vídeo. Esta pode ocupar todo o vídeo, ou apenas parte dele; neste último caso, diversos gráficos podem ser distribuídos simultaneamente por diferentes pontos da tela.

Praticamente todos os gráficos elaborados por programas exigem algum tipo de escalamento. A única exceção são os gráficos cujas coordenadas horizontais e verticais se situam no domínio dos números inteiros positivos, e cujos valores máximos em cada eixo não excedem em número de pixels a resolução da tela na mesma direção.

Suponhamos, por exemplo, que é necessário colocar em gráfico duas variáveis, X e Y, onde X varia de um mínimo de 10 a um máximo de 200, e Y varia de um mínimo de 0 a um máximo de 150. Nesse caso, se o computador a ser usado for da linha Apple II, que tem uma resolução gráfica máxima de 280 pontos na horizontal por 192 pontos na vertical, não precisaremos escalar os eixos X e Y. Esse exemplo, porém, é uma exceção.

Existem ainda casos em que basta escalar apenas um dos eixos. Entretanto, na maioria das vezes, ambos os eixos precisam ser escalados.

Do ponto de vista matemático, o método de escalamento consiste apenas em uma regra de três. A mesma fórmula pode ser aplicada tanto para o eixo horizontal como para o vertical, bastando trocar as variáveis **x** por **y**. A fórmula leva em conta os valores mínimo e máximo do eixo, o número de pontos gráficos que cabem nele, e se os valores numéricos identificadores das marcas dos eixos (rótulos) serão arredondados ou não.

A fórmula de escalamento é:

$$x' = \frac{x - xmin}{xmax - xmin} \cdot pix$$

onde:

x' = valor escalado
x = valor original
xmin = valor mínimo da escala
xmax = valor máximo da escala
pix = número máximo de pixels do eixo.


```
5 COLOR 1,15,15
10 SCREEN 0
20 DIM Y(11)
40 FOR N=0 TO 11
50 INPUT "Qual o valor de Y ";Y
(N)
60 NEXT
65 COLOR 6:SCREEN 2
70 DRAW "BM20,"+STR$(INT(161-Y(
0)))
80 N=0
90 FOR X=20 TO 11*20+20 STEP 20
100 LINE -(X,161-Y(N))
110 N=N+1
120 NEXT
```

```
10  HOME
15  DIM Y(11)
40  FOR N = 0 TO 11
50  INPUT "QUAL O VALOR DE Y ?
";Y(N)
60  NEXT
65  HOME : HGR : HCOLOR= 5
70  HPLOT 20,141 - Y(0)
80 N = 0
90  FOR X = 20 TO 11 * 20 + 20
STEP 20
100  HPLOT  TO X,141 - Y(N)
110 N = N + 1
120  NEXT
```

```
20 DIM Y(11)
40 FOR N=0 TO 11
50 INPUT "VALOR DE Y ";Y(N)
60 NEXT
70 DRAW "BM20,"+STR$(INT(161-Y(
0)))
80 N=0
90 FOR X=20 TO 11*20+20 STEP 20
100 LINE-(X,Y(N)),PSET
110 N=N+1
120 NEXT
```

O programa realiza primeiro a rotina de entrada de dados para depois fazer o gráfico. Os dados são colocados na matriz Y ( ), como variáveis de Y (1) a Y (12) no Spectrum, e Y (0) a Y (11) nos demais computadores. Caso se queira utilizar mais de doze números, basta redimensionar a matriz. Os valores de X são obtidos na linha 90, e o desenho é feito pela linha 100. A linha 110 conta os valores de Y.

COMO USAR ESCALAS

Os valores para os gráficos foram escolhidos até agora de modo a caber dentro dos eixos e da tela. Geralmente, porém, os números que queremos colocar em um gráfico são maiores ou menores que o ideal. Precisamos, assim, de uma escala que os coloque dentro do intervalo válido. A maneira mais simples de fazer isso é olhar todos os valores e decidir então qual é o fator que deveremos usar para multiplicar todos os números, de forma a mostrá-los no gráfico.

A seguir, apresentamos o programa completo, isto é, com todas as modificações necessárias.

```
 20 DIM y(12)
 30 LET xs=2: LET ys=1/50
 40 FOR n=1 TO 12
 50 READ y(n)
 60 NEXT n: CLS
 70 LET n=8
 80 PLOT 40,n
 90 FOR x=1 TO 12
100 DRAW 8*xs,(y(x)-n)*ys
105 LET n=y(x)
110 NEXT x
130 PLOT 40,8
140 DRAW 0,167
150 PLOT 40,8
160 DRAW 210,0
1000 DATA 4000,2000,6000,3000,8
000,4000,5000,2000,6000,1500,30
00,500
```

```
5 COLOR 6,15,15
10 SCREEN 2
20 DIM Y(12)
30 XS=20:YS=1/20
40 FOR N=1 TO 12
50 READ Y(N)
60 NEXT
80 DRAW "BM20,"+STR$(INT(161-YS
*Y(1)))
90 FOR X=1 TO 12
100 LINE-(X*XS,161-YS*Y(X))
110 NEXT
135 COLOR 1
140 LINE (20,0)-(20,160)
160 LINE (20,160)-(255,160)
200 GOTO 200
1000 DATA 1600,1000,2500,3000,3
200,2000,2500,1000,3000,750,150
0,250
```

```
10  HOME : HGR : HCOLOR= 5
20  DIM Y(12)
30 XS = 20:YS = 1 / 20
40  FOR N = 1 TO 12
50  READ Y(N)
60  NEXT
80  HPLOT 20,141 - Y(1) * YS
90  FOR X = 1 TO 12
100  HPLOT  TO X * XS,141 - Y(X
) * YS
110  NEXT
135  HCOLOR= 3
140  HPLOT 20,0 TO 20,140
```

```
160  HPLOT 20,140 TO 279,140
200  END
1000  DATA  1600,1000,2500,300,
450,2000,2500,1000,2700,750,150
0,250
```

```
5 PMODE 3,1
10 PCLS
20 DIM Y(12)
30 XS=20:YS=1/20
40 FOR N=1 TO 12
50 READ Y(N)
60 NEXT
80 DRAW"BM20,"+STR$(INT(161-YS*
Y(1)))
90 FOR X=1 TO 12
100 LINE -(X*XS,Y(X)*YS),PSET
110 NEXT
130 SCREEN 1,1
135 COLOR 2
140 LINE (20,0)-(20,160),PSET
160 LINE (20,160)-(255,160),PSE
T
200 GOTO 200
1000 DATA 1600,1000,2500,3000,3
200,2000,2500,1000,3000,750,150
0,250
```

O programa faz o gráfico a partir dos valores da linha **DATA** 1000. Assim, poupa-se o tempo de digitação dos números enquanto o programa está sendo testado. Como vimos anteriormente, é possível mudar facilmente a via de entrada dos dados.

Os fatores de escala são estabelecidos pela linha 30. A linha 80 move o cursor para a posição inicial e os pontos são traçados pelo laço das linhas 90 a 110. As linhas 130 a 160 desenham os eixos cartesianos.

Se o gráfico não utilizar todo o espaço disponível na tela, podemos mudar o valor dos fatores de escala da linha 30. Para usar a escala nos eixos, digite as próximas linhas.

```
140 DRAW 0,2000*ys
160 DRAW 12*xs,0
```

```
140 LINE (20,160)-(20,161-3200*
YS)
```

```
140  HPLOT 20,140 TO 20,141 - 2
700 * YS
```

```
140 LINE (20,160)-(20,161-3200*
YS),PSET
```

Ao rodarmos o programa, o eixo Y irá somente até a altura do maior valor. O efeito será mais nítido se substituirmos os valores da linha 30. Agora podemos utilizar qualquer valor, desde que modifiquemos os fatores de escala na linha 30.

Para numerar os eixos, digite as próximas linhas:

```
210 FOR x=0 TO 12 STEP 2
220 PRINT AT 21,x*2+4;x
230 NEXT x
300 FOR y=0 TO 8000 STEP 2000
310 PRINT AT (8000-y)/400,0;y
320 NEXT y
```

As linhas 210 a 230 tomam os valores de 0 a 12, que são colocados logo abaixo do eixo X. Os números ao longo do eixo Y são impressos pelas linhas 300 a 320.

```
200 OPEN "GRP:" FOR OUTPUT AS #
1
210 FOR Y=500 TO 3000 STEP 500
220 PRESET (0,151-Y*YS):PRINT#1
,Y/500
230 LINE (18,161-Y*YS)-(20,161-
Y*YS)
240 NEXT
250 PRESET (25,0):PRINT #1,"(*
500)"
260 FOR X=20 TO 255 STEP 20
270 PRESET(X-8,170):PRINT#1,X/2
0
280 LINE (X,162)-(X,160)
290 NEXT
300 GOTO 300
```

A linha 200 abre um arquivo para que possamos escrever na tela de alta resolução. O laço das linhas 210 a 240 escreve os valores abaixo do eixo X e o laço das linhas 260 a 290, no eixo Y. Note como **PRESET** é usado para escolher a posição de impressão. A linha 250 informa a escala das ordenadas.

```
200  FOR Y = 500 TO 2500 STEP 5
00
210  HPLOT 18,Y * YS TO 20,Y *
YS
220  NEXT
230  FOR X = 20 TO 20 * 12 STEP
 20
240  HPLOT X,142 TO X,140
250  NEXT
260  END
```

Este programa apenas coloca tracinhos nos eixos. O Apple não tem uma maneira de escrever com facilidade na tela de alta resolução.

No TK-2000 podemos escrever os números na tela gráfica; neste caso, porém, a tarefa é deixada ao leitor.

O laço das linhas 200 a 220 cuida do eixo Y, enquanto que o das linhas 230 a 250 cuida do eixo X.

Escrever na tela gráfica do TRS-Color é mais complicado, pois não podemos usar o **PRINT**. Os caracteres devem ser desenhados com **DRAW**. Muito longa para ser listada aqui, a rotina que faz isso encontra-se na página 236 (*Recursos Gráficos Sofisticados*), e pode ser facilmente incorporada ao programa apenas com alguns ajustes nos números das linhas.

MICRODICAS

COMO COMPARAR DADOS

A capacidade de desenhar um gráfico em escala é particularmente útil quando se deseja mostrar mais de um conjunto de dados ao mesmo tempo. Por exemplo, as duas metades do ano. Se quisermos compará-las, devemos começar desenhando um gráfico para o primeiro semestre no alto da tela e um para o segundo, embaixo. Podemos colocar os doze valores dentro de uma só matriz. Depois, lemos os seis primeiros e acionamos a sub-rotina que faz o gráfico. Lemos a seguir os outros seis, e repetimos o processo. As mudanças necessárias dizem respeito à posição inicial de cada gráfico e à posição dos eixos, de forma a desenhar dois, e não apenas um eixo. Uma alternativa para essa solução é traçar um único eixo X no meio da tela, de onde partem, em direções diferentes, os dois gráficos.

A ARTE FINAL

Os últimos retoques no desenho ficarão por sua conta. Eles não são essenciais, mas o gráfico ficará mais atraente se for colorido e dispuser de um rodapé. Os comandos necessários para realizar esse trabalho podem ser revisados nos artigos *Como Desenhar em BASIC* (página 113) e *Ordem e Limpeza no Vídeo* (página 146).

CUIDADOS COM FITAS E DISCOS

- PROTEJA A INFORMAÇÃO
- COMO FAZER CÓPIAS CAUTELARES
- INDEXAÇÃO DO MATERIAL
- FAÇA ENVIOS PELO CORREIO

Você não precisa ser um gênio de organização para lidar com computadores. Mas vale a pena fazer um mínimo de esforço para manter em ordem suas fitas e disquetes.

Os sistemas de armazenamento auxiliar baseados em fitas ou discos oferecem ao usuário de computadores domésticos a possibilidade de contar com vastas quantidades de informação e também de milhares de programas de uma forma muito compacta.

Entretanto, a eficiência dos sistemas de armazenamento magnético tem alguns pontos fracos em potencial. Como as informações se concentram em pequenos disquetes ou em fitas cassetes, qualquer dano sofrido por estes pode ter efeitos desastrosos. Ao mesmo tempo, os meios magnéticos são bastante vulneráveis a agressões ambientais.

PROTEÇÃO CONTRA DANOS

Diante disso, é de fundamental importância que a informação permaneça inalterada no disco ou na fita em que foi gravada. Assim, você poderá tê-la de volta sempre que precisar.

Existem dois tipos de danos que podem afetar os meios magnéticos: o tipo mecânico e o magnético.

Os danos mecânicos podem ser causados por fatores externos, como calor, deformação (provocada pela pressão de um corpo mais pesado, por exemplo, poeira, umidade etc.). Esses fatores tendem a deteriorar a base magnética ou a estrutura dos disquetes e das fitas. Para evitar sua ação, é necessário proteger os meios magnéticos sempre que eles não estiverem sendo utilizados. Assim, mantenha os disquetes dentro de sua capa de cartolina e guarde-os em uma caixa ou gaveta apropriada. Conserve as fitas cassetes em suas caixas de plástico, e estas, por sua vez, em uma estante própria (o que facilitará também o trabalho de localizá-las). Armazene discos e fitas longe da poeira, do calor e da umidade.

Nunca toque a superfície exposta de um disco ou fita. Rebobine as fitas sempre que puder, colocando sua parte transparente (*leader*) na abertura, onde a fita está mais sujeita à ação dos agentes externos. Além disso, a fita ficará pronta para ser novamente utilizada. Não deixe fitas por muito tempo dentro do gravador com a tecla **PLAY** acionada; se o fizer, elas sofrerão deformações provocadas pelo cabeçote do gravador. Da mesma forma, nunca deixe disquetes dentro da unidade acionadora (*driver*).

Outro perigo considerável para seus discos e fitas são os campos magnéticos encontrados em aparelhos eletrodomésticos como alto-falantes e alguns tipos de motor. Equipados com ímãs, esses aparelhos geram campos magnéticos fortes mesmo quando não estão funcionando. Portanto, mantenha discos e fitas longe de televisores, motores de automóveis, geladeiras, aparelhos de ar condicionado, ventiladores, e até mesmo de telefones.

SEGURO CONTRA PERDAS

Por outro lado, os acidentes são inevitáveis; por isso, vale a pena tomar precauções extras e fazer duplicatas dos dados e programas mais importantes. Essas cópias cautelares, ou de segurança, são conhecidas em computês pelo nome de *back-up*.

Em geral, é uma boa idéia gravar tudo duas vezes (o que pode ser feito na mesma fita ou disco). Três vezes é melhor ainda. Desse modo, pelo menos uma versão permanecerá intacta. Se o arquivo for importante, faça cópias em novos discos ou fitas, e guarde-as em outro lugar; mas procure não utilizá-las: elas poderão ser necessárias para recuperar o arquivo original em caso de perda.

Proteja suas fitas contra a desgravação, removendo a orelha de plástico da borda da fita. Para proteger os disquetes, coloque uma fita adesiva opaca ou reflexiva sobre o furo retangular que existe em uma ou ambas as bordas (isso, nos disquetes de 5 1/4''. No caso dos disquetes de 8'', a proteção se faz removendo o adesivo).

Com o passar do tempo, discos e fitas se acumularão às centenas em seu arquivo, tornando cada vez mais difícil a localização dos dados e programas. Assim, é aconselhável rotular discos e fitas desde o começo (você pode, por exemplo, numerá-los para referências rápidas; alguns micros permitem identificar disquetes com nomes). Paralelamente, procure armazenar apenas arquivos relacionados em uma fita ou disco. As fitas devem, de preferência, ser curtas e conter de um a três arquivos ou programas, apenas.

Dê a cada arquivo um nome bem claro, dentro das limitações impostas pelo sistema operacional usado pelo seu micro. Se você tiver várias versões de um programa, utilize um sistema de numeração qualquer para identificá-las. Coloque o nome e o número de cada versão dentro dos programas utilizando declarações **REM**.

Para identificar discos e fitas, use etiquetas gomadas adequadas. No caso de fitas, não se esqueça de rotular também as caixas. Não escreva com lápis ou caneta esferográfica sobre a etiqueta de um disquete. Mantenha um caderno ou fichário com todos os nomes de arquivos, programas, comentários e suas respectivas localizações.

PELO CORREIO

Fitas e discos constituem um meio muito conveniente de enviar programas e outros tipos de informação pelo correio. As fitas cassetes são razoavelmente fortes e ''viajam'' bem. Envie-as sem a caixa para reduzir o peso e o custo (não esqueça de travar o movimento das bobinas). Envelopes almofadados ou a caixinha lacrada de fonogramas são ideais para esses envios.

No caso de discos, convém embalá-los entre dois pedaços de papelão rígido para evitar deformações. Se você preferir, pode também colocá-los em caixas metálicas rasas ou de plástico resistente. Em todos os casos, escreva do lado de fora do pacote: ''MEIO MAGNÉTICO. FRÁGIL''. Para maior segurança, envie registrado.

GERAÇÃO DE BLOCOS GRÁFICOS (1)

- CRIE NOVOS CARACTERES
- COMO COLOCAR PADRÕES EM UM BANCO DE MEMÓRIA
- TECLAS DE CONTROLE
- CORES EM ALTA RESOLUÇÃO

Todos os que já tentaram sabem como é árduo o trabalho de criar novos caracteres usando papel e lápis. Mas nem tudo está perdido: eis aqui um programa que facilita esse trabalho.

Conhecidos pela sigla UDG, os blocos gráficos são um dos recursos mais úteis ao programador gráfico. O trabalho de planejá-los e desenhá-los inicialmente no papel, entretanto, exige muito tempo e dedicação. Além disso, o cálculo dos números correspondentes a serem utilizados em linhas **DATA** é bastante complicado. Por outro lado, se algum erro for cometido, no decorrer do processo, sua correção também exigirá um esforço especial.

O programa apresentado neste artigo facilita as coisas. Ele substitui o papel milimetrado pela tela do micro, onde poderemos desenhar o padrão do novo bloco gráfico. O cálculo dos valores a serem colocados em linhas **DATA** também é feito automaticamente.

Outra grande vantagem do programa é que ele nos deixa ver, em tamanho natural, o bloco que estamos criando e nos poupa o trabalho de transferir os dados para outro programa. Várias teclas de controle nos permitirão não só modificar o padrão do desenho, como também produzir o padrão inverso, ou virá-lo da esquerda para a direita. Tudo isso com um simples toque.

Quando estivermos satisfeitos com o padrão criado, poderemos preservá-lo, gravando todo um conjunto de blocos. Uma vez gravado, esse conjunto poderá ser carregado para utilização em outros programas, ou reeditado pelo presente gerador de blocos gráficos.

O programa virá em duas partes. A primeira, neste artigo, cuida do aspecto básico do gerador. Faz um quadriculado na tela, permitindo a movimentação de um cursor para a criação do padrão desejado. Na parte seguinte, serão apresentadas rotinas que possibilitam modificações mais sutis no caractere.

Grave a primeira parte em fita ou disco para completá-la posteriormente. Devido à falta de alta resolução gráfica no ZX-81 e no TRS-80, não mostraremos versões para esses micros.

Colocado em funcionamento, o programa mostra na tela um quadriculado com 8 x 8 quadradinhos. Este é o "papel milimetrado" no qual iremos criar nosso novo bloco gráfico. Agora podemos escolher quais quadradinhos "acender" e quais "apagar" para criar um padrão. Para tanto dispomos de um quadradinho branco, que funciona como cursor. Desprovido de cor especial, este último é, na verdade, a versão mais brilhante — com atributo **BRIGHT** — do quadradinho que está abaixo dele. Assim, podemos ver, ao mesmo tempo, onde o cursor está e se esse quadradinho foi aceso ou apagado. O programa foi escrito de modo que as setas movimentem o cursor e o número 0, e acendam ou apaguem os quadradinhos.

Quem preferir outras teclas para movimentar o cursor deve mudar os caracteres dentro do **IF I$ = ...** nas linhas 6520 a 6555.

O mesmo método serve para adaptar o programa ao uso de joysticks. Quem dispuser de um desses periféricos deve consultar o artigo da página 348.

Aqueles que não quiserem recorrer às setas para mover o cursor terão de escolher as novas teclas com bastante cuidado, pois nenhuma das teclas de controle pode ser usada. Essa parte inicial do programa utiliza apenas três teclas de controle (explicaremos melhor adiante), mas a parte seguinte usa mais. De qualquer modo, as teclas C, I, M, P, R, S

e T estão proibidas.

Usando o teclado ou o joystick, poderemos movimentar o cursor dentro do quadriculado. Quando pressionarmos a tecla 0, no teclado (ou o botão do joystick, se forem feitas as modificações adequadas), veremos a cor do cursor mudar, indicando que o quadradinho onde ele está foi "aceso". Se alterarmos a posição do cursor, provocaremos uma mudança na cor do quadradinho. Dessa maneira, podemos definir o padrão de novos blocos gráficos.

O programa oferece várias opções, que podem ser feitas a qualquer momento da edição. Uma delas é a de guardar o bloco no banco de memória. Para tanto, basta pressionar S — uma das teclas de controle que mencionamos antes.

O programa procurará saber então em que lugar do banco de memória deverá guardar o bloco. O banco é representado pelas letras de A a U, mostradas na tela acima do quadriculado. Depois de termos teclado uma letra entre A e U, haverá uma pausa, enquanto os números correspondentes ao padrão do bloco são colocados no banco de memória, usando **POKE**. Logo em seguida, o novo bloco aparecerá na tela, ocupando seu lugar no banco.

Outra opção consiste em recuperar um bloco guardado e promover sua edição. Para essa opção usamos a tecla P; em seguida, o computador pergunta que bloco desejamos. Em resposta, devemos pressionar a tecla correspondente; o bloco desejado será então colocado no quadriculado.

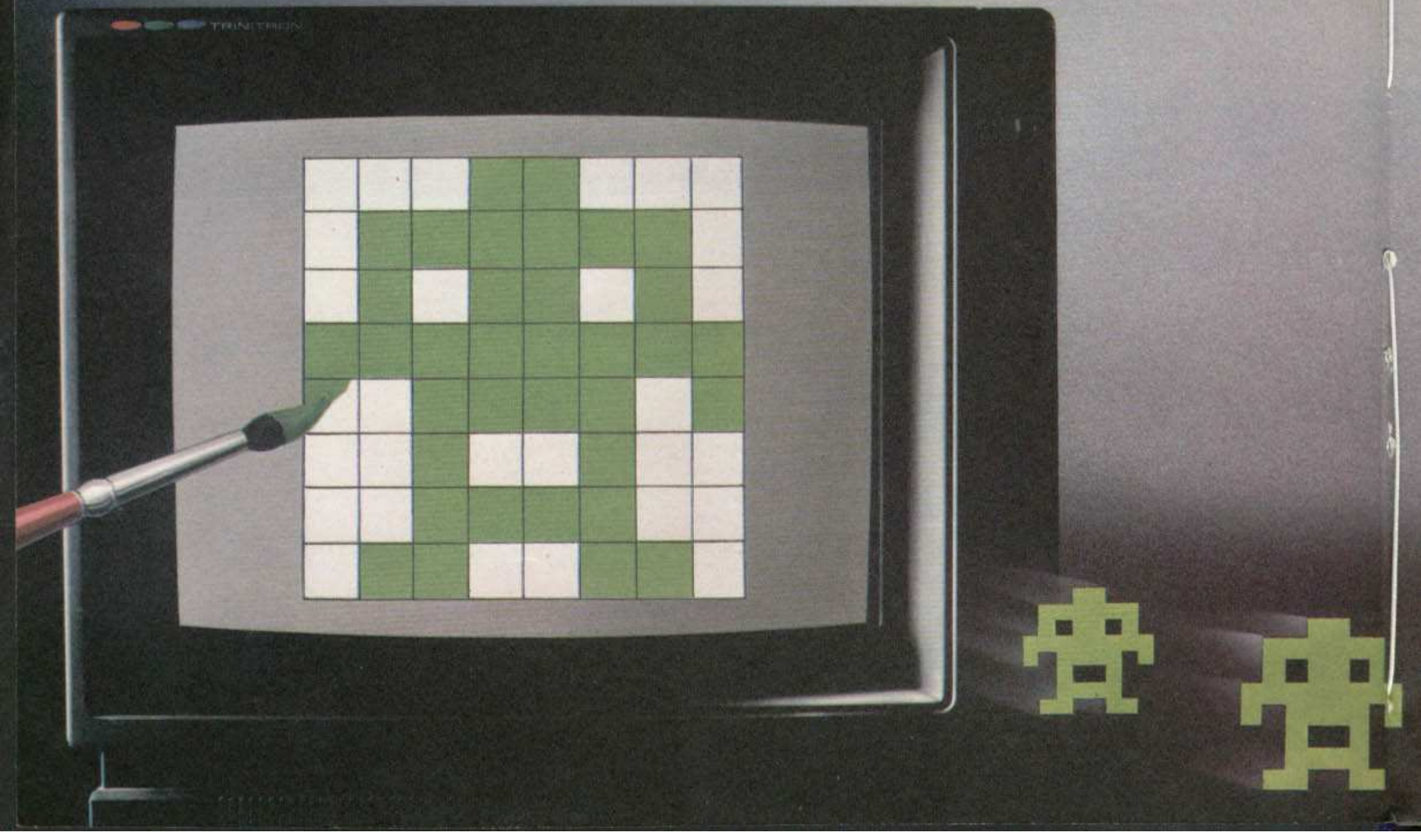
GRAVE BLOCOS NA FITA

Precisamos agora gravar o conjunto de blocos criados, de modo a utilizá-los em outros programas. Ao mesmo tempo, é aconselhável carregar um conjunto de blocos para edição.

Para começar, pressionamos a tecla T. O programa perguntará então se queremos gravar ou carregar um conjunto. No primeiro caso, todo o conteúdo do banco de memória será cuidadosamente gravado; portanto, não devemos esquecer de guardar ali o último caractere que estava sendo editado.

A próxima parte do artigo explicará como usar os recursos do gerador para criar e utilizar muitos e muitos blocos gráficos, incluindo como usar o conjunto criado em outros programas. Novos recursos interessantes serão também adicionados ao programa.

```
 5 CLEAR USR "A"-16
 6 POKE 23675,PEEK 23675-16
 8 BORDER 4: PAPER 7: INK 0:
 CLS
10 FOR N=USR "A" TO USR "B"+7
: READ A: POKE N,A: NEXT N
15 POKE 23675,PEEK 23675+16
 20 POKE 23658,8
 30 LET X=11: LET Y=8: LET NX=
 X: LET NY=Y
100 LET A$="": FOR N=144 TO
 164: LET A$=A$+CHR$ N: NEXT N
110 LET B$="": FOR N=65 TO 85:
 LET B$=B$+CHR$ N: NEXT N
1000 FOR N=87 TO 151 STEP 8: PL
OT N,48: DRAW 0,64: NEXT N
1010 FOR N=48 TO 112 STEP 8: PL
OT 87,N: DRAW 64,0: NEXT N
2000 PRINT AT 3,5;A$
2010 PRINT  INVERSE 1;AT 2,5;B$
2400 GOSUB 6500
2500 IF INKEY$="P" THEN  GOTO 5
000
2510 IF INKEY$="S" THEN  GOTO 5
100
2520 IF INKEY$="T" THEN  GOTO 5
200
3900 GOTO 2000
5000 INPUT "QUE CARACTER (A-U)?
", LINE C$
5010 IF C$<CHR$ 65 OR C$>CHR$ 8
5 THEN  GOTO 5000
5020 LET D=CODE C$+79: LET C$=C
HR$ D: GOSUB 6000: GOTO 2000
5100 INPUT "GUARDAR EM QUE LETR
A (A-U)?", LINE C$
5110 IF C$<CHR$ 65 OR C$>CHR$ 8
5 THEN  GOTO 5100
5120 PRINT AT 18,10;"GUARDANDO"
5130 FOR N=USR C$ TO USR C$+7
5140 LET R=0: LET BIT=128: FOR
M=0 TO 7
5150 IF PEEK (18432+(N-USR C$)*
32+M+11)<>1 THEN  LET R=R+BIT
5160 LET BIT=BIT/2: NEXT M: POK
```



```
E N,R
5170 NEXT N: PRINT AT 18,0;" ";
TAB 31;" ": GOTO 1000
5200 INPUT "C(ARREGAR) OU G(RAV
AR)?", LINE C$: IF C$<>"C" AND
C$<>"G" THEN  GOTO 1000
5220 IF C$="C" THEN  GOTO 5250
5230 INPUT "INTRODUZA O NOME DO
 ARQUIVO", LINE N$: IF N$="" OR
 LEN N$>10 THEN  GOTO 5230
5240 SAVE N$CODE USR "A",168: G
OTO 100
5250 INPUT "INTRODUZA O NOME DO
 ARQUIVO", LINE N$: IF LEN N$>1
0 THEN  GOTO 5250
5260 PRINT AT 19,0;: LOAD N$COD
E USR "A": PRINT AT 20,0;" ";TA
B 31;" ": GOTO 100
6000 LET B=USR "A"+8*(CODE C$-1
44)
6010 POKE 23675,PEEK 23675-16
6020 FOR N=0 TO 7
6030 LET V=PEEK (B+N)
6040 LET BIT=128: FOR M=0 TO 7
6050 IF V>=BIT THEN  PRINT AT 8
+N,11+M;CHR$ 144: LET V=V-BIT:
GOTO 6060
6055 PRINT AT 8+N,11+M;CHR$ 145
6060 LET BIT=BIT/2: NEXT M
6070 NEXT N: POKE 23675,PEEK 23
675+16: RETURN
6500 POKE 22528+32*Y+X,120: PAU
SE 0
6510 LET I$=INKEY$
6520 IF I$="5" AND X>11 THEN  L
ET NX=X-1
6530 IF I$="8" AND X<18 THEN  L
ET NX=X+1
6540 IF I$="6" AND Y<15 THEN  L
ET NY=Y+1
6550 IF I$="7" AND Y>8 THEN  LE
T NY=Y-1
6552 IF I$="0" THEN  GOSUB 7000
6555 IF I$="" THEN  GOTO 6580
6560 POKE 22528+Y*32+X,56
6570 LET X=NX: LET Y=NY
6580 POKE 22528+Y*32+X,120
6590 RETURN
7000 POKE 23675,PEEK 23675-16
7010 IF PEEK (18432+(Y-8)*32+X)
=1 THEN  PRINT  BRIGHT 1;AT Y,X
;CHR$ 144: GOTO 7030
7020 PRINT  BRIGHT 1;AT Y,X;CHR
$ 145
7030 POKE 23675,PEEK 23675+16:
RETURN
9000 DATA 85,171,85,171,85,171,
85,255,1,1,1,1,1,1,1,255
```

Quando o programa é executado, um quadriculado de 8 x 8 pontos surge na tela. Este será o "papel quadriculado" onde planejaremos a forma de nosso bloco gráfico.

Temos também um cursor em forma de cruz. Conforme as cores escolhidas, esse cursor será pouco ou muito visível, podendo até mesmo desaparecer quando sua cor for "transparente" (código de cor zero). O programa foi escrito de maneira que as setas movimentem o cursor e a barra de espaço acenda ou apague o ponto.

Se você quiser utilizar um joystick para mover o cursor (os pontos são acesos pelo botão de disparo), deve mudar o valor das instruções **STICK** e **STRIG**, nas linhas 70 e 20, respectivamente.

Assim, o cursor pode ser movimentado dentro do quadriculado. Quando pressionarmos a tecla de espaço, o ponto que está sob o cursor será aceso, assumindo a cor de frente; se pressionarmos a tecla novamente, ele será apagado, adotando a cor de fundo.

No início do programa, a cor de fundo é branca e a cor de frente, preta. Se quisermos modificar esta última, devemos pressionar a tecla C. O computador nos perguntará a seguir o código da nova cor de frente, ou seja, um número de 0 a 15 em decimal. A escolha dessa cor pode ser feita com apenas um toque; para isso, devemos digitar seu número em hexadecimal (de 0 a F). Já a cor de fundo pode ser modificada pressionando-se a tecla F.

É importante ter em mente as regras de apresentação das cores na tela de alta resolução do MSX. Cada conjunto de oito pontos adjacentes na horizontal, correspondentes a uma linha do nosso bloco gráfico, tem apenas uma cor de frente e uma cor de fundo. Se uma segunda cor de frente for atribuída a um ponto, todos os pontos acesos daquela linha do bloco passarão a ter essa cor. O mesmo acontecerá com os pontos apagados, se tentarmos introduzir uma segunda cor de fundo. Em nosso programa, esses cuidados ficam por sua conta; o computador pode mostrar até oito cores numa mesma linha de bloco *no quadriculado*. No momento em que esse bloco for colocado na tela, em tamanho natural, apenas as duas últimas cores — uma de frente e outra de fundo — aparecerão.

Essa limitação no número de cores pode causar inconvenientes. Se uma linha do bloco tiver a cor de frente igual à de fundo, o traçado do quadriculado naquela linha desaparecerá. Neste caso, de pouco valeria modificar o programa, fazendo-o redesenhar o quadriculado; isso, na verdade, criaria novos problemas. Na realidade, o traçado está ali: só não podemos vê-lo, porque ele tem a mesma cor do fundo.

O programa tem ainda outras funções, disponíveis a qualquer momento durante a edição de um bloco. Uma delas "guarda" o bloco em um banco de memória, que fica protegido na parte alta da memória. Para fazer isso, basta apertar a tecla G.

O programa solicita então um número correspondente à posição que o bloco ocupará no banco de memória. Esse número pode valer de 0 a 255. Uma vez informado da posição desejada, o computador mostra o bloco, em tamanho natural, no alto da tela. Ali fica representado todo o conteúdo do banco de memória. Um mesmo bloco pode ser guardado em várias posições diferentes.

Outra opção oferecida é a de "recuperar" um bloco que esteja no banco, trazendo-o de volta ao quadriculado, onde ele pode sofrer modificações. Isto é feito pela tecla R. Note que apenas o desenho do bloco é trazido para o quadriculado. O bloco permanece guardado no banco, e só será apagado quando outro for guardado na mesma posição.

COMO GRAVAR OS BLOCOS EM FITA

Após tanto trabalho, você certamente vai querer gravar os blocos criados para um eventual uso futuro. O programa dispõe, para tanto, de uma função de gravação em fita, acompanhada de outra, que permite a leitura de um banco previamente gravado.

Para gravar ou ler uma fita, é preciso pressionar a tecla T. O computador perguntará então se você deseja um **SAVE** ou um **LOAD**. Se você escolher gravar, o banco de memória será transferido em bloco para a fita. Portanto, não se esqueça de guardar o último bloco que estava sendo editado.

O banco de memória utilizado nada mais é do que uma região da memória do micro, protegida para esse fim. Ela contém, no entanto, alguns valores inúteis para nós (*lixo*, no jargão do programador). Esse "lixo" não é visível durante a execução do programa, mas pode ser transferido para a fita, quando não usamos todas as 256 posições. Ele torna-se visível quando carregamos um banco da fita.

Em geral, as posições múltiplas de 16 apresentam pontos aleatórios. Esse problema, porém, só afeta posições não ocupadas, nunca interferindo nos blocos. Da mesma forma, se interrompermos o programa por qualquer motivo, ao voltarmos a executá-lo, não veremos o banco na tela; mas todos os blocos criados estarão lá, podendo ser recuperados, especialmente quando soubermos suas posições.

Quando quisermos ler um conjunto de blocos da fita, o programa oferecerá a opção de ver seu conteúdo por meio da tecla V. Se soubermos exatamente onde estão os blocos, podemos poupar tempo e trabalho. Nesse caso, a tecla **<ENTER>** nos trará de volta à edição (não veremos nada na tela, mas todo o conteúdo do banco estará lá, podendo ser recuperado).

A próxima parte deste artigo explicará como utilizar estes e outros recursos do programa para criar blocos gráficos, bem como as formas de utilizá-los em outros programas.

```
10 CLEAR 200,&HD000:SCREEN1:FOR
 I=28*8 TO 28*8+7:A=VPEEK(BASE(
7)+I):A$=A$+CHR$(A):NEXT:COLOR
1,15,15:C=1:F=15:C1=C:C2=F:SCRE
EN 2,0
20 DIM B(8,8):SPRITE$(0)=A$:ON
STRIG GOSUB 500,500:STRIG(0) ON
30 GOSUB 1430
50 X=96:Y=64:GOTO 200
60 K$=INKEY$:IF K$="" THEN 60
70 A=STICK(0)
T(8,170):PRINT#1,"o número do b
loco :":CLOSE 1:GOSUB 9200
1010 X$=INKEY$:IF X$="" THEN 10
10
1015 IF X$=CHR$(13) THEN GOSUB
9100:GOTO 1050
1020 IF X$=CHR$(29) AND N>0 THE
N N=N-1:GOSUB 9200:GOTO 1010
1025 IF X$=CHR$(28) AND N<255 T
HEN N=N+1:GOSUB 9200:GOTO 1010
1030 IF X$=CHR$(31) AND N>9 THE
N N=N-10:GOSUB 9200:GOTO 1010
1040 IF X$=CHR$(30) AND N<245 T
HEN N=N+10:GOSUB 9200:GOTO 1010
1045 GOTO 1010
1050 FOR I=0 TO 7:P(I)=PEEK(&HD
100+N*8+I)
1052 FOR I=0 TO 7:P(I)=PEEK(&HD
100+N*8+I)
1055 C(I)=PEEK(&HE100+N*8+I)
1060 FOR J=7 TO 0 STEP -1
1070 B(I,J)=P(I)-INT(P(I)/2)*2:
P(I)=INT(P(I)/2)
1080 IF B(I,J)=0 THEN LINE (96+
8*J+1,64+8*I+1)-(96+8*J+6,64+8*
I+7),C(I)AND15,BF  ELSE LINE (9
6+8*J+1,64+8*I+1)-(96+8*J+6,64+
8*I+7),(C(I)-(C(I)AND15))/16,BF
1090 NEXT J,I:RETURN
1430 FOR I=96 TO 160 STEP 8:LIN
E (I,64)-(I,128),1:NEXT
1440 FOR I=64 TO 128 STEP 8:LIN
E (96,I)-(160,I),1:NEXT
1450 RETURN
1500 N=0:GOSUB 9000:PRINT#1,"Us
e as setas para escolher":PRESE
80 ON A GOTO 90,130,100,130,110
,130,120,130
85 GOTO 130
90 IF Y>64 THEN Y=Y-8:GOTO 200
95 GOTO 200
100 IF X<152 THEN X=X+8:GOTO 20
0
105 GOTO 200
110 IF Y<120 THEN Y=Y+8:GOTO 20
0
115 GOTO 200
120 IF X>96 THEN X=X-8:GOTO 200
125 GOTO 200
130 IF K$="G" THEN GOSUB 1500:G
OTO 200
140 IF K$="R" THEN GOSUB 1000:G
OTO 200
150 IF K$="T" THEN GOSUB 2000:G
OTO 200
160 IF K$="C" THEN GOSUB 2500:G
OTO 200
170 IF K$="F" THEN GOSUB 2600:G
OTO 200
200 IF C+1>15 OR C+1=F THEN S=C
 ELSE S=C+1
210 PUT SPRITE 0,(X,Y),S
220 GOTO 60
500 SWAP C1,C2:LINE (X+1,Y+1)-(
X+6,Y+7),C2,BF
510 IF C2=F THEN B((Y-64)/8,(X-
96)/8)=0 ELSE B((Y-64)/8,(X-96)
/8)=1
520 C((Y-64)/8)=C*16+F
530 RETURN
1000 N=0:GOSUB 9000:PRINT#1,"Us
e as setas para escolher":PRESE
```

```
T(8,170):PRINT#1,"o número do b
loco :":CLOSE 1:GOSUB 9200
1510 X$=INKEY$:IF X$="" THEN 15
10
1515 IF X$=CHR$(13) THEN GOSUB
9100:GOTO 1550
1520 IF X$=CHR$(29) AND N>0 THE
N N=N-1:GOSUB 9200:GOTO 1510
1525 IF X$=CHR$(28) AND N<255 T
HEN N=N+1:GOSUB 9200:GOTO 1510
1530 IF X$=CHR$(31) AND N>9 THE
N N=N-10:GOSUB 9200:GOTO 1510
1540 IF X$=CHR$(30) AND N<245 T
HEN N=N+10:GOSUB 9200:GOTO 1510
1545 GOTO 1510
1550 FOR I=0 TO 7:P(I)=0:FOR J=
0 TO 7
1560 P(I)=P(I)+B(I,J)*2^(7-J)
1570 NEXT J
1580 POKE &HD100+N*8+I,P(I)
1585 POKE &HE100+N*8+I,C(I)
1590 VPOKE BASE(12)+N*8+I,P(I)
1600 VPOKE BASE(11)+N*8+I,C(I)
1610 NEXT I:RETURN
2000 GOSUB 9000:PRINT#1,"(S)AVE
 ou (L)OAD ?"
2010 X$=INKEY$:IF X$<>"S" AND X
$<>"L" THEN 2010
2020 GOSUB 9100:IF X$="L" THEN
2100
2030 GOSUB 9000:PRINT#1,"Posici
one o gravador":PRESET(8,170):P
RINT#1,"e aperte <ENTER>"
2035 IF INKEY$<>CHR$(13) THEN 2
035
2040 BSAVE "CAS:UDG",&HD100,&HF
37F
2050 GOSUB 9100:RETURN
2100 GOSUB 9000:PRINT#1,"Posici
one o gravador":PRESET(8,170):P
RINT#1,"e aperte <ENTER>"
2105 IF INKEY$<>CHR$(13) THEN 2
105
2110 BLOAD "CAS:"
2120 GOSUB 9100
2130 GOSUB 9000:PRINT#1,"(V) pa
ra ver o banco":PRESET(8,170):P
RINT#1,"(ENTER) para voltar à e
dição"
2140 X$=INKEY$:IF X$="" THEN 21
40
2150 IF X$=CHR$(13) THEN GOSUB
9100:RETURN
2160 IF X$<>"V" THEN GOTO 2140
2170 FOR N=0 TO 255
2180 FOR J=0 TO 7
2190 VPOKE BASE(12)+N*8+J,PEEK(
&HD100+N*8+J)
2200 VPOKE BASE :1)+N*8+J,PEEK(
&HE100+N*8+J)
2220 NEXT J,N
2230 GOSUB 9100:RETURN
2500 GOSUB 9000:PRINT#1,"Número
 da cor de frente (0-F)"
2510 X$=INKEY$:IF X$="" THEN 25
10
2515 IF X$<"0" OR (X$>"9"AND X$
<"A") OR X$>"F" THEN 2510
2520 C=VAL("&H"+X$):C1=C:C2=F
2530 GOSUB 9100:RETURN
2600 GOSUB 9000:PRINT#1,"Número
 da cor de fundo (0-F)"
2610 X$=INKEY$:IF X$="" THEN 26
10
2615 IF X$<"0" OR (X$>"9"AND X$
<"A") OR X$>"F" THEN 2610
2620 F=VAL("&H"+X$):C1=C:C2=F
2630 GOSUB 9100:RETURN
9000 OPEN "GRP:" FOR OUTPUT AS
#1:PRESET(8,160):COLOR 1,15:RET
URN
9100 CLOSE 1:LINE (0,160)-(255,
191),15,BF:RETURN
9200 LINE (190,168)-(255,191),1
5,BF:GOSUB 9000:PRESET(190,168)
:PRINT#1,N;:CLOSE1:RETURN
```

Até agora, sempre que precisávamos de um desenho em alta resolução, usávamos o comando **DRAW**. Este é muito bom para figuras móveis, devido à velocidade de operação. Ele apresenta, contudo, dois inconvenientes: a extensão dos dados, já que as regras de definição das figuras são bem complicadas, e a dificuldade na utilização de cores, pois as figuras ficam deformadas quando coloridas.

O Apple e o TK-2000 também podem mostrar na tela caracteres definidos pelo usuário, usando regras bem semelhantes às dos demais micros. Tais regras devem ser empregadas quando for importante a cor, mas não a velocidade. Convém, nesse caso, consultar as seções correspondentes, dedicadas a outros computadores.

Quando executamos o programa, vemos um quadriculado de 8 x 8 pontos na tela. Este será o "papel quadriculado" onde desenharemos nosso bloco gráfico.

Temos também um cursor, na forma de um ponto. Para movimentá-lo, utilizamos as teclas Z, X, P e L, como de costume. Usando a barra de espaço podemos "acender" o ponto correspondente, apagando-o com a tecla A.

Para obtermos caracteres coloridos, precisamos entender como o Apple e o TK-2000 representam cores na tela. Embora nosso quadriculado tenha oito pontos de largura (ou seja, na horizontal), na tela aparecerão apenas os sete primeiros (a partir da esquerda). São, portanto, quarenta posições com sete pontos cada, resultando nos 280 pontos da tela de alta resolução. Se um ponto da linha estiver apagado, aparecerá em preto. Caso esteja aceso, contudo, sua cor vai depender de sua posição *na tela*. Se o ponto ocupar uma posição par, será violeta. Se ocupar uma posição ímpar, será verde. Tais regras se aplicam quando o oitavo ponto, que fica na extrema direita do bloco e não aparece na tela, estiver apagado. Se o oitavo ponto estiver aceso, as cores violeta e verde serão substituídas por azul e vermelho, respectivamente. Se dois pontos adjacentes estiverem acesos, assumirão a cor branca.

Existem, assim, seis cores disponíveis no modo de alta resolução. Sua exibição, porém, está sujeita às seguintes limitações:

1. Ponto em coluna ímpar é preto, violeta ou azul.

2. Ponto em coluna par é preto, verde ou vermelho.

3. Cada linha de bloco pode ter as cores violeta e verde ou azul e vermelha. Não é possível misturar cores de grupos diferentes na mesma linha.

4. Dois pontos adjacentes e acesos ficarão brancos, mesmo que estejam em linhas diferentes.

Deve-se ressaltar ainda que as cores dependem da posição do ponto *na tela, e não no bloco gráfico*. Isto quer dizer que a cor do ponto varia com a sua coordenada horizontal — de 0 a 279. Como a largura de cada linha de bloco na tela é de sete pontos, quando planejamos um bloco não sabemos quais os pontos que irão ocupar posições pares ou ímpares. Uma vez que tenhamos desenhado o nosso bloco, podemos guardá-lo no banco de memória do programa. Para isso, basta apertar a tecla G. O computador pergunta então em qual posição do banco vamos colocar o caractere — 0 a 300. Informado esse número, o caractere é mostrado na tela, em posição correspondente à do banco e em tamanho natural. Sua cor dependerá da posição no banco. Para facilitar o trabalho, é conveniente armazenar os blocos nas posições pares, onde é possível planejar as cores, pois as colunas da tela coincidem com as do bloco. Nas posições ímpares, as cores são de grupos invertidos, já que as posições pares e ímpares são invertidas.

O banco nada mais é que uma porção protegida de memória. Ele pode ocupar a página 2 de gráficos, tornando o trabalho mais simples. Essa página, porém, não deve ser usada enquanto o programa estiver no micro.

O uso da tecla R permite recuperar um bloco gravado no banco, trazendo seu desenho para o quadriculado. O bloco recuperado, por sua vez, permanece no banco. Um mesmo bloco pode ser guardado em diferentes posições.

COMO GRAVAR EM DISCO E FITA

Aqueles que dispõem de unidade de disco podem usar a tecla T para gravar ou carregar todo o conteúdo do banco

de memória. Feito isso, o computador perguntará qual a opção desejada. Se você selecionar a primeira opção, o conteúdo do banco será gravado. Não se esqueça, portanto, de guardar o último bloco editado.

Quando carregamos um banco, as posições que não contêm blocos são preenchidas por linhas brancas.

Quem não dispõe de disco, ou tem um TK-2000, deve parar o programa — com <**RESET**> ou <**CTRLX**><**C**> — para poder gravar o banco de memória em fita. Para tanto, é preciso usar o monitor, digitando **CALL −151** — ou **LM** no TK-2000. A seguir, digite (no Apple):

```
4000.5FFF W
```

No caso do TK-2000, temos algumas modificações:

```
A000.BFFF W "nome"
```

Para ler a fita, troque a letra **W** por **R**.

Depois de ter carregado o banco, execute novamente o programa. O conteúdo do banco poderá ser visualizado por meio da tecla T (caso tenham sido feitas as modificações sugeridas).

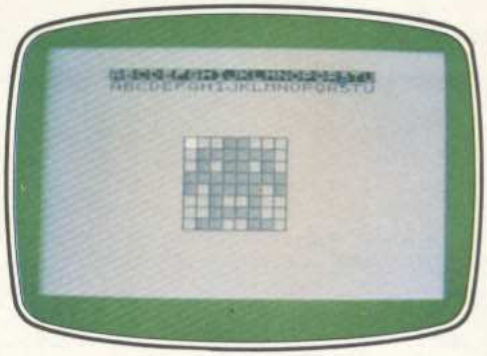
Na próxima parte do artigo, traremos mais informações sobre a criação e utilização de blocos gráficos no Apple. Além disso, novas funções serão acrescentadas ao gerador de blocos.

```
10  HOME : HGR : HCOLOR= 3: DIM
 B(8,8)
20 E = 16384:T = 8192
30  GOSUB 1030
50 X = 108:Y = 72: GOTO 200
60  GET K$:LX = X:LY = Y
70  IF K$ = "Z" AND X > 108 THE
N X = X - 8: GOTO 200
80  IF K$ = "X" AND X < 164 THE
N X = X + 8: GOTO 200
90  IF K$ = "P" AND Y > 72 THEN
 Y = Y - 7: GOTO 200
100  IF K$ = "L" AND Y < 121 TH
EN Y = Y + 7: GOTO 200
110  IF K$ = " " THEN B((Y - 72
) / 7,(X - 108) / 8) = 1: HCOLO
R= 2: GOSUB 500: GOTO 200
120  IF K$ = "A" THEN B((Y - 72
) / 7,(X - 108) / 8) = 0: HCOLO
R= 0: GOSUB 500: GOTO 200
130  IF K$ = "G" THEN  GOSUB 15
00: GOTO 50
140  IF K$ = "R" THEN  GOSUB 10
00: GOTO 50
150  IF K$ = "T" THEN  GOSUB 20
00: GOTO 50
200  HCOLOR= 0: HPLOT LX + 3,LY
 + 4 TO LX + 4,LY + 3: HPLOT LX
 + 4,LY + 4 TO LX + 3,LY + 3
210  HCOLOR= 5: HPLOT X + 3,Y +
 4 TO X + 4,Y + 3: HPLOT X + 4,
Y + 4 TO X + 3,Y + 3
230  GOTO 60
500  FOR K = X + 1 TO X + 6: HP
LOT K,Y + 1 TO K,Y + 6: NEXT :
RETURN
1000  HOME : VTAB 22: INPUT "QU
AL O NUMERO DO BLOCO ?";N: IF N
 > 320 THEN 1000
1005  FOR I = 0 TO 7:P(I) =  PE
EK (E + N * 8 + I)
1010  FOR J = 0 TO 7
1015 B(I,J) = P(I) -  INT (P(I)
 / 2) * 2:P(I) =  INT (P(I) / 2
)
1020  HCOLOR= 2 * B(I,J): FOR K
 = 109 + 8 * J TO 114 + 8 * J:
HPLOT K,73 + 7 * I TO K,78 + 7
* I: NEXT K,J,I
1025  HOME : RETURN
1030  HCOLOR= 3
1040  FOR I = 108 TO 172 STEP 8
: HPLOT I,72 TO I,127: NEXT
1050  FOR I = 72 TO 128 STEP 7:
 HPLOT 108,I TO 172,I: NEXT
1060  RETURN
1500  HOME : VTAB 22: INPUT "QU
AL O NUMERO DO BLOCO ?";N: IF N
 > 320 THEN 1500
1502 L =  INT (N / 40):C = N -
L * 40
1505  FOR I = 0 TO 7:P(I) = 0:
FOR J = 0 TO 7
1510 P(I) = P(I) + B(I,J) * 2 ^
 J
1520  NEXT J: POKE E + N * 8 +
I,P(I)
1525  POKE T + L * 128 + C + 10
24 * I,P(I)
1530  NEXT I: HOME : RETURN
2000  HOME : VTAB 22: INPUT "(S
)AVE OU (L)OAD ?";A$: IF A$ <
> "S" AND A$ <  > "L" THEN 2000
2010  INPUT "NOME DO ARQUIVO ?"
;AR$
2020  IF A$ = "L" THEN 2100
2030  PRINT : PRINT  CHR$ (4);"
BSAVE ";AR$;",A";E;",L3000"
2040  HOME : RETURN
2100  PRINT : PRINT  CHR$ (4);"
BLOAD ";AR$
2110  FOR N = 0 TO 320
2120 L =  INT (N / 40):C = N -
L * 40
2130  FOR I = 0 TO 7
2140  POKE T + L * 128 + C + 10
24 * I, PEEK (E + N * 8 + I)
2150  NEXT I,N
2160  HOME : RETURN
```

Aqueles que não dispõem de unidade de disco não devem copiar as linhas 2000 a 2100. Uma instrução **REM** deve ser colocada na linha 2000.

Os usuários do TK-2000 *não* devem copiar as linhas 2000 a 2100, fazendo ainda as seguintes modificações:

```
20 E = 40960:T = 8192
```



No Spectrum podem-se criar novos blocos...

Quando colocamos o programa para funcionar, ele nos faz duas perguntas iniciais, relacionadas com o tipo de bloco gráfico que queremos criar.

A primeira se refere ao **PMODE** desejado. **PMODE4** nos dá uma grande resolução em preto e branco ou em preto e verde apenas. A segunda opção, **PMODE3**, tem uma resolução três vezes mais baixa na horizontal — pontos com o triplo de largura —, mas permite que usemos quatro cores.

A segunda pergunta diz respeito ao conjunto de cores que vamos utilizar. **SCREEN0** no **PMODE4** nos dá preto e verde, enquanto em **PMODE3** nos dá vermelho, azul, verde e amarelo. **SCREEN1** nos dá preto e branco em **PMODE4** e ciano, magenta, laranja e branco em **PMODE3**.

Feitas estas opções, não poderemos mais mudar o **PMODE** sem usar **RUN** novamente, embora possamos mudar o **SCREEN**.

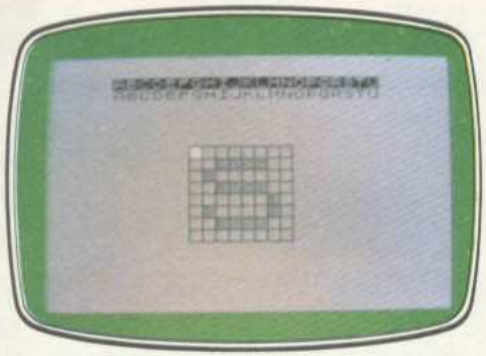
A seguir, o computador coloca na tela o quadriculado, que é a base para o desenho do bloco gráfico. Podemos então usar as setas para mover o cursor e assim desenhar o bloco.

Com o cursor sobre um determinado ponto, devemos pressionar <**ENTER**> para acender esse ponto. Para apagar pontos, basta mudar sua cor, como veremos mais adiante.

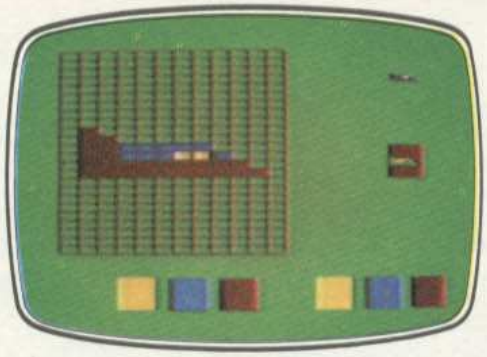
Dessa forma, desenharemos nosso bloco rápida e facilmente. A porção seguinte do programa terá uma opção de uso de joystick.

Como podemos observar, o quadriculado não é o único quadrado na tela: abaixo dele há mais oito e à sua direita, mais dois.

Os dois da direita são versões atualizadas do bloco criado, em modo normal e invertido. À medida que desenhamos, esses dois "modelos" vão mudando, de maneira a possibilitar a visualização do resultado em tamanho real.



...ou usar os caracteres do próprio micro.



Blocos mais largos no TRS-Color.

Os oito quadrados na parte inferior da tela servem para guardar os blocos que já foram criados. Isso significa que até oito blocos podem ser guardados ao mesmo tempo no banco de memória do programa. Alguns desses blocos não são visíveis, uma vez que têm pontos com a mesma cor do pano de fundo. Eles não mudam junto com a edição, mas sim quando guardamos no banco um caractere recém-criado.

AS TECLAS DE CONTROLE

O programa tem uma série de teclas de controle, que nos possibilitam fazer opções.

Duas delas são fundamentais: as teclas S e G. S guarda no banco de memória o bloco que foi editado. Quando pressionamos essa tecla, o programa nos pede um número de 1 a 8, que corresponde à posição do bloco no banco. Isso permite que substituamos qualquer um dos blocos do banco.

Já a tecla G, seguida do número correspondente, faz a operação inversa — recupera um bloco do banco e o coloca no quadriculado. Embora o programa utilize uma rotina em linguagem de máquina, o processo leva alguns segundos.

Recuperar um bloco não modifica o conteúdo da posição correspondente no banco. Assim, podemos recuperar um determinado bloco do banco, trazendo-o para o quadriculado quantas vezes quisermos, desde que não guardemos nada na mesma posição do banco.

Essa parte do programa utiliza duas outras teclas de controle. Se pressionarmos C, mudaremos a cor. Se estivermos em um modo com quatro cores, podemos usar qualquer uma delas. No modo de duas cores, escolhemos uma ou outra. Selecionada a cor adequada, podemos apagar qualquer ponto, corrigindo, assim, eventuais erros cometidos. Após digiar C, devemos informar o número da cor desejada.

A última tecla de controle dessa parte do programa é T, que permite a gravação ou a recuperação de todo um banco de memória contendo oito blocos.

```
10 CLEAR 200,30998:DEFUSR0=3100
0
20 T=0:FOR K=0 TO 43:READ N:T=T
+N:POKE K+31000,N:NEXT:READ C:I
F T<>C THEN PRINT "ERRO NOS DAD
OS":END
50 DATA 189,179,237,31,3,142,12
3,12,230,192,134,8,183,121,68,7
9,88,73,122,121,68,125,121,23
60 DATA 39,5,88,73,122,121,68,1
67,128,125,121,68,38,233,140,12
5,76,38,221,57,4725
110 CLS:PRINT"SELECIONE MODO GR
AFICO (3-4)";
120 A$=INKEY$:IF A$<"3" OR A$>"
4" THEN 120
130 T=5-VAL(A$):PRINT A$
140 PRINT"SELECIONE TIPO DE TEL
A(0-1)";
150 A$=INKEY$:IF A$<"0" OR A$>"
1" THEN 150
160 PRINT A$:ST=VAL(A$)
200 PMODE 5-T,1
210 DIM A(14),C1(1),C2(1),C3(1)
,C4(1)
220 PCLS 4:GET(0,0)-(9,5),C4,G
230 PCLS 3:GET(0,0)-(9,5),C3,G
240 PCLS 2:GET(0,0)-(9,5),C2,G
250 PCLS 1:GET(0,0)-(9,5),C1,G
260 PCLS:SCREEN 1,ST
270 C=-1:F=3*T-2:GOSUB 2070
280 FOR X=8 TO 255 STEP 32:C=(C
+1) AND 3:COLOR C+1:LINE(X,165)
-(X+23,188),PSET,BF:NEXT:COLOR
6-T
290 FOR X=3 TO 147 STEP T*6:LIN
E (X,3)-(X,147),PSET:NEXT
300 FOR Y=3 TO 147 STEP 6:LINE
(3,Y)-(147,Y),PSET:NEXT
310 X=12:Y=12
320 X1=X*6+4:Y1=Y*6+4
330 PUT(X1,Y1)-(X1+5*T-1,Y1+4),
C1,NOT
340 A$=INKEY$
350 GOSUB 1500
360 IF A$="" THEN 380
370 ON INSTR("PCTRGSMIV",A$) GO
SUB 2200,2500,2300,2800,2600,27
00,2900,2100,2400
380 IF Y<0 THEN Y=23
390 IF Y>23 THEN Y=0
400 IF X<0 THEN X=24-T
410 IF X>23 THEN X=0
420 GOTO 320
1500 PUT (X1,Y1)-(X1+5*T-1,Y1+4
),C1,NOT
1510 IF PEEK(338)=191 GOSUB2000
1520 IF PEEK(343)=247 THENX=X-T
1530 IF PEEK(344)=247 THENX=X+T
1540 IF PEEK(341)=247 THENY=Y-1
1550 IF PEEK(342)=247 THENY=Y+1
1560 RETURN
2000 GOSUB 4000
2010 P=3*Y+INT(X/8):PX=7-X+8*IN
T(X/8)
2020 IF T=2 THEN VL=F-1 ELSE VL
=-(F=1)
2030 PK=PEEK(P+VARPTR(A(0)))
2040 PK=PK AND(255.1-2^PX):IF T
=2 THEN PK=PK AND(255.1-2^(PX-1
))
2050 PK=PK OR VL*2^(PX+1-T)
2060 POKE P+VARPTR(A(0)),PK
2070 PUT (216,10)-(239,33),A,PS
ET
2080 PUT (216,70)-(239,93),A,PR
ESET
2090 RETURN
2100 RETURN
2200 RETURN
2300 A$=INKEY$:IF A$<>"S" AND A
$<>"L" THEN 2300
2310 IF A$="S" THEN 2330
2320 CLOADM:SCREEN 1,ST:RETURN
2330 CSAVEM"",6800,7679,6800
2340 SCREEN 1,ST:RETURN
2400 ST=1-ST:SCREEN 1,ST:RETURN
2500 A$=INKEY$:IF A$<"0" OR A$>
"8" THEN 2500
2510 F=((VAL(A$)-1)AND 3)+1:RET
URN
2600 A$=INKEY$:IF A$<"1" OR A$>
"8" THEN 2600
2610 J=VAL(A$)-1
2620 GET(J*32+8,165)-(J*32+31,1
88),A
2630 GOSUB 3000:GOTO 2070
2700 A$=INKEY$:IF A$<"1" OR A$>
"8" THEN 2700
2710 J=VAL(A$)-1
2720 PUT(J*32+8,165)-(J*32+31,1
88),A,PSET
2730 RETURN
2800 RETURN
2900 RETURN
3000 CL=F:POKE 30999,T-1:N=USR0
(VARPTR(A(0)))
3010 FOR K=0 TO 23:FOR J=0 TO 2
3 STEP T: F=PEEK(31500+K*24/T+J
/T)+3-T
3020 X1=J*6+4: Y1=K*6+4
3030 GOSUB 4000:NEXT J,K:F=CL:R
ETURN
4000 ON F GOTO 4010,4020,4030,4
040
4010 PUT(X1,Y1)-(X1+5*T-1,Y1+4)
,C1,PSET:RETURN
4020 PUT(X1,Y1)-(X1+5*T-1,Y1+4)
,C2,PSET:RETURN
4030 PUT(X1,Y1)-(X1+5*T-1,Y1+4)
,C3,PSET:RETURN
4040 PUT(X1,Y1)-(X1+5*T-1,Y1+4)
,C4,PSET:RETURN
```

A FUNÇÃO INKEY$ NO TK-2000

Os usuários do TK-2000 freqüentemente se decepcionam quando tentam sem sucesso adaptar programas de jogos feitos para outras máquinas, como o MSX ou TRS-Color. A causa dessa frustração é a ausência de uma função muito importante: o comando **INKEY$**.

O que faz o **INKEY$**? Existe uma maneira de substituí-lo por algum outro comando próprio do TK-2000, que cumpre a mesma função? A resposta (positiva) já foi brevemente abordada no artigo *E agora... O que Fazer?* (página 161). Aqui vamos estudar tudo isso mais detalhadamente. Assim, você mesmo poderá adaptar os programas de jogos mostrados nos artigos das páginas 28, 46, 61, 121 e 153.

O COMANDO GET

O **INKEY$** pode, em muitos casos, ser substituído pelo comando **GET**, que existe no TK-2000. A utilização desse comando é muito simples. Ele é equivalente ao **INPUT**, só que aceita apenas um caractere de cada vez, e não exige que se pressione a tecla <RETURN> para dar prosseguimento ao programa. Além disso, a tecla pressionada não é mostrada na tela.

Desse modo, muitos programas de outras máquinas, que usam o **INKEY$** apenas para realizar tais funções, podem ser adaptados para o TK-2000 com o auxílio do comando **GET**. Por exemplo, uma linha programada para interromper o fluxo do programa até que o usuário pressione uma tecla qualquer.

```
1000 PRINT "PRESSIONE QUALQUER
     TECLA"
1010 LET A$=INKEY$
1020 IF A$="" THEN GOTO 1010
```

Isso poderia até ficar mais simples no TK-2000 (e no Apple II):

```
1000 PRINT "PRESSIONE QUALQUER
     TECLA"
1010 GET A$
```

O **INKEY$** na linha 1010 do primeiro programa é uma função do BASIC, cujo objetivo é "varrer" o teclado e verificar se alguma tecla foi pressionada. Se isso acontecer, o caractere correspondente será devolvido ao programa. No caso da linha 1010, esse caractere será armazenado na variável **A$**. Se nenhuma tecla tiver sido pressionada, o cordão vazio, "", será retornado. Por isso, há necessidade da linha 1020.

Já o comando **GET** não é uma função. De forma semelhante ao comando **INPUT**, ele interrompe o programa e fica aguardando até que uma tecla qualquer seja pressionada. Só então, ele armazena esse caractere na variável-argumento (no caso, **A$**), e passa para a linha seguinte do programa.

UM GET COM MOVIMENTAÇÃO

A principal desvantagem do **GET**, entretanto, é que ele interrompe o fluxo do programa até que a tecla seja pressionada. Em contrapartida, a grande vantagem do **INKEY$** é que o computador pode ser programado para fazer outras coisas, enquanto espera que o

Apesar de seu nome estranho, o comando INKEY$ não é nenhum bicho-de-sete-cabeças. De grande importância para a programação de jogos, ele pode, contudo, ser facilmente substituído.

- COMO FUNCIONA O **INKEY$**
- UM SUBSTITUTO PARA O **INKEY$**
- APLICAÇÕES PARA O **PEEK**
- UM **INKEY$** MAIS SOFISTICADO
- LIMITES DE TEMPO

usuário pressione alguma tecla. No programa apresentado, por exemplo, poderíamos colocar outras linhas de programação entre as linhas 1010 e 1020 (para contar o tempo passando, movimentar objetos gráficos na tela etc).

Ora, o **GET** não permite fazer isso. Daí por que se torna necessário encontrar um substituto.

Não existe nenhum comando ou função específicos para isso no TK-2000. Porém, a função **PEEK**, que é usada para examinar uma posição absoluta de memória (veja artigo da página 261), pode ser usada para "varrer" o teclado. A locação de memória número 39 (decimal) serve para essa finalidade. Enquanto nenhuma tecla for pressionada essa locação conterá o número 48 (decimal). Quando uma tecla for pressionada, um valor diferente deste será depositado automaticamente na memória. O programa seguinte ilustra como podemos aproveitar essa propriedade:

```
10 HOME
20 LET K=PEEK(39)
30 IF K<>48 THEN PRINT K
40 GOTO 20
```

A linha 20 copia o conteúdo da memória 39 na variável **K**. A 30 mostrará esse valor na tela apenas se ele for diferente de 48, ou seja, se alguma tecla tiver sido pressionada. Em caso contrário, o programa saltará de novo para a linha 20, de modo a manter vigilância sobre a memória 39.

O único problema é que os códigos numéricos retornados para cada tecla pressionada não correspondem ao código ASCII dos caracteres aos quais elas estão atribuídas.

Essa dificuldade, entretanto, pode ser facilmente solucionada por uma pequena tabela de conversão, mostrada no fim deste artigo. Mas, se você preferir, pode descobrir quais são esses códigos usando o programa.

Para entender melhor como o comando **PEEK** (39) pode ser usado no lugar do comando **INKEY$**, acompanhe o programa apresentado a seguir. Ele desenha uma reta na tela, orientada em determinada direção. Pressione uma das teclas de controle do cursor (flechas) uma única vez, e a linha sofrerá uma mudança de direção.

Ao ser acionada a tecla **<FIRE>**, o programa terminará.

```
10 HGR2:HCOLOR 3
20 LET X=100:Y=90:DX=1:DY=0
30 HPLOT X,Y
40 FOR I=1 TO 40:NEXT I
50 LET P=PEEK(39)
60 IF P=48 THEN X=X+DX:Y=Y+DY:
   GOTO 30
70 IF P=36 THEN DX=0:DY=-1:GOTO
   30
80 IF P=30 THEN DX=0:DY=1:GOTO
   30
90 IF P=18 THEN DX=-1:DY=0:GOTO
   30
95 IF P=24 THEN DX=1:DY=0:GOTO
   30
100 IF P=46 THEN TEXT:END
110 GOTO 30
```

O laço principal do sistema está compreendido entre as linhas 30 e 110. A cor e a posição inicial do ponto da tela são definidos nas linhas 10 e 20. A linha 20 também define duas variáveis, **DX** e **DY**, que darão o *incremento* a ser somado a cada coordenada — X e Y —, quando o ponto for deslocado. Note que, com **DX = 1** e **DY = 0**, inicialmente o ponto vai se deslocar de um em um só na direção horizontal.

A linha 50 examina a memória vinculada ao teclado. Se nenhuma tecla tiver sido pressionada, o código resultante é 48, e as variáveis **X** e **Y** são incrementadas de **DX** e **DY**, respectivamente, na linha 60.

As linhas 70 a 100 verificam qual foi a tecla pressionada. Observe que os códigos 36, 30, 18 e 24 correspondem às teclas do cursor; se uma destas for pressionada, os valores de **DX** e **DY** serão ajustados de modo a provocar movimento na direção apontada (a linha 40, por sua vez, serve apenas para diminuir a velocidade de deslocamento do ponto na tela).

Finalmente, a linha 100 interromperá o programa, quando a tecla **<FIRE>** for pressionada.

UM INKEY$ MAIS SOFISTICADO

O uso do comando **PEEK** implica a necessidade de trabalhar com valores não padronizados de códigos de teclas. Isso pode ser desvantajoso em muitas aplicações onde se deseja receber o código ASCII da tecla, como também em programas de senha secreta (veja página 166), e outros.

Infelizmente, não existe nenhum **PEEK** apropriado para trabalhar com valores ASCII. Mas podemos "enxertar" em nosso programa em BASIC uma pequena rotina em linguagem de máquina (cujo funcionamento não será explicado por enquanto), que atribua a um **PEEK** essa propriedade.

Digite o comando **NEW** para apagar o programa anterior, e entre e execute o programa abaixo:

```
1 FOR I=768 TO 774
2 READ N:POKE I,N:NEXT I
3 DATA 32,67,240,141,0,4,96
```

Como você deve ter notado, aparentemente não acontece nada. Tudo que o programa faz é carregar um programinha de sete bytes em código de máquina numa parte não usada da memória do micro, que começa na locação 768. Isso é feito por intermédio do comando **POKE** na linha 2. O programa está armazenado na linha **DATA**.

Depois de rodar o programa, você pode apagá-lo com o comando **NEW**, pois ele não será mais necessário (o programa em código de máquina ficará armazenado até que a máquina seja desligada). Mas, se você preferir, pode também deixá-lo onde está.

Para fazer funcionar esse programa, é preciso utilizar o comando **CALL 768** (que indica o endereço onde começa o programa). Toda vez que isso é feito, o computador "varre" o teclado e retorna a um valor numérico correspondente ao código ASCII da tecla pressionada, mais 128. Se nenhuma tecla for pressionada, o código 0 será retornado. Para encontrar esse valor, usamos o comando **PEEK (1024)**.

O programa abaixo faz um teste.

```
10 HOME
20 CALL 768
30 K=PEEK(1024)
40 IF K>0 THEN PRINT K
50 GOTO 20
```

Como você já deve ter observado, a lógica do programa é bem clara: a linha 20 chama a rotina de máquina e retorna o código da tecla pressionada na locação 1024 de memória.

A linha 40 imprimirá esse valor se este for maior do que 0.

Ao executar o programa você notará que os valores retornados aparentemente não pertencem ao código ASCII. A letra A maiúscula, por exemplo, retorna 193; a letra B, 194, e assim por diante. Experimente diminuir 128 desse valor: o resultado será 65, 66 etc.: ou seja, o código ASCII.

Tente agora substituir a linha 40 pelo seguinte programa:

```
40 IF K>0 THEN PRINT CHR$(K);
```

Em seguida:

```
40 IF K>159 THEN PRINT CHR$(
   K-128);
```

Qual a diferença entre estas duas versões?

UM PEQUENO TESTE

Agora, como exemplo do uso do comando **CALL**, digite o programa:

```
10 HOME:SPEED=100
20 PRINT "TESTE DE DIGITACAO":
   PRINT
30 FOR I=1 TO 10
40 LET T=0:N$=""
50 FOR N=1 TO 7
55 LET R=48+INT(RND(1)*10)
60 N$=N$+CHR$(R):NEXT N
70 VTAB 10:HTAB 15:PRINT N$
80 VTAB 20:HTAB 1:PRINT "
   ":VTAB 20:HTAB 1
90 LET R$=""
100 FOR J=1 TO 7
110 CALL 768:K=PEEK(1024)
115 IF K>0 THEN 130
120 LET T=T+1:IF T>200 THEN 180
125 GOTO 110
130 LET K$=CHR$(K-128):R$=R$+K$
140 PRINT K$;
150 NEXT J
155 VTAB 10:HTAB 15
160 IF R$=N$ THEN PRINT
    "ACERTOU":GOTO 190
170 PRINT "ERROU !":GOTO 190
180 VTAB 10:HTAB 15:PRINT
    "DEMOROU"
```

```
190 FOR J=1 TO 1000:NEXT J
200 NEXT I
```

O programa gera uma seqüência aleatória de sete dígitos e pede que eles sejam teclados rapidamente. O objetivo do exercício é digitar o número todo, sem errar, dentro de um intervalo de tempo máximo.

As linhas 50 a 70 geram esse número, armazenando-o na variável **N$** e mostrando-o na tela. As linhas 100 a 150 efetuam a leitura das teclas pressionadas pelo usuário e a colocam em uma variável **RS**.

Finalmente, as linhas 155 a 190 verificam qual foi o resultado — este pode indicar várias alternativas, como erro, acerto, ou demora em datilografar todo o número de teste.

As linhas 110 e 115 efetuam a "varredura" do teclado para verificar se algo foi pressionado. Em seguida, a linha 120 aumenta a contagem de tempo (**T**), enquanto o usuário não pressiona nenhuma tecla. Se alguma tecla foi pressionada, as linhas 130 e 140 concatenam o resultado na variável de resposta e mostram o caractere em outra locação da tela.

COMO FUNCIONA

Quem conhece um mínimo de código de máquina será capaz de entender facilmente o funcionamento da rotina 768. Você pode tentar carregá-la no TK-2000 usando o Míni-Assembler, em lugar do programa apresentado linhas atrás. Simplesmente digite **ASS** e pressione <**RETURN**>. Depois, cada vez que aparecer um sinal de exclamação, digite as linhas abaixo.

Para sair, pressione as duas teclas <**RESET**> ao mesmo tempo:

```
300: JSR $F043
303: STA $400
306: RTS
```

O número 300 está em hexadecimal: equivale ao 768 em decimal e é o endereço de origem da rotina (mas poderia ser um número diferente deste).

O comando **JSR** pula para a subrotina em código de máquina armazenada no endereço FO43, hexadecimal, que efetua a varredura do teclado (**JSR** equivale ao **GOSUB** do BASIC).

O comando **STA** copia o conteúdo do acumulador do micro (onde está o código da tecla pressionada) e o armazena na memória 1024 ($400 em hexa; mas poderia ser outra).

Finalmente, o comando **RTS** retorna para o BASIC.

TABELA DE CÓDIGOS DE TECLA

Caractere	ASCII	PEEK (39)	CALL 768	Caractere	ASCII	PEEK (39)	CALL 768
nenhum	-	48	0	C	67	3	195
espaço	32	12	160	D	68	9	196
!	33	151	161	E	69	15	197
"	34	150	162	F	70	8	198
#	35	149	163	G	71	7	199
$	36	148	164	H	72	37	200
%	37	147	165	I	73	33	201
&	38	153	166	J	74	38	202
'	39	154	167	K	75	39	203
(	40	155	168	L	76	40	204
)	41	156	169	M	77	44	205
*	42	157	170	N	78	43	206
+	43	163	171	O	79	34	207
,	44	45	172	P	80	35	208
-	45	161	173	Q	81	17	209
.	46	46	174	R	82	14	210
/	47	175	175	S	83	10	211
0	48	29	176	U	84	32	212
1	49	23	177	V	85	2	213
2	50	22	178	W	86	16	214
3	51	21	179	X	87	4	215
4	52	20	180	Y	88	31	216
5	53	19	181	Z	89	5	217
6	54	25	182	↑	112	36	240
7	55	26	183	↓	113	30	241
8	56	27	184	←	8	18	136
9	57	28	185	→	21	24	149
A	65	11	193	<FIRE>	46	46	174
B	66	1	194	<RETURN>	13	42	141

COMO FUNCIONA O PRINT USING

- O QUE É UM **PRINT** FORMATADO
- PARA QUE SERVE O **PRINT USING**
- COMO USAR MÁSCARAS DE FORMATAÇÃO

O comando PRINT USING controla a forma com a qual aparecem na tela os números e as cadeias alfanuméricas.

Embora recursos de formatação como a vírgula, a função **TAB** etc., funcionem bem na maioria das aplicações, em alguns casos é interessante controlar não só o posicionamento de números e nomes a serem exibidos no vídeo, como também os seus formatos. Ora, isso não pode ser feito apenas pelo comando **PRINT**. Alguns micros dispõem para isso de um poderoso comando auxiliar do **PRINT**, que é chamado **USING**. Outros, porém, não contam com esse recurso. É o caso das linhas ZX-81, Spectrum, Apple II e TK-2000. Mais adiante, estudaremos um modo de superar essa dificuldade, obtendo efeitos semelhantes aos provocados pelo comando **PRINT USING**.

PARA QUE SERVE O PRINT USING

O **PRINT USING** facilita a especificação de formatos de saída em tela, possibilitando a programação de máscaras de saída. Para entender melhor como o **USING** funciona, digite o exemplo a seguir, que mostra as raízes quadradas de dez números inteiros:

```
10 CLS
20 FOR I=1 TO 10
30 PRINT USING "##    ##.####";
I;SQR(I)
40 NEXT I
```

Os números ficam alinhados em duas colunas e a coluna da direita tem todos os valores com quatro decimais. Para verificar o efeito do **PRINT USING**, substitua a linha 30 por:

```
30 PRINT I,SQR(I)
```

Para usar a especificação **USING** dentro de um **PRINT** basta colocar entre aspas o gabarito de saída. O caractere sustenido (#) indica onde deve ir cada dígito dos elementos da lista de saída. Um espaço em branco assinala onde começa e termina o gabarito relativo a cada elemento. O ponto indica onde deve ser feita a quebra entre as partes inteira e fracionária.

O gabarito estabelecido pelo **PRINT** pode ter especificações sucessivas para mais de um item de saída (no exemplo, o mesmo **PRINT** mostra a variável **I** e o resultado da raiz quadrada de **I**, sendo o primeiro com três dígitos). Enquanto existirem elementos de saída a serem impressos, *após* o **USING**, o computador continuará procurando gabaritos dentro da cadeia entre aspas.

O **USING** aceita ainda variáveis alfanuméricas.

A função **STRING$** gera uma cadeia de caracteres, com o comprimento desejado:

```
10 CLS
12 INPUT "QUANTAS DECIMAIS ";N
15 U$="##    ##." + STRING$(ND ,
"#")
20 FOR I=1 TO 10
30 PRINT USING U$;I;SQR(I)
40 NEXT I
```

A tabela apresentada nesta página relaciona os diversos sinais convencionais utilizados em um **PRINT USING** e seus efeitos para a especificação de gabaritos, para números etc.

O **PRINT USING** encontra suas aplicações mais interessantes na formatação de relatórios de saída em tela ou impressora, onde se requer um controle perfeito sobre o alinhamento de colunas, a formatação de valores numéricos etc. Uma única variável ou especificação entre aspas no **USING** pode ser usada para formatar uma linha de saída inteira da tabela. Se a linha for muito longa, o programa deve ser estruturado com base em vários **PRINT USING** separados por pontos e vírgulas.

É comum, em tabelas com valores monetários, empregar-se os sinais + e − no final (e não no começo) da quantia. Isso também é possível no **PRINT USING**.

Em programas para preenchimento automático de cheques, os sinais $ (cifrão), $$ (duplo cifrão), ** (protegido) e **$ (cifrão protegido) especificam como será preenchida a parte esquerda de campos com valores monetários:

```
10 M$=" **$###,###,###.## 
cruzados"
15 D$=" ##/##/####"
20 INPUT "QUANTIA A SER PAGA "
; Q
30 INPUT "DIA,MES,ANO";D,M,A
40 CLS
50 PRINT "PAGUE-SE A QUANTIA DE
" ;
60 PRINT USING M$;Q;
70 PRINT "NO DIA ";:PRINT USING
D$;D;M;A
```

O TRS-80 e o TRS-Color têm as mesmas regras de uso do **PRINT USING**. Os sinais demarcadores são os mesmos, e a palavra **USING** deve suceder a expressão **PRINT**. Já o MSX apresenta algumas diferenças:

- Em vez do sinal de porcentagem (%), esse micro utiliza uma barra inversa (\) para demarcar o espaço das cadeias alfanuméricas.
- O ampersand (&) serve para indicar uma substituição completa por uma cadeia alfanumérica.
- O termo **USING** pode ser misturado aos itens de uma linha de impressão:

```
70 PRINT "NO DIA ";USING D$;D;
M;A
```

CARACTERES DO PRINT USING

Sinal	Efeito
#	coloca um dígito numérico
.	coloca um ponto decimal
,	separa a parte inteira em grupos de três
−	coloca um sinal negativo após um número negativo, ou branco após um positivo
+	coloca um sinal − após um número negativo ou um sinal + após um positivo
^	assinala um formato exponencial
**	preenche um campo com asteriscos à esquerda
$$	coloca um cifrão antes de um valor numérico
**$	coloca um cifrão e asteriscos à esquerda
!	coloca o primeiro caractere de uma cadeia
%	assinala o início e o fim de um campo alfanumérico

CLASSIFICAÇÃO DOS MICROCOMPUTADORES POR LINHA

LINHA	FABRICANTE	MODELO
Apple II +	Appletronica	Thor 2010
Apple II +	CCE	MC-4000 Exato
Apple II +	CPA	Absolutus
Apple II +	CPA	Polaris
Apple II +	Digitus	DGT-AP
Apple II +	Dismac	D-8100
Apple II +	ENIAC	ENIAC II
Apple II +	Franklin	Franklin
Apple II +	Houston	Houston AP
Apple II +	Magnex	DM II
Apple II +	Maxitronica	MX-2001
Apple II +	Maxitronica	MX-48
Apple II +	Maxitronica	MX-64
Apple II +	Maxitronica	Maxitronic I
Apple II +	Microcraft	Craf II Plus
Apple II +	Milmar	Apple II Plus
Apple II +	Milmar	Apple Master
Apple II +	Milmar	Apple Senior
Apple II +	Omega	MC-400
Apple II +	Polymax	Maxxi
Apple II +	Polymax	Poly Plus
Apple II +	Spectrum	Microengenho I
Apple II +	Spectrum	Spectrum ed
Apple II +	Suporte	Venus II
Apple II +	Sycomig	SIC I
Apple II +	Unitron	AP II
Apple II +	Victor do Brasil	Elppa II Plus
Apple II +	Victor do Brasil	Elppa Jr.
Apple IIe	Microcraft	Craft IIe
Apple IIe	Microdigital	TK-3000 IIe
Apple IIe	Spectrum	Microengenho II
MSX	Gradiente	Expert GPC-1
MSX	Sharp	Hotbit HB-8000
Sinclair Spectrum	Microdigital	TK-90X
Sinclair Spectrum	Timex	Timex 2000
Sinclair ZX-81	Apply	Apply 300
Sinclair ZX-81	Engebras	AS-1000
Sinclair ZX-81	Filcres	NEZ-8000
Sinclair ZX-81	Microdigital	TK-82C
Sinclair ZX-81	Microdigital	TK-83
Sinclair ZX-81	Microdigital	TK-85
Sinclair ZX-81	Prologica	CP-200
Sinclair ZX-81	Ritas	Ringo R-470
Sinclair ZX-81	Timex	Timex 1000
Sinclair ZX-81	Timex	Timex 1500
TRS-80 Mod. I	Dismac	D-8000
TRS-80 Mod. I	Dismac	D-8001/2
TRS-80 Mod. I	LNW	LNW-80
TRS-80 Mod. I	Video Genie	Video Genie I
TRS-80 Mod.III	Digitus	DGT-100
TRS-80 Mod.III	Digitus	DGT-1000
TRS-80 Mod.III	Kemitron	Naja 800
TRS-80 Mod.III	Prologica	CP-300
TRS-80 Mod.III	Prologica	CP-500
TRS-80 Mod.III	Sysdata	Sysdata III
TRS-80 Mod.III	Sysdata	Sysdata Jr.
TRS-80 Mod.IV	Multix	MX-Compacto
TRS-80 Mod.IV	Sysdata	Sysdata IV
TRS-Color	Codimex	CS-6508
TRS-Color	Dynacom	MX-1600
TRS-Color	LZ	Color 64
TRS-Color	Microdigital	TKS-800
TRS-Color	Prologica	CP-400

CLASSIFICAÇÃO DOS MICROCOMPUTADORES POR NOME

FABRICANTE	MODELO	PAÍS	LINHA
Appletronica	Thor 2010	Brasil	Apple II +
Apply	Apply 300	Brasil	Sinclair ZX-81
CCE	MC-4000 Exato	Brasil	Apple II +
CPA	Absolutus	Brasil	Apple II +
CPA	Polaris	Brasil	Apple II +
Codimex	CS-6508	Brasil	TRS-Color
Digitus	DGT-100	Brasil	TRS-80 Mod.III
Digitus	DGT-1000	Brasil	TRS-80 Mod.III
Digitus	DGT-AP	Brasil	Apple II +
Dismac	D-8000	Brasil	TRS-80 Mod. I
Dismac	D-8001/2	Brasil	TRS-80 Mod. I
Dismac	D-8100	Brasil	Apple II +
Dynacom	MX-1600	Brasil	TRS-Color
ENIAC	ENIAC II	Brasil	Apple II +
Engebras	AS-1000	Brasil	Sinclair ZX-81
Filcres	NEZ-8000	Brasil	Sinclair ZX-81
Franklin	Franklin	USA	Apple II +
Gradiente	Expert GPC1	Brasil	MSX
Houston	Houston AP	Brasil	Apple II +
Kemitron	Naja 800	Brasil	TRS-80 Mod.III
LNW	LNW-80	USA	TRS-80 Mod. I
LZ	Color 64	Brasil	TRS-Color
Magnex	DM II	Brasil	Apple II +
Maxitronica	MX-2001	Brasil	Apple II +
Maxitronica	MX-48	Brasil	Apple II +
Maxitronica	MX-64	Brasil	Apple II +
Maxitronica	Maxitronic I	Brasil	Apple II +
Microcraft	Craft II Plus	Brasil	Apple II +
Microcraft	Caft IIe	Brasil	Apple IIe
Microdigital	TK-3000 IIe	Brasil	Apple IIe
Microdigital	TK-82C	Brasil	Sinclair ZX-81
Microdigital	TK-83	Brasil	Sinclair ZX-81
Microdigital	TK-85	Brasil	Sinclair ZX-81
Microdigital	TK-90X	Brasil	Sinclair Spectrum
Microdigital	TKS-800	Brasil	TRS-Color
Milmar	Apple II Plus	Brasil	Apple II +
Milmar	Apple Master	Brasil	Apple II +
Milmar	Apple Senior	Brasil	Apple II +
Multix	MX-Compacto	Brasil	TRS-80 Mod.IV
Omega	MC-400	Brasil	Apple II +
Polymax	Maxxi	Brasil	Apple II +
Polymax	Poly Plus	Brasil	Apple II +
Prologica	CP-200	Brasil	Sinclair ZX-81
Prologica	CP-300	Brasil	TRS-80 Mod.III
Prologica	CP-400	Brasil	TRS-Color
Prologica	CP-500	Brasil	TRS-80 Mod.III
Ritas	Ringo R-470	Brasil	Sinclair ZX-81
Sharp	Hotbit HB-8000	Brasil	MSX
Spectrum	Microengenho I	Brasil	Apple II +
Spectrum	Microengenho II	Brasil	Apple IIe
Spectrum	Spectrum ed	Brasil	Apple II +
Suporte	Venus II	Brasil	Apple II +
Sycomig	SIC I	Brasil	Apple II +
Sysdata	Sysdata III	Brasil	TRS-80 Mod.III
Sysdata	Sysdata IV	Brasil	TRS-80 Mod.IV
Sysdata	Sysdata Jr.	Brasil	TRS-80 Mod.III
Timex	Timex 1000	USA	Sinclair ZX-81
Timex	Timex 1500	USA	Sinclair ZX-81
Timex	Timex 2000	USA	Sinclair Spectrum
Unitron	AP II	Brasil	Apple II +
Victor do Brasil	Elppa II Plus	Brasil	Apple II +
Victor do Brasil	Elppa Jr.	Brasil	Apple II +
Video Genie	Video Genie I	USA	TRS-80 Mod. I

UM LOGOTIPO PARA CADA MODELO DE COMPUTADOR

INPUT foi especialmente projetado para microcomputadores compatíveis com as sete principais linhas existentes no mercado.
Os blocos de textos e listagens de programas aplicados apenas a determinadas linhas de micros podem ser identificados por meio dos seguintes símbolos:


Sinclair ZX-81


TRS-80


TK-2000


MSX


Spectrum


TRS-Color


Apple II

Quando o emblema for seguido de uma faixa, então tanto o texto como os programas que se seguem passam a ser específicos para a linha indicada.

NO PRÓXIMO NÚMERO

APLICAÇÕES
Um programa para completar a geração de blocos gráficos.
Inverta cores e formas e crie novos caracteres.

PROGRAMAÇÃO DE JOGOS
De como uma víbora desnutrida pode se transformar numa serpente gigantesca. Divirta-se com o jogo da cobra.

PROGRAMAÇÃO BASIC
Como planejar cuidadosamente a página-título e outras telas do seu programa.

CÓDIGO DE MÁQUINA
Como é armazenado um programa em BASIC.
O uso do **PEEK**.