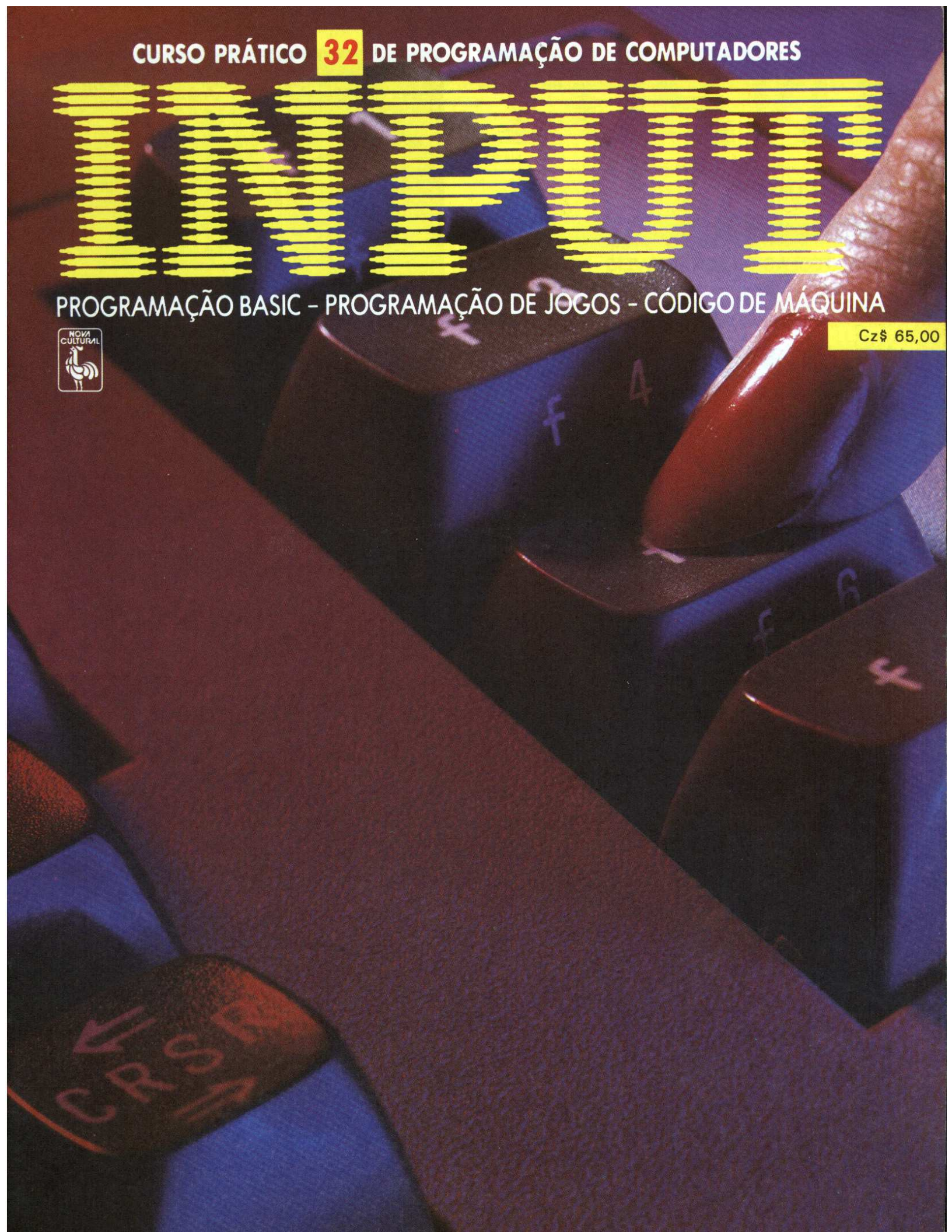
CURSO PRÁTICO 32 DE PROGRAMAÇÃO DE COMPUTADORES

INFO

PROGRAMAÇÃO BASIC - PROGRAMAÇÃO DE JOGOS - CÓDIGO DE MÁQUINA

NOVA CULTURAL

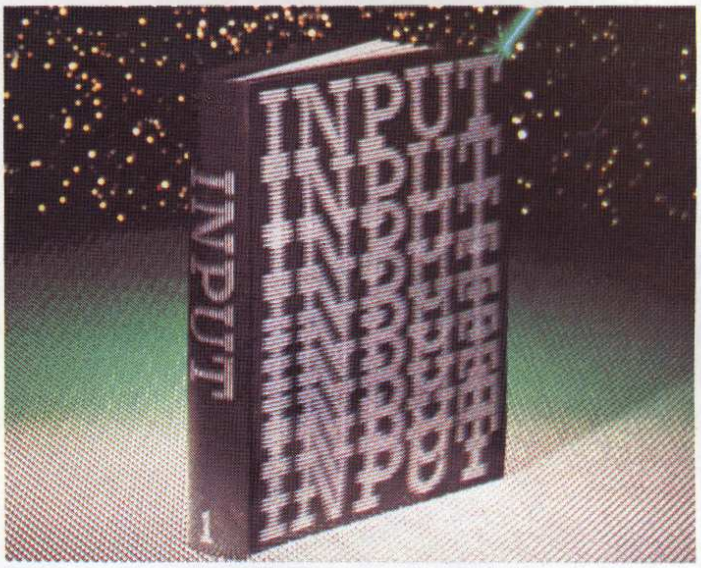
Cz$ 65,00



INPUT

Vol. 3 Nº 32

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:
Berta Sztark Amar, Stefania Crema
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, Cultura S.A.
Presidente: Richard Civita
Diretoria: Flávio Barros Pinto, João Gomez,
Menahem 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.

MSX: TECLAS PROGRAMÁVEIS

- O QUE SÃO TECLAS DE FUNÇÃO
- PROGRAMAÇÃO DAS TECLAS
- MONTAGEM DE UM MENU
- COMO DETECTAR INTERRUPÇÕES
- PROGRAME A TECLA <**STOP**>

O MSX oferece um recurso fantástico para dar aos programas um acabamento profissional: as teclas programáveis. Explorando-as bem, você conseguirá menus de funções instantâneos na tela..

A maioria dos computadores profissionais, como os cobiçados micros da linha PC, de dezesseis bits, tem teclados imensos, com cem teclas ou mais. Isso não só simplifica o acesso a um grande número de funções embutidas no sistema, pela pressão a uma única tecla, como permite que o usuário reprograme a função de parte das teclas, de acordo com suas necessidades.

Em número de dez a doze, as teclas programáveis pelo usuário, também chamadas de *teclas de função*, em geral se situam num bloco isolado, na parte

superior ou lateral do teclado. Essa disposição facilita a correspondência entre as teclas e os rótulos a elas atribuídos, cujo posicionamento pode ser exibido na última linha do vídeo, também sob controle de programa.

Os micros da linha MSX dispõem de cinco teclas programáveis, que estão localizadas na parte superior do teclado. Na realidade, elas possibilitam o acesso a dez funções, pois, pressionando-se a tecla <**SHIFT**> simultaneamente com uma tecla programável, ativa-se uma segunda função que pode ser atribuída àquela tecla.

Os recursos de programação disponíveis no interpretator BASIC do MSX são muito ricos. Aprendendo a explorá-los de maneira criativa, você se habilitará a produzir programas com um desempenho altamente profissional. Neste artigo, apresentamos alguns truques interessantes, muitos dos quais não se encontram no Manual de BASIC que acompanha o computador. Você logo será capaz de aproveitá-los com sucesso em outros programas.

TECLAS PROGRAMÁVEIS

As teclas de função podem ser usadas de dois modos em um programa:

— para dar entrada a uma cadeia de caracteres no computador, pressionando-se uma única tecla;

— para interromper automaticamente a execução de um programa e redirecioná-lo para executar uma ou mais sub-rotinas específicas, cada qual associada a uma tecla programável.

Em ambos os casos, é possível (mas não obrigatório) exibir em uma linha reservada (a última linha da tela) uma lista de rótulos de identificação das teclas. Assim, o usuário não precisará recorrer sempre à memória para saber o que faz cada tecla programável. Essa linha, uma vez "ligada", fica presente na tela, mesmo quando seu conteúdo "rola".

COMANDOS INSTANTÂNEOS

O modo mais simples de utilização das teclas programáveis, que consiste em dar entrada a uma cadeia de caracteres, é exemplificado pela operação normal do próprio computador.

Como você já deve ter observado, ao ligar a máquina (ou ao pressionar o botão <**RESET**> de inicialização, existente em alguns micros da linha MSX), aparece uma série de rótulos na linha inferior da tela. Eles correspondem a comandos ou instruções do BASIC, ou seja, **COLOR**, **LIST**, **RUN**, **GOTO** etc.

Se você pressionar uma das teclas programáveis (identificadas na parte superior do teclado pelos rótulos **F1**, **F2** etc.), verá que a palavra a ela associada é entrada por extenso no computador, aparecendo na tela logo adiante do cursor.

Caso pressione a tecla <**SHIFT**> o conteúdo da linha inferior da tela mudará para outros quatro rótulos adicionais, que correspondem às funções das teclas **F6** a **F10**. Portanto, a pressão simultânea das teclas **F1** e <**SHIFT**> provocará a entrada automática da cadeia associada.

O carregamento dos rótulos associados a cada tecla é feito automaticamente, no momento de inicialização do interpretador BASIC. Para apagar a linha da tela, basta dar o comando:

```
KEY OFF
```

seja em modo direto, seja dentro de um programa.

Tente fazer isso e, em seguida, pressione novamente uma das teclas de função. Você verá que ela continua funcionando — a cadeia de caracteres a que se associa aparece do mesmo jeito na tela. A função do comando **KEY OFF**, portanto, resume-se ao desligamento da linha de exibição.

O comando seguinte faz exatamente o contrário do anterior, ou seja, torna a exibir a linha de tela associada às teclas programáveis:

```
KEY ON
```

Este comando também pode ser digitado em modo direto ou ser colocado dentro de um programa.

No caso da programação BASIC, a vantagem de se associar os comandos mais comuns às teclas programáveis é evidente: em vez de digitar o comando por extenso, pressiona-se a tecla programável uma vez. Com isso, evitamos erros de digitação e aceleramos consideravelmente o processo de entrada de um programa BASIC.

Experimentando todas as teclas programáveis, você observará que algumas delas, ao serem pressionadas, simplesmente dão entrada à cadeia de caracteres (por exemplo, **LIST**), e mais nada acontece. O cursor fica parado no fim da palavra entrada. Para que o comando seja executado, você deverá pressionar a tecla <**ENTER**>, que o envia para o processador. No exemplo dado, a execução resultaria na listagem de um programa inteiro na tela.

Você pode, porém, digitar o número ou os números de linha sobre os quais o comando **LIST** vai atuar, e só depois pressionar <**ENTER**>.

Algumas teclas programáveis, por outro lado, dão entrada à palavra associada, mas a executam imediatamente — é o caso de **RUN**. Como veremos adiante, essa diferença de funcionamento pode ser especificada facilmente pelo programador. Tudo depende da função que se quer associar a cada tecla.

Para verificar na tela quais os rótulos que se associam a cada uma das teclas programáveis, digite o comando **KEY LIST**. As palavras que são enviadas imediatamente, sem a necessidade do <**ENTER**>, aparecem assinaladas por uma flechinha no final. Sabe-se, desse modo, que o caractere de controle equivalente ao <**ENTER**> foi agregado ao final da palavra.

PROGRAMAÇÃO DAS TECLAS

É muito fácil atribuir uma cadeia de caracteres a uma tecla programável, e pode-se fazê-lo tanto em modo direto quanto dentro de um programa. A instrução em BASIC a ser utilizada é:

```
KEY n,"cadeia"
```

onde **n** é o número da tecla de função, e "**cadeia**", o rótulo que deve ser associado à tecla. Experimente digitar, por exemplo:

```
KEY 1,"CLS"
```

O rótulo da tecla **F1**, que inicialmente era **COLOR**, passa a ser **CLS**, ao ser inicializado o computador. Confira, digitando **KEY LIST**.

Agora, pressione a tecla **F1**. Você verá então que o comando **CLS** aparece instantaneamente na tela, logo após o cursor, mas não é executado. Para que isso ocorra, você deverá pressionar a tecla <**ENTER**>.

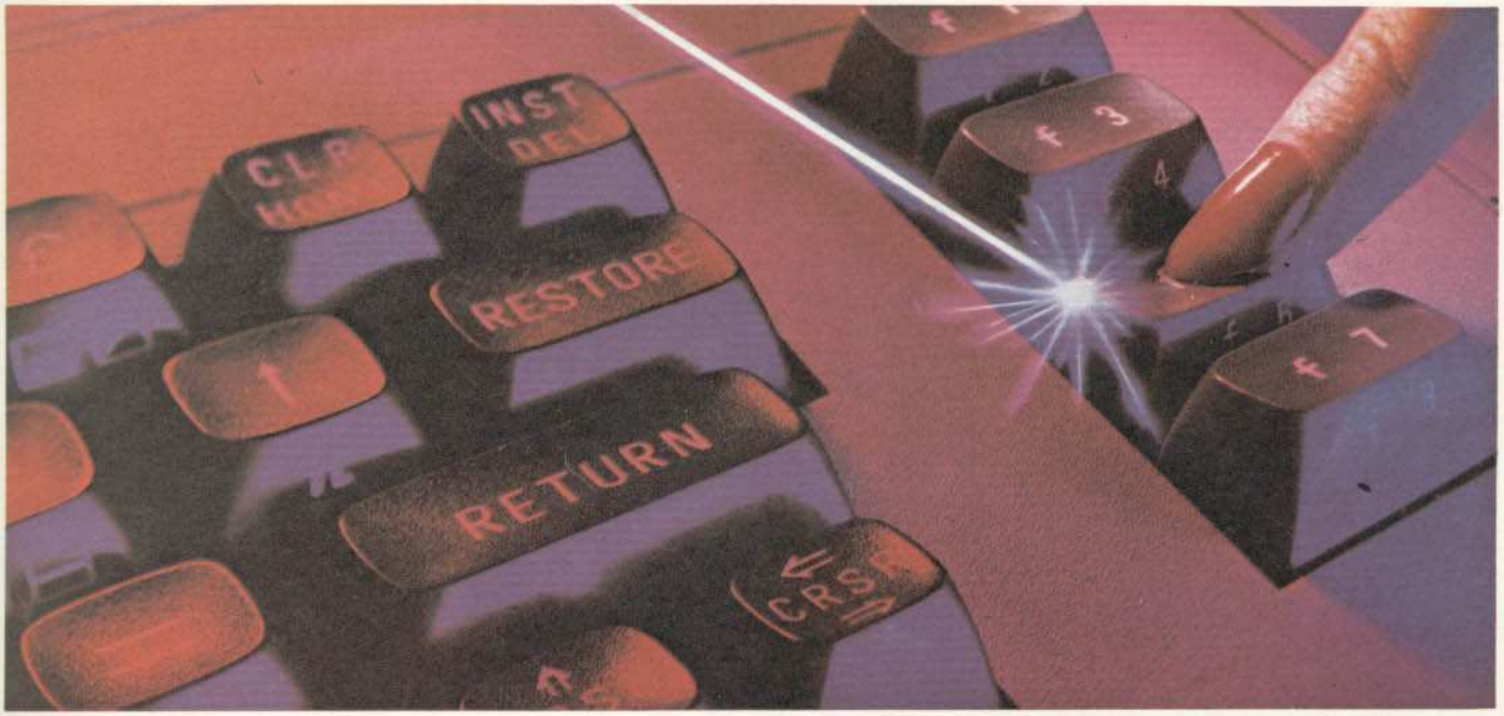
Se quisermos que a execução do comando seja automática, precisaremos definir a tecla de outro modo:

```
KEY 1,"CLS"+CHR$(13)
```

Ao listar o conteúdo das teclas programáveis com **KEY LIST**, você constatará que a marca de retorno foi acrescentada no fim da palavra **CLS**.

O caractere ASCII 13, adicionado à cadeia de caracteres por meio da função **CHR$**, corresponde exatamente ao código gerado internamente ao se pressionar a tecla <**ENTER**>. Ou seja, equivale a pressioná-la automaticamente, quando a tecla **F1** é acionada.

Cabem exatamente seis caracteres no espaço alocado para o rótulo de cada te-



cla, na linha de menu. Isto é o que cabe na tela, o que não significa que você esteja limitado a definir um rótulo de apenas seis caracteres. Experimente digitar:

```
KEY 1,"DEFUSR1=31100"+CHR$(13)
```

Apenas as seis primeiras letras (**DEFUSR**) aparecerão na tela, mas a cadeia será enviada por extenso quando se pressionar a tecla **F1**. Portanto, se for necessário "esconder" a marca de retorno em um rótulo, basta digitar a seguinte linha:

```
KEY 1,"CLS    "+CHR$(13)
```

Deixe três espaços entre o final da palavra e o segundo sinal de aspas, nesse exemplo.

Se a cadeia de caracteres a ser associada à tecla tiver que conter aspas em seu interior, não dará certo fazer algo como:

```
KEY 1,"LOAD "" CODE"
```

A função **CHR$**, mais uma vez, resolve o dilema:

```
KEY 1,"LOAD "+CHR$(44)+
CHR$(44)+" CODE"
```

O código ASCII 44, evidentemente, corresponde ao caractere aspas.

OUTROS USOS

Não é obrigatório atribuir às teclas programáveis apenas comandos ou instruções em BASIC. Na realidade, pode-se atribuir a elas qualquer cadeia de caracteres, o que possibilita muitas aplicações interessantes.

Suponhamos, por exemplo, que você vá passar uma longa mensagem ao computador usando um programa de processamento de textos. Se notar que certas palavras ou frases mais longas serão empregadas várias vezes (como **microcomputador**, **Secretaria Especial de Informática** etc.), bastará carregar previamente as teclas de função com estas cadeias (não seguidas do caractere ASCII 13). Pelas seis primeiras letras da frase, você poderá identificar a tecla associada a ela e, simplesmente, pressioná-la. Isso poupará muito tempo de digitação!

TÉCNICAS DE PROGRAMAÇÃO DE MENUS

Como já foi mencionado, as teclas programáveis são muito utilizadas para a programação rápida de menus, oferecendo duas vantagens: o menu fica sempre presente na linha reservada da tabela, e é muito mais fácil localizar as teclas associadas às suas opções do que se estivéssemos utilizando teclas normais do teclado.

Os programas que se seguem vão mostrar algumas das técnicas mais comuns de programação de menus utilizando as teclas de função.

Suponhamos que você queira elaborar um programa de banco de dados, com as quatro funções clássicas: inserir registros, apagar registros, modificar registros e listar o banco de dados. Além dessas funções precisaríamos de uma quinta opção: a de término da execução do programa.

Já vimos como montar um menu desse tipo de diversas maneiras — usando **PRINT**, **INPUT**, **INKEY$** etc. Para usar teclas programáveis, poderíamos fazer o seguinte:

```
10 ST$=STRING$(38,"_")
20 KEY OFF:CLS
30 KEY 1,"Insere"
40 KEY 2,"Apaga "
50 KEY 3,"Modif "
60 KEY 4,"Lista "
70 KEY 5,"FIM   "
80 LOCATE 10,0
90 PRINT "BANCO DE DADOS"
100 LOCATE 0,1:PRINT ST$
110 LOCATE 0,21:PRINT ST$
120 KEY ON
130 A$=INPUT$ (6)
140 LOCATE 0,10
150 PRINT "FUNÇÃO: ";A$
160 IF A$="FIM   " THEN CLS:END
170 GOTO 130
```

A linha 10 define uma variável, **ST$**, igual a uma cadeia de 38 tracinhos: é a linha de separação, usada para dar melhor aparência à tela de entrada. A linha 20 "desliga" a linha de identificação das teclas programáveis e limpa a tela. As linhas 30 a 70 programam as teclas de função **F1** a **F5**. As linhas de 90 a 110 compõem a tela e a 120 "liga" novamente a linha de exibição dos rótulos das teclas de função, desta vez com os rótulos que foram atribuídos no programa.

Finalmente, as linhas 130 a 150 processam o resultado da pressão às teclas de função. Observe que a instrução

INPUT$**, própria do MSX, tem um funcionamento semelhante ao do **INKEY$, só que inclui um determinado número de pressões às teclas, de cada vez. **INPUT$(6)** significa que o computador deve esperar que seis caracteres sejam digitados (não os mostra na tela) para, então, prosseguir o programa.

Note que colocamos seis caracteres em cada rótulo, preenchendo-os com espaços em branco ao final, quando necessário. Assim, qualquer pressão a uma tecla programável imediatamente atenderá à condição da linha 130, armazenando o resultado na variável **A$** e seguindo para as linhas 140 a 160.

Às linhas 140 e 150 cabe simplesmente escrever o resultado dessa atuação no meio da tela. A linha 160 exemplifica o que fazer a partir daí. Ela verifica se **FIM** foi pressionado e executa **CLS:END**, terminando o programa, se isto aconteceu.

Diversos **IF** seriam utilizados para chamar (**GOSUB**) as rotinas de inserção, apagamento etc., no nosso banco de dados. A linha 170 faz tudo voltar ao menu. Pode ser interessante dar um **KEY OFF** antes de chamar as rotinas, para que o usuário não tenha a impressão de que o menu ainda continua disponível. Isso pode ser feito seletivamente, como veremos mais adiante.

O programa seguinte, que é uma modificação do anterior, testa o resultado da pressão a uma tecla de função com mais economia.

```
10 ST$=STRING$(38,"_")
20 KEY OFF:CLS
30 KEY 1,"Insere"
40 KEY 2,"Apaga "
50 KEY 3,"Modif "
60 KEY 4,"Lista "
70 KEY 5,"FIM   "
80 LOCATE 10,0
90 PRINT "BANCO DE DADOS"
100 LOCATE 0,1:PRINT ST$
110 LOCATE 0,21:PRINT ST$
120 KEY ON
130 A$=LEFT$(INPUT$(6),1)
140 LOCATE 0,10
150 ON INSTR("IAMLF",A$) GOSUB
170,180,190,200,900
155 GOTO 130
170 PRINT "ROTINA DE INSERÇÃO
 "
175 RETURN
180 PRINT "ROTINA DE APAGAMENTO
 "
185 RETURN
190 PRINT "ROTINA DE MODIFICAÇÃ
O"
195 RETURN
200 PRINT "ROTINA DE LISTAGEM
 "
205 RETURN
900 REM FIM DO PROGRAMA
905 CLS:END
```

Observe que, agora, a linha 130 é usada para extrair e armazenar em **A$** apenas o primeiro caractere do rótulo da tecla de função acionada. Portanto, os rótulos precisam ter, como no exemplo, letras iniciais diferentes, para evitar confusões.

Na linha 15 está a vantagem dessa técnica. Ela permite utilizar a forma bem mais compacta do **ON...GOSUB**, para dirigir o programa à rotina chamada pelo menu.

A função **INSTR**, que talvez você não conheça, significa, em inglês, *in string*, ou seja, *dentro de um cordão*. Ela verifica se um caractere ou cadeia de caracteres está presente dentro de uma outra cadeia. Em caso negativo, o valor zero é retornado. Em caso afirmativo, a posição do caractere (sua ordem no primeiro cordão) é retornada. Por exemplo:

```
PRINT INSTR ("COMPUTADOR","P")
```

retorna o valor 4, enquanto:

```
LET A$="X"
PRINT INSTR ("COMPUTADOR",A$)
```

retorna o valor zero.

No programa, o cordão **IAMLF** especificado corresponde às letras iniciais das cinco opções oferecidas pelo menu. Sendo pressionada uma das teclas programáveis, a letra inicial do rótulo é armazenada em **A$** (linha 130) e testada pela função **INSTR**, na linha 150. O número retornado serve para endereçar a sub-rotina que será selecionada em **ON...GOSUB.**

A listagem do programa anterior mostra apenas a primeira e a última linhas das rotinas. Se estiver interessado, complete-as você mesmo.

INTERRUPÇÕES PROGRAMADAS

Os programas vistos até aqui parecem não apresentar vantagens mais significativas em relação à programação convencional de menus que aprendemos anteriormente. Na realidade, eles apenas acrescentam a possibilidade de utilização das teclas programáveis e da linha reservada de rotulação.

Entretanto, as teclas de função do micro MSX oferecem um recurso bem mais poderoso de programação: a *interrupção programada*.

Quando pressionamos uma das teclas de função **F1** a **F10**, o comportamento do computador é diferente do observado em relação às teclas normais do teclado. Em vez de simplesmente gerar um caractere ou cordão de caracteres, a pressão à tecla de função "avisa" o interpretador BASIC — qualquer que seja o ponto do programa que esteja sendo executado — que tal pressão ocorreu, gerando o que se denomina, em linguagem técnica, de interrupção do processador (UCP).

Para entender como funciona uma interrupção pelo teclado, basta tomar como exemplo uma tecla que já é préprogramada para gerar um tipo específico de interrupção: a tecla <**STOP**>.

O sistema operacional e interpretador BASIC do MSX está sempre "atento" à tecla <**STOP**>, por meio de um dispositivo especial de hardware, para detectar se ela foi pressionada. A verificação é feita a cada poucos microssegundos e, não importa o que o computador esteja fazendo no momento da interrupção, imediatamente o processador passa a executar uma rotina de serviço que, no caso, servirá para "congelar" um programa em execução ou interrompê-lo, retornando ao modo de entrada (quando <**CONTROL**> e <**STOP**> são pressionadas simultaneamente).

Pois bem, **F1** a **F10** são teclas de interrupção *programáveis*, ou seja, o programador pode especificar a rotina de serviço que deve ser acionada pelo sistema operacional correspondente a cada tecla.

Existem diversas instruções do BASIC que permitem programar interrupções a partir das teclas **F1** a **F10**. Uma delas consiste em:

```
ON KEY GOSUB n1,n2...
```

que verifica se alguma tecla de função foi pressionada, ou seja, se ocorreu uma interrupção. Conforme a tecla pressionada, um número inteiro entre 1 e 10 é retornado. Ele chama a sub-rotina na ordem indicada, nas linhas n1, n2 etc. **ON KEY GOSUB 1000, 1100**, por exemplo, provocará um salto para a sub-rotina que começa na linha 1000, se a tecla F1 for acionada, ou para a linha 1100, se a tecla F2 for acionada. A instrução **ON KEY GOSUB** deve ser colocada em um ponto da listagem que assegure a sua execução pelo menos uma vez, antes das detecções de interrupção, que, como a rotulação das teclas, ocorrem no começo do programa.

Porém, precisamos de mais uma condição para montar a "armadilha" que detecta interrupções. A capacidade de gerar interrupções pode ser "ligada" ou "desligada" de cada tecla de função individualmente, pelas instruções:

```
KEY (n) ON

KEY (n) OFF

KEY (n) STOP
```

onde **n** é o número da tecla. Quando se liga o computador, todas as teclas estão em modo **OFF**, ou seja, "desligadas". Ao ser detectada uma pressão à tecla de função, um **KEY STOP** é dado automaticamente, inibindo toda a interrupção subseqüente, até que ocorra um **RETURN** da sub-rotina chamada pelo **ON KEY GOSUB**.

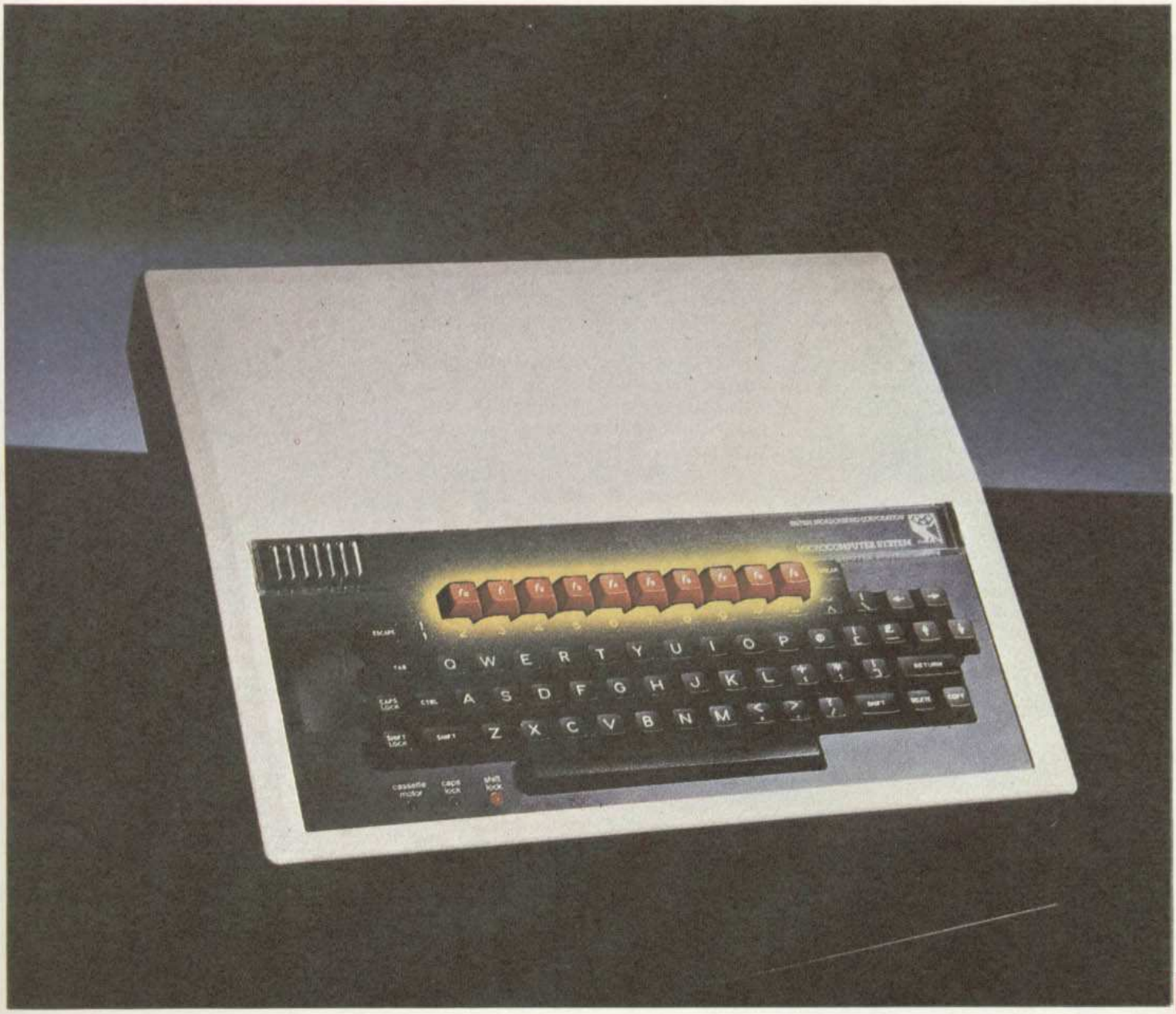
Embora o comando **KEY (n) STOP** iniba a interrupção do programa, ele continua sendo capaz de detectar uma pressão à tecla de função indicada. Assim, ao retornar da rotina de serviço de uma tecla de função, o programa poderá imediatamente rumar para outra rotina, se, enquanto isso, outra tecla de função tiver sido pressionada.

Se o programador quiser inibir tanto a interrupção quanto a detecção, deve usar um comando **KEY (n) OFF**.

No programa seguinte você encontrará exemplos das técnicas mais elementares de utilização das instruções **ON KEY GOSUB** e **KEY ON**. O objetivo é imitar uma máquina de somar simples, que apresenta o resultado quando a tecla **F1** é pressionada.

```
10 CLS
20 PRINT TAB(10);"SOMADORA MSX
I"
30 ON KEY GOSUB 200
40 S=0
50 KEY (1) ON:KEY 1,"SOMA  "
60 KEY 2," ":KEY 3," " : KEY
4," ":KEY 5," "
120 '
130 '   ALÇA DE SOMA
140
150 LOCATE 6,10
160 PRINT "                    "
170 LOCATE 0,10
180 INPUT "VALOR ";V
190 S=S+V:GOTO 150
200 '
210 '   SUBROTINA PARA TECLA F1
220 '
230 LOCATE 0,12
240 PRINT "SOMA = ";S
250 RETURN 150
```

A linha 30, logo no começo do programa, prepara a armadilha para detectar interrupções pelas teclas de função e desviar o fluxo de processamento. A linha 40 zera a variável **S**, que é o acumulador de somas; a linha 50 "liga" a tecla **F1** e rotula seu propósito na linha



reservada para exibição na tela. A linha 60 "limpa" os rótulos das teclas programáveis que não serão usadas.

As linhas 150 a 190 do programa realizam a função de soma de valores. Elas formam uma alça sem fim, o que é típico de todo programa que utiliza as teclas de função do contexto do **ON KEY GOSUB**. A alça sem fim faz o programa esperar ou executar alguma coisa (no caso somar os números entrados pelo usuário), até que uma tecla de interrupção seja pressionada.

Se se pressionar a tecla **F1** enquanto o programa está esperando a entrada de um valor, nada acontece de imediato. Mas, assim que a tecla <**ENTER**> for pressionada, a rotina que começa na linha 200 será acionada, pois foi a especificada na linha 30. Ela simplesmente mostra a soma acumulada até o último número entrado e retorna para a linha 30, de modo a dar prosseguimento à soma.

E o que acontece se alguma tecla programável não desativada for acidentalmente pressionada? Não acontece nada, pois o funcionamento do **ON KEY GOSUB** é tal que o desvio se realiza apenas para as sub-rotinas indicadas.

UM PROGRAMA MAIS FUNCIONAL

Embora não apresente problemas de funcionamento, o programa anterior tem dois defeitos: não oferece provisão para interrupção ao final do processamento, nem a possibilidade de zerar o acumulador para permitir nova soma sem sair do programa. O seguinte, mais completo, faz tudo isso:

```
10 CLS
20 PRINT TAB(10);"SOMADORA MSX
II"
30 ON KEY GOSUB 200,300,400
40 S=0
50 KEY (1) ON:KEY 1,"SOMA  "
60 KEY (2) ON:KEY 2,"LIMPA "
70 KEY (3) ON:KEY 3,"FIM   "
80 KEY 4," ":KEY 5," "
120 '
130 '   ALÇA DE SOMA
140 '
150 LOCATE 6,10
160 PRINT "                  "
170 LOCATE 0,10
180 INPUT "VALOR ";V
190 S=S+V:GOTO 150
200 '
210 '   SUBROTINA PARA TECLA F1
220 '
230 LOCATE 0,12
240 PRINT "SOMA = ";S
250 RETURN 150
300 '
310 '   SUBROTINA PARA TECLA F2
320 '
330 LOCATE 0,12
340 PRINT "SOMA = 0          "
350 S=0:RETURN 150
400 '
410 '   SUBROTINA PARA TECLA F3
420 '
430 CLS:END
```

Agora, a linha 30 permite o desvio para três rotinas: de soma (linha 200), de zeragem (linha 300) e de término (linha 400). Experimente pressionar outras teclas de função enquanto uma rotina está sendo executada: se você conseguir fazê-lo a tempo, notará que nada acontece.

Em um programa curto como este, a definição individual de cada **KEY** é mais do que satisfatória. Entretanto, se precisar incluir muitas teclas, você poderá obter um programa mais compacto e elegante fazendo esta definição dentro de um laço de repetição. Para ter um exemplo, substitua estas linhas no programa anterior:

```
50 FOR I=1 TO 5
60 READ C$:KEY (I) ON
70 KEY I,C$:NEXT I
80 DATA "SOMA  ","LIMPA ","FIM
 "
```

O laço que vai das linhas 50 a 70 faz um índice **I** variar de 1 a 5. A linha 60 lê sucessivamente os rótulos das teclas, em **DATA**. Observe que os rótulos não usados são definidos como um espaço em branco.

As linhas 60 e 70 ainda se encarregam, respectivamente, de "ligar" a tecla e atribuir-lhe o rótulo **C$**. Note que os comandos aceitam variáveis como argumentos.

Ao experimentar o programa, você deve ter reparado que as teclas programáveis não interrompem o programa enquanto ele está esperando entradas pelo usuário, em um comando **INPUT**. Como se trata de um inconveniente, recorreremos a um expediente que possibilitará a ocorrência de interrupção em qualquer momento. A condição para isso é que o computador esteja dentro de um laço de repetição durante a entrada de um valor na linha 30. Tal condição pode ser atendida se "simularmos" um comando **INPUT**, por meio de vários **INKEY$** sucessivos.

A rotina seguinte, que começa na linha 500, faz esse serviço, e retorna um valor numérico **V**. Substitua também a linha 30, como foi indicado:

```
180 PRINT "VALOR ";:GOSUB 500
500 V$=""
510 C$=INKEY$:IF C$="" THEN 510
520 IF ASC(C$)=13 THEN V=VAL(
V$):RETURN
530 PRINT C$:V$=V$+C$:GOTO 510
```

Experimente o programa modificado e veja como ele é capaz de aceitar interrupções também dentro do ciclo de entrada de dados.

PROGRAMAÇÃO DA TECLA <STOP>

Afirmamos um pouco antes que <**STOP**> é uma tecla de interrupção pré-programada, ou seja, que executa uma rotina fixa ao ser acionada. Bem, a afirmação é totalmente verdadeira!

De fato, o usuário também pode programar a tecla <**STOP**>, definindo o que ela deve fazer quando for pressionada. Assim, <**STOP**> equivale às teclas de função, só que não dispõe de espaço próprio na linha reservada de rotulação, para dizer ao usuário o que ela faz. Mas podemos utilizar a própria definição da tecla e usá-la sempre para terminar um programa ou um subprograma. Esta função de "escape" de um nível de programa para outro é a marca registrada dos superprogramas comerciais atualmente disponíveis para computadores pessoais.

O comando que se segue desvia o processamento para a sub-rotina indicada, quando as teclas <**CTRL**> e <**STOP**> são pressionadas simultaneamente (nada acontece se se pressionar apenas a tecla <**STOP**>).

`ON STOP GOSUB`

Para que uma instrução **ON STOP GOSUB** funcione, é necessário ativar a armadilha interna, por meio de um comando **STOP ON**. A partir do momento em que este comando for encontrado, o desvio será realizado para a sub-rotina indicada no **ON STOP GOSUB**, sempre que se pressionar simultaneamente as teclas <**CTRL**> e <**STOP**>. Como nas teclas de função, os comandos **STOP OFF** e **STOP STOP** "desligam", respectivamente, a interrupção e a detecção, para as teclas mencionadas. Experimente alterar o programa, adicionando a opção de interrupção pelas teclas <**CTRL**> <**STOP**>:

```
45 ON STOP GOSUB 400
90 STOP ON
```

A possibilidade de detectar o acionamento da tecla <**STOP**> tem uma outra utilidade: muitas vezes, o programador deseja impedir a interrupção acidental ou intencional de seu programa por um outro usuário, seja para protegê-lo contra olhos curiosos, seja para evitar perdas de dados ou outras condições críticas. O **ON STOP GOSUB** funciona muito bem para esta finalidade, no micro MSX.

SPRITES PARA O TRS-80 (1)

- SÍMBOLOS GRÁFICOS
- UTILIZAÇÃO DE CARACTERES GRÁFICOS EM PROGRAMAS
- AS FUNÇÕES **CHR$** E **STRING$**
- TABELA DE REFERÊNCIA

Os usuários do TRS-80 não precisam ficar frustrados por não disporem dos fabulosos sprites. Como verão aqui, é possível obter animações de excelente qualidade em suas máquinas.

Se você já tentou produzir gráficos animados em um microcomputador da linha TRS-80 usando os comandos **SET** e **RESET**, com certeza chegou à conclusão de que a tarefa é praticamente impossível. Esses comandos servem para acender ou apagar pixels individuais na tabela de baixa resolução, permitindo compor, sem maiores dificuldades, figuras complexas, como um avião ou um foguete. Entretanto, se tentarmos animar o desenho como um todo — deslocá-lo na tela, por exemplo —, obteremos efeitos ridículos, graças à lentidão do interpretador BASIC.

Vimos em artigos anteriores que, no que se refere aos micros pessoais mais modernos — como os das linhas Spectrum, TRS-Color, MSX e Apple —, a solução para esse problema consiste em definir caracteres gráficos por software, armazená-los em uma porção da memória (área UDG no Spectrum, sprites autênticos no MSX e variáveis numéricas ou alfanuméricas nos demais computadores) e animá-los.

Evidentemente, o TRS-80 não conta com recursos tão maravilhosos. Mas há um "jeitinho" que nos permite obter animações fantasticamente rápidas... em BASIC! Veremos como em uma série de artigos. Antes, porém, precisamos entender o funcionamento dos caracteres gráficos pré-definidos do TRS-80.

CARACTERES GRÁFICOS

O que são caracteres gráficos? Como você já sabe, os caracteres que aparecem no vídeo têm códigos numéricos inteiros que, teoricamente, podem variar entre 0 e 255 — cada caractere correspondendo, portanto, a um byte da memória de vídeo. Parte dessa codificação é convencionada internacionalmente: trata-se do chamado código ASCII, que vai de 32 a 126. Os códigos 0 a 31 são normalmente utilizados para funções de controle do vídeo, e dependem do tipo de computador. O mesmo acontece com os códigos 127 a 255. Nesta faixa cada fabricante utiliza os códigos de uma maneira, geralmente para acomodar caracteres gráficos.

O TRS-80 possui 64 caracteres gráficos de teclado, ocupando a faixa de códigos que vai de 128 a 191.

O caractere gráfico com o código 128 é um espaço em branco, semelhante ao que recebe o código 32 no ASCII. Entretanto, não deixa de ser útil, sobretudo para efeito de certas manipulações nas telas gráficas.

Um aspecto bastante interessante dos caracteres gráficos pré-definidos é que eles se comportam exatamente como qualquer outro caractere alfanumérico, para fins de impressão na tela. Assim, eles podem ser usados como argumento de comandos **PRINT**, o que nos permite obter boa velocidade na exibição de gráficos no micro TRS-80.

Ao contrário dos caracteres gráficos pré-definidos do TK-2000, do MSX, do ZX-81 e do Spectrum, os do TRS-80 não podem ser entrados pelo teclado. Assim, para especificá-los dentro de um programa, é necessário utilizar as funções **CHR$** e **STRING$**.

Para imprimir na tela um retângulo totalmente preenchido, escrevemos, por exemplo:

```
PRINT CHR$(191)
```

O programa abaixo mostra a tabela de correspondência entre códigos numéricos e gráficos na tela do TRS-80:

```
10 CLS
20 FOR J=129 TO 191 STEP 9
30 FOR I=J TO J+8
40 PRINT I;CHR$(I);
50 NEXT I:PRINT:NEXT J
```

O programa funciona da seguinte maneira: a linha 10 limpa a tela. O laço que vai da linha 20 à linha 50 (controlado pela variável **J**) produz valores numéricos que variam entre 129 e 191 (a faixa de códigos gráficos, portanto), em incrementos de 9. Assim, o laço das linhas 30 a 50 (controlado pela variável **I**) pode incrementar de 1 em 1 os valores intermediários. Este é um artifício utilizado para produzir na tela uma tabela clara e bem-feita, com nove códigos por linha.

A linha 40 do programa exibe o código numérico (**I**) e o caractere gráfico correspondente (**CHR$(I)**). O **PRINT** entre os dois **NEXT** na linha 50 serve para mudar a linha de impressão.

Se você pretende empregar caracteres gráficos freqüentemente em um programa, convém armazenar os seus códigos em uma variável alfanumérica. Com esse procedimento, você não só poderá utilizá-los com mais facilidade como também não precisará consultar a toda hora a tabela de códigos gráficos, ao desenvolver um programa.

Da mesma maneira, se você quiser utilizar uma cadeia de caracteres gráficos em vários pontos de um programa, armazene-a em uma variável alfanumérica. Para isso, recorra à função **STRING$**.

Execute o programa abaixo e veja como ficam linhas compostas pelo mesmo caractere gráfico.

```
10 CLS
20 INPUT"CODIGO GRAFICO
(129-191)";C
30 INPUT"COMPRIMENTO (1-63)";N
40 S$=STRING$(N,C)
50 PRINT @ 256,S$
60 FOR I=1 TO 500:NEXT
70 GOTO 10
```

A função **STRING$** tem várias utilidades, quando empregada com códigos gráficos. Uma delas, por exemplo, é preencher instantaneamente a tela com um caractere gráfico. Lembre-se de que a tela de textos do TRS-80 tem 1024 caracteres (dezesseis linhas de 64 colunas cada). Se usarmos, por exemplo, uma linha de programa composta de quatro **PRINT STRING$(256,191)**, separados por ponto e vírgula, o resultado será um preenchimento total da tela com o caractere gráfico 191, que corresponde a um bloco inteiramente preenchido (veja a tabela constante do seu Manual de Programação).

Substituindo o segundo argumento da função **STRING$** por outro código qualquer, poderemos usar o mesmo recurso para preencher a tela com blocos gráficos diversos.

PROGRAMAÇÃO DE GRÁFICOS EM 3-D (2)

Até aqui, limitamos nossos wireframes a linhas e formas bidimensionais. Mas, com algumas alterações nas rotinas utilizadas anteriormente, poderemos estruturar imagens tridimensionais.

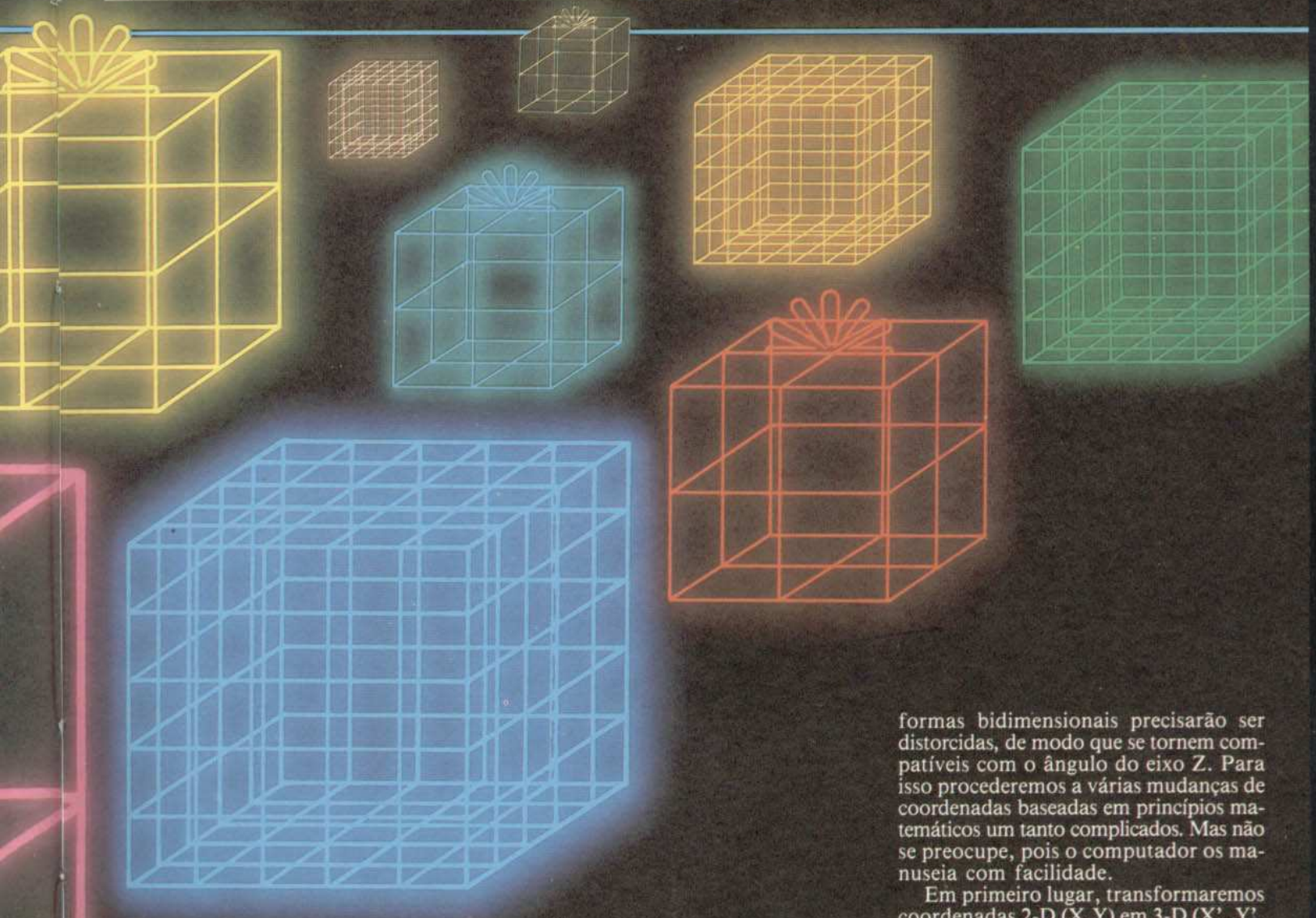
Na teoria, pelo menos, não faz sentido falar na criação de um desenho tridimensional — uma caixa, por exemplo —, num plano bidimensional, seja ele um pedaço de papel ou uma tela de televisão. No entanto, podemos recorrer a técnicas de desenho que dão às figuras a impressão de solidez.

PERSPECTIVA E PROJEÇÃO ISOMÉTRICA

A pintura de uma paisagem, por exemplo, parece ter profundidade e distância quando o pintor faz o desenho em perspectiva. Trata-se de um truque visual baseado no fato de que os objetos parecem ficar menores à medida que se distanciam, e de que linhas paralelas parecem convergir à distância.

A perspectiva não é a única técnica usada nesse tipo de representação. Em desenho técnico é comum ouvirmos falar de projeção isométrica. Como no desenho em perspectiva, as linhas que o observador não vê são desenhadas num certo ângulo. Porém, elas não conver-

- DESENHANDO EM TRÊS DIMENSÕES
- PROJEÇÕES ISOMÉTRICAS
- MUDANÇA DE COORDENADAS
- MONTAGEM DE UM CUBO COM GRADES
- COMO UTILIZAR O PROGRAMA PARA OBTER OUTRAS FORMAS
- UM LAÇO NA CAIXA



gem e os objetos não parecem ficar menores à distância. A vantagem da projeção isométrica é que a direção da linha não afeta a escala e, por isso, pode-se saber a medida exata de cada linha diretamente do desenho.

Embora possamos utilizar o computador para fazer um desenho em perspectiva, é bem mais fácil representar a projeção isométrica. Esta baseia-se simplesmente na criação de um terceiro eixo, além do eixo X (horizontal) e do eixo Y (vertical) já existentes. Este terceiro eixo, o Z, está posicionado num certo ângulo com os outros dois. Qualquer linha desenhada nesse ângulo pode ser entendida como se afastando (ou se aproximando) do observador. O diagrama dos eixos na figura da página 630 ajudará a entender melhor o que dissemos até agora.

ALTERAÇÕES NO PROGRAMA

Recorrendo à projeção isométrica, podemos produzir imagens tridimensionais tipo wireframe a partir de formas bidimensionais, utilizando as rotinas para grades e círculos apresentadas no primeiro artigo da série. Algumas dessas formas bidimensionais precisarão ser distorcidas, de modo que se tornem compatíveis com o ângulo do eixo Z. Para isso procederemos a várias mudanças de coordenadas baseadas em princípios matemáticos um tanto complicados. Mas não se preocupe, pois o computador os manuseia com facilidade.

Em primeiro lugar, transformaremos coordenadas 2-D (X,Y) em 3-D (X', Y', Z'), no plano desejado. Isto significa que devemos especificar o quanto o plano se estende ao longo do eixo Z. Depois transformaremos as coordenadas 3-D (X', Y', Z') num novo conjunto de coordenadas 3-D (Xe,Ye,Ze), baseado na direção e posição da qual olhamos o objeto. Finalmente, voltaremos a transformar (Xe,Ye,Ze) em coordenadas 2-D (Xp,Yp), para que o conjunto possa ser mostrado na tela. Acompanhe os diagramas da página 631; eles mostram a modificação de uma forma de desenho nos seis lados de uma imagem.

Durante a transformação, podemos levar em consideração a perspectiva,

Posso modificar o programa para desenhar imagens diferentes?

A rotina-base de nosso programa é a que compõe a grade. A rotina que ela utiliza para traçar linhas poderia ser modificada para desenhar outras formas. No caso de um triângulo, por exemplo, não haverá maiores problemas, mas, se tivermos que acessar a rotina várias vezes seguidas, tal como no desenho de um tetraedro (figura de quatro lados triangulares), precisaremos de um pouco mais de atenção. No desenho da base, os valores dados às variáveis de transformação serão iguais aos da caixa. Em compensação, a alteração das variáveis de transformação (linhas 270 a 300) para obter o ângulo e posição correta dos outros lados constitui uma tarefa muito difícil.

É mais fácil mudar o programa para que desenhe caixas de diferentes tamanhos ou figuras que não se fecham. Faça, por exemplo, com que o valor de **L** na linha 120 varie com o tempo, alterando, assim, o tamanho da caixa; depois, mude o valor de **N** na mesma linha. Atribuindo valores maiores a **N**, a caixa parecerá sólida. Para evitar que as faces da frente e de trás sejam desenhadas, apague a linha 311 (que faz um laço sobre a caixa) e digite:

```
155 GOTO 220
```

Depois, rode o programa.

mas isso complica bastante o programa. Vamos nos limitar, assim, ao desenho isométrico de uma caixa, deixando a perspectiva para um próximo artigo. Como o primeiro e o último conjuntos de coordenadas são bidimensionais, e todas as transformações são lineares (linhas retas ficam retas e linhas paralelas permanecem paralelas), precisaremos alterar apenas as rotinas de inicialização e de desenho.

Caso tenha armazenado em fita ou disco o programa dado no primeiro artigo, carregue-o novamente e digite as linhas que vêm a seguir. É aconselhável gravar todas as listagens, pois, à medida que desenvolvermos o programa em artigos futuros, voltaremos a chamar algumas rotinas. Se você não tem o programa gravado, digite as linhas 5000 a 5130 do artigo anterior desta série. Elas compõem a rotina que desenha uma grade.

Apague as linhas 150 e 155 antes de digitar as que se seguem, mas não rode o programa ainda.

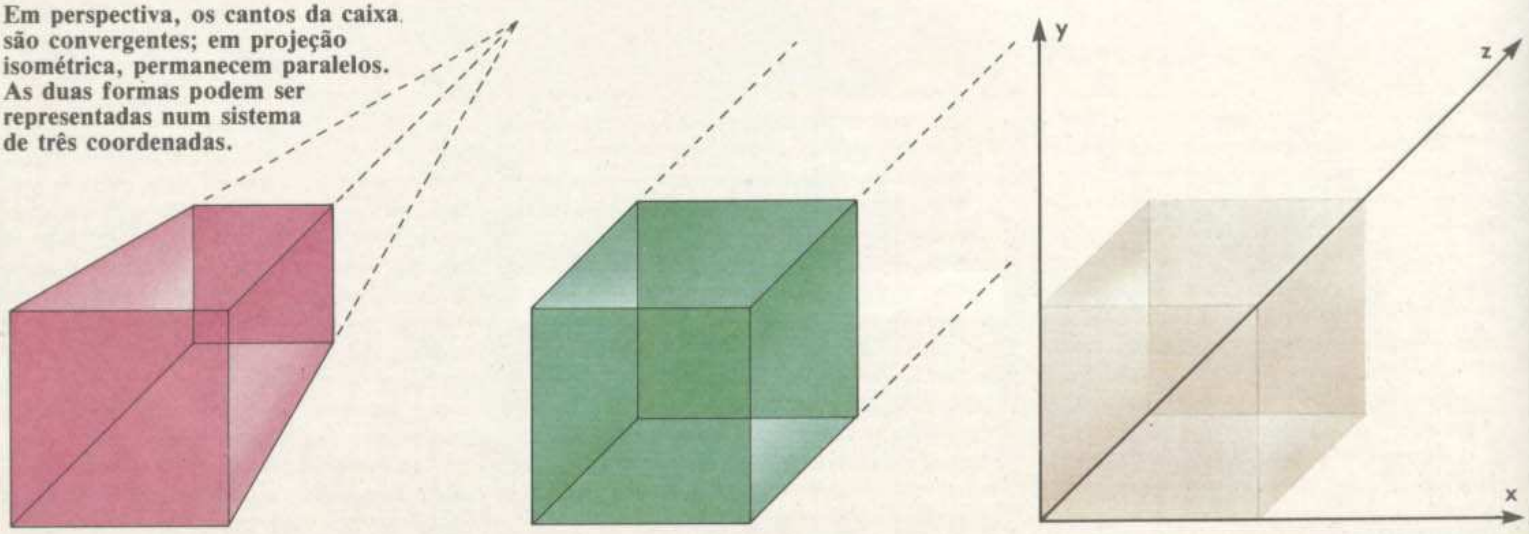
```
9000 LET YX=0: LET XY=0: LET YX
=0: LET YY=1: LET XO=0: LET YO=
0
9010 BORDER 4: PAPER 7: INK 0:
CLS
9070 RETURN
9100 REM MOVER
9110 PLOT XS*XX+XY*YS+XO+127,YX
*XS+YY*YS+YO+76
9120 RETURN
9200 REM DESENHA
9210 DRAW XE*XX+XY*YE+XO+127-PE
EK 23677,YX*XE+YY*YE+YO+76-PEEK
 23678
9220 RETURN
9500 REM LINHA
9510 GOSUB 9100
9520 GOSUB 9200
9550 RETURN
```

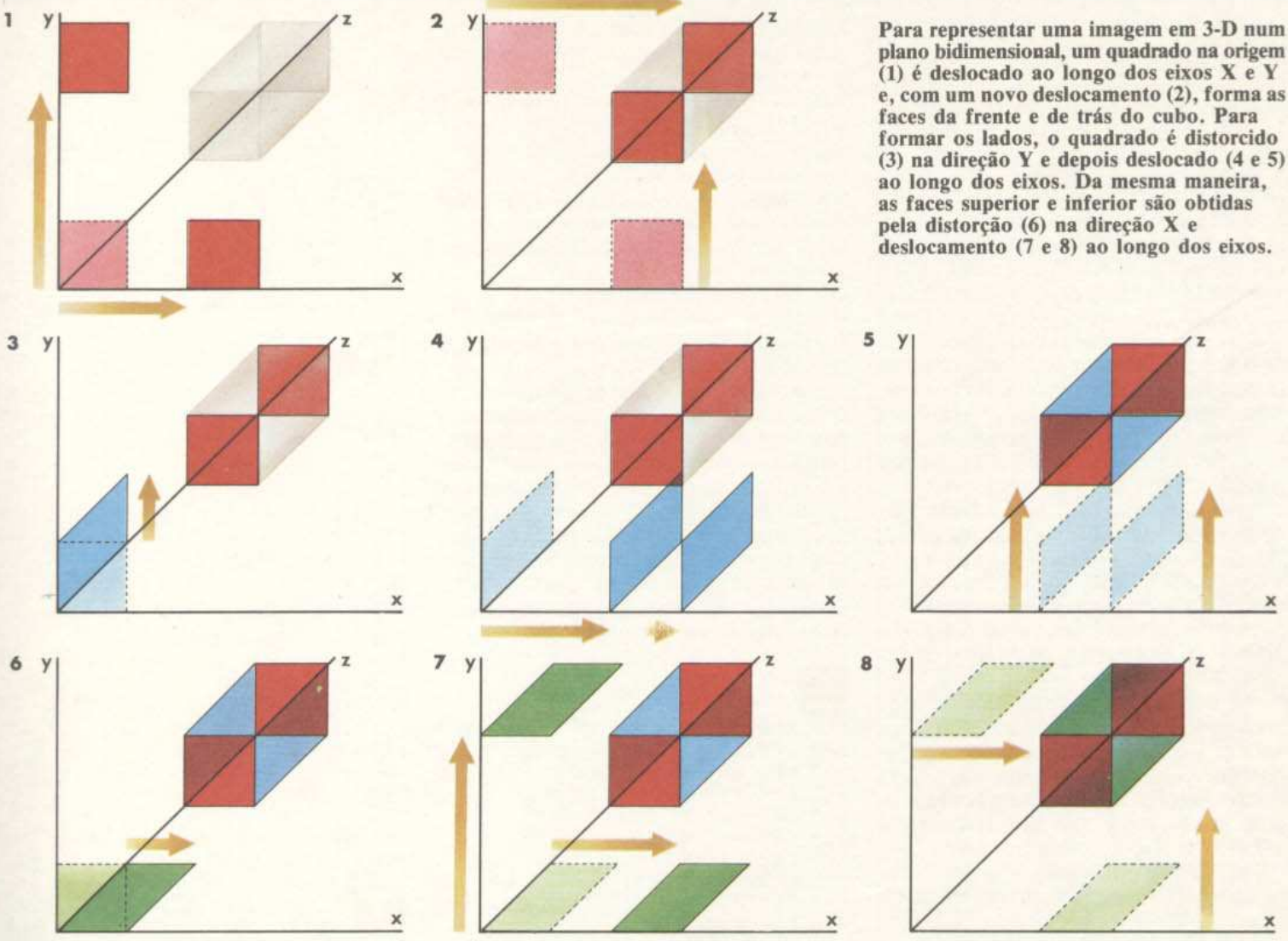
```
9000 CLS
9030 XX=1
9040 YY=1
9070 RETURN
9500 LINE(XS*XX+XY*YS+XO+127,YX
*XS+YY*YS+YO+95)-(XE*XX+XY*YE+X
O+127,YX*XE+YY*YE+YO+95),15
9550 RETURN
```

```
9000 XX = 1:XY = 0:YX = 0:YY =
1:XO = 0:YO = 0
9010  HGR : HCOLOR= 3
9070  RETURN
9100  REM  MOVE
9108 X1 = XS * XX + XY * YS + X
O + 127
9109 Y1 = YX * XS + YY * YS + Y
O + 76
9110  HPLOT X1,Y1
9120  RETURN
9200  REM  DESENHA
9208 X2 = XE * XX + XY * YE + X
O + 127
9209 Y2 = YX * XE + YY * YE + Y
O + 76
9210  HPLOT X1,Y1 TO X2,Y2
9220  RETURN
9500  REM  LINHA
9510  GOSUB 9100
9520  GOSUB 9200
9550  RETURN
```

```
9000 PCLS
9030 XX=1
9040 YY=1
9070 RETURN
9500 LINE(XS*XX+XY*YS+XO+127,YX
*XS+YY*YS+YO+95)-(XE*XX+XY*YE+X
O+127,YX*XE+YY*YE+YO+95),PSET
9550 RETURN
```

Em perspectiva, os cantos da caixa são convergentes; em projeção isométrica, permanecem paralelos. As duas formas podem ser representadas num sistema de três coordenadas.





Para representar uma imagem em 3-D num plano bidimensional, um quadrado na origem (1) é deslocado ao longo dos eixos X e Y e, com um novo deslocamento (2), forma as faces da frente e de trás do cubo. Para formar os lados, o quadrado é distorcido (3) na direção Y e depois deslocado (4 e 5) ao longo dos eixos. Da mesma maneira, as faces superior e inferior são obtidas pela distorção (6) na direção X e deslocamento (7 e 8) ao longo dos eixos.

A rotina das linhas 9000 a 9070 inicializa as variáveis de transformação **(XX,XY,YX,XO,YO)** em valores que fazem a rotina de desenho funcionar como antes. No micro TRS-Color não é necessário inicializar variáveis, mas fazê-lo facilita bastante a compreensão do programa.

A rotina movimenta os eixos de maneira que a origem destes esteja no centro da tela. A origem é deslocada porque fica mais fácil supor que o olho do observador está diretamente acima dela no centro da tela. Isso ficará mais claro no próximo artigo, onde veremos como mudar o ponto de vista e acrescentar perspectiva.

Para observar o efeito de cada variável de transformação e ter uma idéia mais clara do que está acontecendo, modifique as linhas 100 a 180:

```
100 GOSUB 9000
120 CLS
130 LET XA=-40: LET YA=-20:
LET LW=40: LET LH=40: LET NX=5
: LET NY=5
135 GOSUB 5000
140 LET XA=0: LET YA=-20: LET
LW=40: LET LH=40: LET NX=10:
LET NY=10
145 GOSUB 5000
160 INPUT "INTRODUZA XX,XY,YX,
YY,XO,YO",XX,XY,YX,YY,XO,YO
170 GOTO 120
180 STOP
```

```
100 SCREEN 2:COLOR 15,4,4
105 PI=4*ATN(1)
110 GOSUB 9000
120 CLS:SCREEN 2
130 XA=-40:YA=-20:LW=40:LH=40:N
X=10:NY=10
135 GOSUB 5000
140 XA=0:YA=-20:LW=40:LH=40:NX=
10:NY=10
145 GOSUB 5000
150 A$=INKEY$:IF A$="" THEN 150
 ELSE SCREEN 0:CLS
160 INPUT"ENTRE XX,XY,YX,YY,XO,
YO ";XX,XY,YX,YY,XO,YO
170 GOTO 120
```

```
100  GOSUB 9000
120  HOME : HGR : VTAB (24)
130 XA =  - 40:YA =  - 20:LW =
40:LH = 40:NX = 5:NY = 5
135  GOSUB 5000
140 XA = 0:YA =  - 20:LW = 40:L
H = 40:NX = 10:NY = 10
145  GOSUB 5000
160  INPUT "ENTRE XX,XY,YX,YY,X
O,YO ";XX,XY,YX,YY,XO,YO
170  GOTO 120
180  STOP
```

```
100 PMODE 4:SCREEN 1,1
105 PI=4*ATN(1)
110 GOSUB 9000
120 CLS:PCLS:SCREEN 1,1
130 XA=-40:YA=-20:LW=40:LH=40:N
X=5:NY=5
135 GOSUB 5000
```

```
140 XA=0:YA=-20:LW=40:LH=40:NX=
10:NY=10
145 GOSUB 5000
150 A$=INKEY$:IF A$="" THEN 150
160 INPUT"INTRODUZA XX,XY,YX,YY
,XO,YO ";XX,XY,YX,YY,XO,YO
170 GOTO 120
```

Ao rodar o programa, você verá um pequeno quadrado no centro da tela, com uma linha pedindo que forneça novos valores para **XX,XY,YX,YY,XO** e **YO** (no MSX, deve-se apertar qualquer tecla para que a linha seja vista). Estas são as variáveis de transformação.

XX e **YY** ajustam os fatores da escala nas direções horizontal e vertical. Se fornecermos valores negativos a essas variáveis, ocorrerá uma reflexão sobre o respectivo eixo. **XO** e **YO** ajustam a posição da grade na tela. **XY** e **YX** são responsáveis por rotações e distorções no desenho. As variáveis de transformação têm valores iniciais **XX** = 1, **XY** = 0, **YX** = 0, **YY** = 1, **XO** = 0 e **YO** = 0. Para modificar a imagem, precisaremos fornecer diferentes valores a essas variáveis; por isso, a linha 160 contém uma declaração **INPUT**.

Aqui estão alguns conjuntos de valores que você pode experimentar no programa. Forneça-os lado a lado, separados por vírgula, exatamente na ordem em que são pedidos, e tecle <**ENTER**> ou <**RETURN**> (no Spectrum, tecle <**ENTER**> após cada um):

– 1,0,0,1,0,0: causa uma reflexão em relação ao eixo vertical na área central da tela.

0.5,0,0,2,0,0: aumenta a imagem duas vezes na vertical e diminui pela metade na horizontal.

1,0,0,1,100, – 50: move a imagem cem unidades para a direita e cinqüenta unidades para cima (ou para baixo no caso do Spectrum). Usuários do Spectrum podem tentar 1,0,0,1,50, – 40 para uma mudança mais significativa.

0, – 1,1,0,0,0: faz a imagem rodar em 90 graus.

1,0.5,0,1,0,0: distorce a imagem na horizontal.

1,0.5,0.5,1,0,0: distorce a imagem na horizontal e na vertical.

Para prever o efeito de cada conjunto de valores é preciso algum conhecimento sobre aritmética de matrizes. Portanto, embora saibamos como variar a forma do desenho, é um pouco mais difícil determinar a combinação que produzirá uma forma em 3-D. O próximo programa faz isso com as mesmas rotinas que temos usado. Apague as linhas 135, 145 e 175 do programa anterior antes de inserir esta seção:

MICRO DICAS

FORA DA TELA

Alguns computadores não conseguem desenhar pontos que estejam fora da tela. Assim, se fornecermos a **L** um valor muito grande, o programa será interrompido e obteremos uma mensagem de erro. Não é o caso dos micros da linha MSX, os quais desenham qualquer seção de pontos, ignorando aqueles que estiverem fora da tela. Contanto que a imagem esteja inteiramente na tela, os usuários de outros computadores não deverão ter nenhum tipo de problema.

No próximo artigo, veremos como fazer uma checagem no programa, de modo a garantir que somente linhas possíveis sejam desenhadas. Eliminaremos, assim, as interrupções por mensagens de erro.


```
110 GOSUB 9000
120 LET L=108: LET N=4
130 LET DX=.45*L: LET DY=.3*L
140 LET XN=-(L+DX)/2
150 LET YN=-(L+DY)/2
160 LET XO=XN
170 LET YO=YN
180 GOSUB 500
190 LET XO=XN+DX
200 LET YO=YN+DY
210 GOSUB 500
220 LET XY=DX/L: LET XO=XN
230 LET YY=DY/L: LET YO=YN
240 GOSUB 500
250 LET YO=YN+L
260 GOSUB 500
270 LET XX=DX/L: LET XY=0
280 LET YX=DY/L: LET YY=1: LET
YO=YN
290 GOSUB 500
300 LET XO=XN+L
310 GOSUB 500
320 STOP
500 LET XA=0: LET YA=0: LET LW
=L: LET LH=L: LET NX=N: LET NY
=N
510 GOSUB 5000
520 RETURN
```

```
110 GOSUB 9000
120 L=120:N=2
130 DX=.45*L:DY=.3*L
140 XN=-(L+DX)/2
150 YN=-(L+DY)/2
160 XO=XN
170 YO=YN
180 GOSUB 500
190 XO=XN+DX
200 YO=YN+DY
210 GOSUB 500
220 XY=DX/L:XO=XN
230 YY=DY/L:YO=YN
240 GOSUB 500
250 YO=YN+L
260 GOSUB 500
270 XX=DX/L:XY=0
280 YX=DY/L:YY=1:YO=YN
290 GOSUB 500
300 XO=XN+L
310 GOSUB 500
320 GOTO 320
500 XA=0:YA=0:LW=L:LH=L:NX=N:NY
=N
510 GOSUB 5000
520 RETURN
```

```
100  GOSUB 9000
120 L = 108:N = 4
130 DX = .45 * L:DY = .3 * L
140 XN =  - (L + DX) / 2
150 YN =  - (L + DY) / 2
160 XO = XN
170 YO = YN
180  GOSUB 500
190 XO = XN + DX
200 YO = YN + DY
210  GOSUB 500
220 XY = DX / L:XO = XN
230 YY = DY / L:YO = YN
240  GOSUB 500
250 YO = YN + L
260  GOSUB 500
270 XX = DX / L:XY = 0
280 YX = DY / L:YY = 1:YO = YN
290  GOSUB 500
300 XO = XN + L
310  GOSUB 500
320  STOP
500 XA = 0:YA = 0:LW = L:LH = L
:NX = N:NY = N
510  GOSUB 5000
520  RETURN
```

```
100 PMODE4:SCREEN 1,1
110 GOSUB 9000
120 L=120:N=5
130 DX=.45*L:DY=.3*L
140 XN=-(L+DX)/2
150 YN=-(L+DY)/2
160 XO=XN
170 YO=YN
180 GOSUB 500
190 XO=XN+DX
200 YO=YN+DY
210 GOSUB 500
220 XY=DX/L:XO=XN
230 YY=DY/L:YO=YN
240 GOSUB 500
250 YO=YN+L
260 GOSUB 500
270 XX=DX/L:XY=0
280 YX=DY/L:YY=1:YO=YN
290 GOSUB 500
300 XO=XN+L
310 GOSUB 500
320 GOTO 320
```

```
500 XA=0:YA=0:LW=L:LH=L:NX=N:NY
=N
510 GOSUB 5000
520 RETURN
```

Ao rodar o programa, um cubo aparecerá no centro da tela. Seus lados são traçados pela mesma rotina que desenha grades. Esta, com as variáveis adequadas, muda a imagem para a forma desejada. Duas outras variáveis devem ser ajustadas antes que o cubo seja traçado: a variável **L** (que determina qual o tamanho do lado do cubo) e a variável **N** (que determina o número de quadrados na grade).

A variável **DX** especifica o quanto, à direita ou à esquerda, a face de trás está da face da frente; **DY** define o quanto, acima ou abaixo, estas mesmas faces estão uma da outra. **DX** e **DY** fazem com que o cubo pareça ter largura e altura iguais, não adquirindo a forma de um paralelepípedo.

Um outro par de variáveis — **XN** na linha 140 e **YN** na linha 150 — especifica a posição inicial da face da frente do cubo. Esses valores são transferidos para **XO** na linha 160 e **YO** na linha 170. Em seguida, a linha 180 desenha a face de trás. Como ambas as faces são idênticas, para desenhar a face da frente precisamos apenas de um incremento na direção X (linha 190) e outro na direção Y (linha 200).

As linhas 220 e 230 transformam a grade (**XY** e **YX**) e redefinem os valores. Depois, a linha 240 desenha a face de cima. A linha 250 reposiciona verticalmente esse desenho para fazer a face de baixo, que é igual à de cima. De modo similar, as linhas 270 a 310 transformam a grade para obter os lados direito e esquerdo.

Todos os quadrados da grade têm, em cada lado, o mesmo formato e o mesmo tamanho, e todas as linhas paralelas permanecem paralelas, já que não fizemos nenhuma modificação para o desenho em perspectiva, nem adaptamos o programa para mostrar outros ângulos do cubo. Estes serão nossos próximos passos no estudo das técnicas de elaboração de wireframes.

Enquanto não chegamos lá, acrescente um laço ao nosso cubo, transformando-o num pacote de presente. Para isso, mude as linhas 120 e 130 e, depois, adicione as linhas que se seguem. Não se esqueça de gravar o programa!

```
 120 LET L=60: LET N=2
 130 LET DX=.3*L: LET DY=.4*L
 311 GOSUB 9920
9920 FOR M=0 TO PI STEP .01
9930 LET D=COS (6*M)
9940 LET S=D*COS M: LET T=D*SIN
 M
9950 LET S=S*20: LET T=ABS (T*2
0)
9960 PLOT S+127,T+106
9970 NEXT M
9980 RETURN
```

```
120 L=60:N=2
130 DX=.6*L:DY=.5*L
311 GOSUB 10000
10000 REM fita
10010 FOR M=0 TO PI STEP.02
10020 D=COS(6*M)
10030 S=D*COS(M):T=D*SIN(M)
10040 S=S*24:T=-ABS(T*20)
10050 PSET(S+127,T+65),15
10060 NEXT
10130 RETURN
```

```
120 L = 60:N = 2
130 DX = .3 * L:DY = .4 * L
311  GOSUB 9920
9920  FOR M = PI TO 2 * PI STEP
 .01
9930 D =  COS (6 * M)
9940 S = D *  COS (M):T = D *
SIN (M)
9950 S = S * 20:T =  -  ABS (T
* 20)
9960  HPLOT S + 127,T + 46
9970  NEXT M
9980  RETURN
```

```
100 PMODE4:SCREEN 1,1
110 GOSUB 9000
120 L=60:N=2
130 DX=.6*L:DY=.5*L
140 XN=-(L+DX)/2
150 YN=-(L+DY)/2
160 XO=XN
170 YO=YN
180 GOSUB 500
190 XO=XN+DX
200 YO=YN+DY
210 GOSUB 500
220 XY=DX/L:XO=XN
230 YY=DY/L:YO=YN
240 GOSUB 500
250 YO=YN+L
260 GOSUB 500
270 XX=DX/L:XY=0
280 YX=DY/L:YY=1:YO=YN
290 GOSUB 500
300 XO=XN+L
310 GOSUB 500
311 GOSUB 10000
320 GOTO 320
500 XA=0:YA=0:LW=L:LH=L:NX=N:NY
=N
510 GOSUB 5000
520 RETURN
10000 DRAW"BM151,65"
10010 FOR M=0 TO 4*ATN(1) STEP
.02
10020 D=COS(6*M)
10030 S=D*COS(M):T=D*SIN(M)
10040 S=S*24:T=ABS(T*20)
10050 LINE-(S+127,65-T),PSET
10060 NEXT
10130 RETURN
```

GRÁFICOS: BARRAS E SEGMENTOS

Seja qual for a fonte de seus dados — orçamento doméstico, negócios e até hobbies —, organize-os em gráficos de barras ou de segmentos. Será mais fácil entender o que eles têm a dizer.

Já tratamos, em artigo anterior, da organização de dados em gráficos. Veremos agora como usar gráficos de barras (histogramas) ou de segmentos para mostrar dados numéricos numa forma não linear. Esses dois tipos de gráfico, empregados com freqüência na apresentação de dados estatísticos e comerciais, podem ser coloridos, o que os torna muito mais atraentes.

O gráfico de barras é especialmente indicado para a organização de dados que flutuam bastante. O de segmentos é útil sobretudo quando se quer demonstrar como diferentes valores estão relacionados como partes de um todo. Assim, para fazer a escolha mais adequada, leve em conta o tipo específico de dados de que dispõe.

Os programas que se seguem mostram como empregar o microcomputador na elaboração dos dois tipos de gráfico. Devido à falta de comandos gráficos adequados no ZX-81 e no TRS-80, deixamos de apresentar programas para essas máquinas.

MONTAGEM DO GRÁFICO DE BARRAS

A rotina para montar o gráfico de barras é essencialmente a mesma de qualquer outro gráfico. Uma vez de posse dos dados, é necessário entrá-los no computador; em seguida é preciso definir os eixos, desenhar as barras e acrescentar legendas. Como vimos no artigo da página 481, podemos desenhar as barras à medida que entramos os dados ou entrar todos os dados previamente e só depois traçar o gráfico. Uma terceira opção consiste em ler os dados por meio das instruções **READ/DATA**, alterando-os sempre que se quiser um gráfico diferente. Seja qual for o método escolhido, convém armazenar os dados numa matriz numérica, de modo que o micro possa identificar as coordenadas de cada barra.

LEITURA DOS DADOS

Digite as linhas seguintes para o computador ler os dados. Ao executá-las, não verá nada na tela, mas é bom rodar cada parte do programa para evitar erros. (No Apple ou no TK-2000, para ter de volta a tela de texto, digite **TEXT** e tecle **RETURN**.) De qualquer forma, você pode verificar se aconteceu algo digitando **PRINT A(3)**, por exemplo. Note que os dados já foram armazenados na memória do micro.

```
 10 LET n=12
 20 DIM a(n)
 70 FOR t=1 TO n
 80 READ a(t)
 90 NEXT t
3010 DATA 3,6,5,9,6,3,6,8,3,5,9
,4
```

```
5 N=12
20 DIM A(N)
60 PMODE 3,1:PCLS:SCREEN 1,0
70 FOR T=1 TO N
80 READ A(T)
90 NEXT
3010 DATA 1,5,3,8,6,2,8,5,2,9,5
,10
```

```
10 N=12
20 DIM A(N)
60 COLOR 10,5,5 :SCREEN 2
70 FOR T=1 TO N
80 READ A(T)
90 NEXT
3010 DATA 1,5,6,4,7,9,4,3,8,3,9
,8
```

```
10 N = 12
20   DIM A(N)
60   HOME : HGR
70   FOR T = 1 TO N
80   READ A(T)
90   NEXT
3010  DATA    2,5,3,8,6,2,0,9,7,
8,5,4
```

Neste módulo, o programa seleciona o modo gráfico, determina o número de barras a serem traçadas — 12 (linha 10, ou 5, no TRS-Color) — e dimensiona a matriz para esse número (linha 20).

Pode-se usar um número maior para traçar mais barras. Nesse caso, é necessário acrescentar dados à linha 3010, para que haja uma equivalência entre o número de barras e o de dados. É interessante ter um número maior de dados quando se está testando o programa, de modo que se possa aumentar ou diminuir o número de barras sem risco de que ocorra um erro do tipo "falta de dados" ou *end of data*.

- COMO FAZER UM HISTOGRAMA
- GRÁFICO DE BARRAS TRIDIMENSIONAL
- PROGRAME UM GRÁFICO DE SEGMENTOS

A ESCALA DOS EIXOS

O computador conhece agora o valor absoluto de cada barra. Precisamos informá-lo sobre como fazer a escala dos dados, de maneira que cada conjunto deles preencha a tela de forma adequada. Caso contrário, pode-se ter números tão pequenos que praticamente não sejam vistos ou, no outro extremo, valores que caiam fora do intervalo possível da tela. Para fazer a escala, digite as próximas linhas. Novamente nada vai aparecer na tela, mas execute o programa como um teste.

```
  30 LET dx=239/n
  40 READ dy
3000 DATA 18
```

```
30 DX=164/N
40 READ DY
3000 DATA 1
```

```
30 DX=164/N
40 READ DY
3000 DATA 1
```

```
30 DX =  INT (170 / N)
40  READ DY
3000  DATA   15
```

Esta parte do programa calcula a escala do eixo X dividindo o comprimento disponível pelo número de barras (linha 30). O valor máximo de barras varia de micro para micro, mas, conhecendo o tamanho de sua tela gráfica e mudando os valores de N, você identificará facilmente o valor mais adequado para um gráfico legível.

Os valores do eixo Y são multiplicados por um fator para que tomem o tamanho mais adequado. Esse valor não

é encontrado automaticamente, mas lido (linha 40) de uma declaração **DATA** (linha 3000). Para cada novo conjunto de valores, avalie qual o melhor fator a ser usado e coloque-o na linha 3000. Nesse ponto, você pode montar uma rotina para dar entrada a tal valor a partir da linha 40 (usando **INPUT**). Se preferir agir assim, não se esqueça de apagar a linha 3000.

O restante do programa é composto de duas rotinas: uma que traça os eixos e outra que desenha as barras. Digite mais estas linhas, mas não as execute, uma vez que as rotinas chamadas ainda não se encontram na memória e um erro será detectado.

```
100 GOSUB 1000
140 GOSUB 2000
160 STOP
```

As linhas 100 e 140 chamam sub-rotinas que traçam os eixos e barras.

```
100 GOSUB 1000
110 FOR T=1 TO N
130 X=(T-1)*DX+18:Y=188-16*A(T)
*DY
140 GOSUB 2000
150 NEXT
160 GOTO 160
```

A linha 100 desvia o programa para a linha que desenha os eixos. A 110 abre um laço que marca o número de barras a serem desenhadas. A cada volta, a linha 130 ajusta a coordenada de **X** — multiplicando-a por **DX** e somando 18 para deixar uma margem à esquerda — e a coordenada de Y — subtraindo do total de linhas disponíveis o valor lido **(A(T))** multiplicado pelo fator **DY** (dado) e pela constante 16. A linha 140 desvia o programa para a rotina que desenha as barras e a 150 fecha o laço, chamando o próximo valor.

```
100  GOSUB 1000
110  FOR T = 1 TO N
130 X = (T - 1) * DX + 8 + A:Y
= 150 - A(T) * DY
140  GOSUB 2000:A = A + 3
150  NEXT
160  END
```

A linha 100 chama a sub-rotina que traça os eixos. A linha 110 abre um laço que determina o número de barras a serem desenhadas. Em seguida, a linha 130 determina a coordenada **X** para o ponto inicial do traçado da barra, multiplicando **T-1** por **DX** e somando **8** — para garantir uma margem à esquerda — e **A** (que provoca um avanço de uma barra em relação à outra — para evitar que as barras fiquem coladas umas às outras). A coordenada **Y** (altura da barra) é calculada subtraindo-se do total de linhas disponíveis o valor lido **(A(T))** pelo fator **DY** (que você determina). A linha 140 chama a sub-rotina que desenha as barras e a seguir incrementa o valor de **A**. A linha 150 fecha o laço, chamando o próximo valor a ser desenhado.

O DESENHO DAS BARRAS

Agora, digite as rotinas que traçam os eixos e desenham as barras:

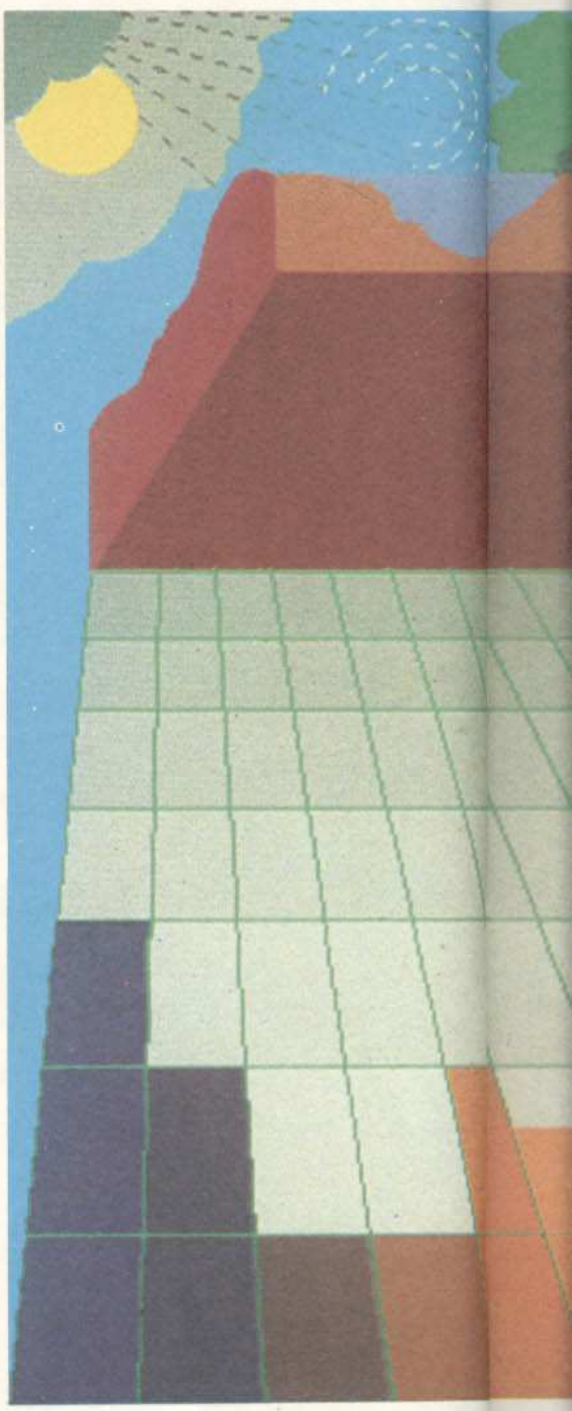
```
1000 PLOT 16,0: DRAW 0,170
1020 PLOT 16,0: DRAW 235,0
1030 RETURN
2000 FOR a=1 TO n
2010 LET s=16+(a-1)*dx
2020 FOR t=s TO s+dx-4
2030 PLOT t,0: DRAW 0,a(a)*dy
2040 NEXT t
2100 NEXT a
2110 RETURN
```

As linhas 1000 e 1020 desenham os eixos, deixando uma margem de dezesseis unidades à esquerda do eixo **Y**. Para desenhar as barras, a linha 2000 toma um valor de cada vez, a linha 2010 coloca o valor na escala **(*dx)** e a linha 2020 monta um laço para desenhar as linhas verticais que formarão a barra. O "-4" no fim da linha provoca um pequeno espaçamento entre as barras. As linhas restantes desenham as linhas que completam a barra e fecham os respectivos laços.

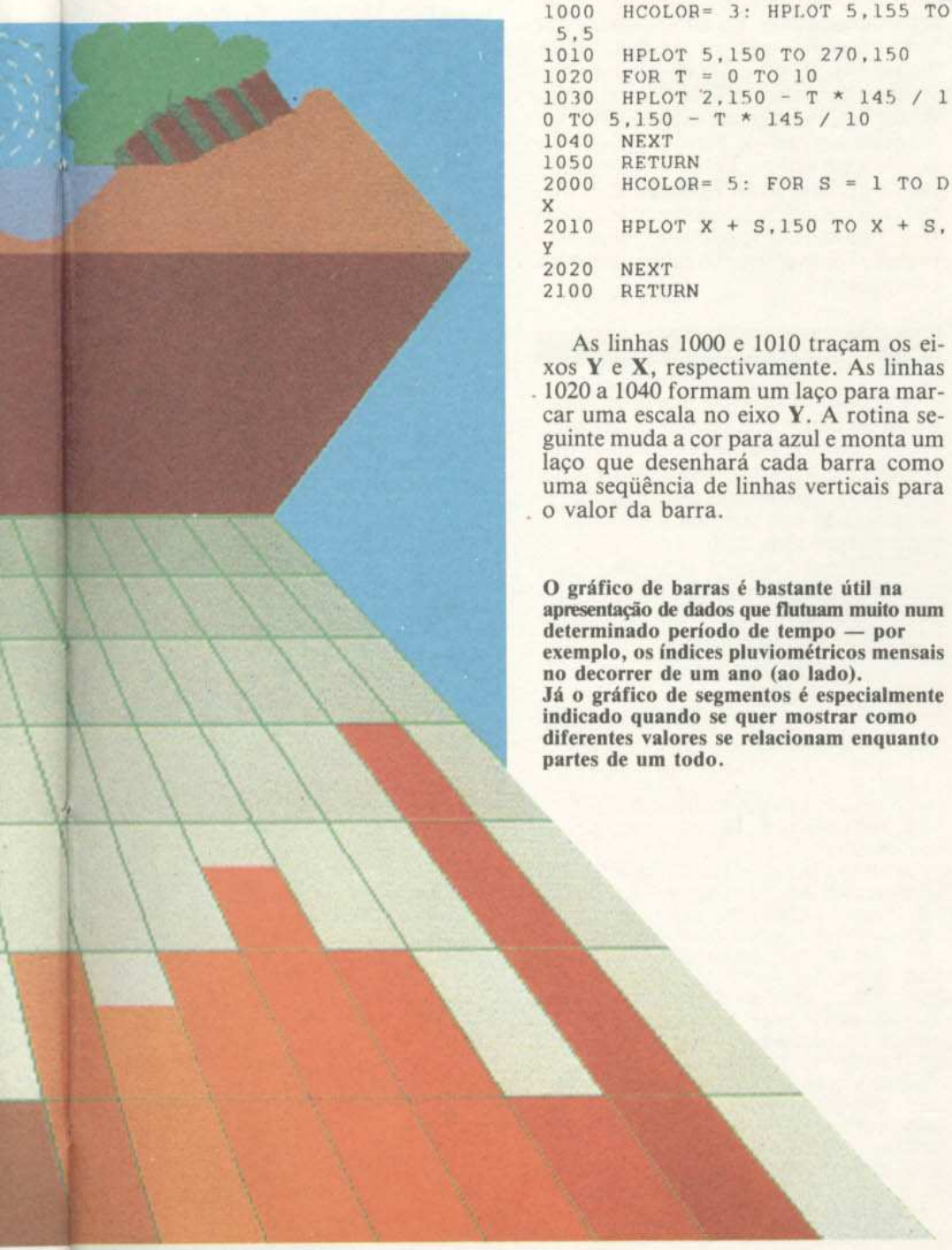
```
1000 LINE(0,25)-(0,191),PSET
1020 COLOR 3
1030 LINE -(255,191),PSET
1050 COLOR 2:FOR T=0 TO 10
1060 LINE(0,191-T*166/10)-(3,19
1-T*166/10),PSET
1070 NEXT
1080 RETURN
2000 COLOR 2
2090 LINE(X,190)-(X+DX,Y),PSET,
BF
2100 RETURN
```

A primeira linha desta parte do programa desenha o eixo **Y**, de cima até embaixo, na tela. A linha 1030, por sua vez, se incumbe de traçar o eixo **X**, da esquerda para a direita. Observe que o sinal "–" faz com que seja traçada uma linha da última posição do cursor até a posição indicada entre parênteses. A linha 1050 muda a cor e abre um laço que executa a marcação de uma escala no eixo **Y**.

A rotina seguinte (linhas 2000 a 2100) desenha um retângulo para cada barra e o preenche de amarelo.



```
1000 LINE(8,25)-(8,191),6
1030 LINE -(255,191),15
1050 FOR T=0 TO 10
1060 LINE(8,191-T*166/10)-(10,1
91-T*166/10),15
1070 NEXT
1080 RETURN
2000 COLOR 10,5,5
2090 LINE(X,190)-(X+DX,Y),2,BF
2100 RETURN
```



A linha 1000 traça o eixo **Y**, deixando uma margem de oito unidades. A linha 1030 traça o eixo **X**. O laço das linhas 1050 a 1070 marca uma escala no eixo dos **Y** para tornar mais fácil a leitura dos dados.

A rotina das linhas 2000 a 2100 tem como tarefa mudar a cor de frente para amarelo e desenhar um retângulo colorido para cada valor.

```
1000   HCOLOR= 3: HPLOT 5,155 TO
 5,5
1010   HPLOT 5,150 TO 270,150
1020   FOR T = 0 TO 10
1030   HPLOT 2,150 - T * 145 / 1
0 TO 5,150 - T * 145 / 10
1040   NEXT
1050   RETURN
2000   HCOLOR= 5: FOR S = 1 TO D
X
2010   HPLOT X + S,150 TO X + S,
Y
2020   NEXT
2100   RETURN
```

As linhas 1000 e 1010 traçam os eixos **Y** e **X**, respectivamente. As linhas 1020 a 1040 formam um laço para marcar uma escala no eixo **Y**. A rotina seguinte muda a cor para azul e monta um laço que desenhará cada barra como uma seqüência de linhas verticais para o valor da barra.

O gráfico de barras é bastante útil na apresentação de dados que flutuam muito num determinado período de tempo — por exemplo, os índices pluviométricos mensais no decorrer de um ano (ao lado).
Já o gráfico de segmentos é especialmente indicado quando se quer mostrar como diferentes valores se relacionam enquanto partes de um todo.

TERCEIRA DIMENSÃO

Ao executar esse programa, você verá que a aparência do gráfico de barras é muito menos acadêmica e mais atraente que a dos gráficos lineares, embora os dois tipos sejam construídos com o mesmo número de dados. Mas pode-se melhorar ainda mais a apresentação dos dados com a adição de um efeito tridimensional às barras, o que dará ao desenho um aspecto sólido e natural e um destaque especial às cores.

Antes de entrar o novo programa, grave o que você já tem na memória do computador (para uma posterior comparação). Depois, sem digitar **NEW**, tecle as alterações que se seguem. Se você quiser usar os comandos de edição do seu micro para reduzir o trabalho de digitação, fique atento para não deixar escapar pequenas diferenças que possam existir entre as linhas.

Acrescente ao seu programa apenas algumas linhas. Veja adiante.

```
130 X=(T-1)*D2+18:Y=188-16*A(T)
*DY
1000 LINE (0,191)-(0,25),PSET:L
INE-(DX*.6,26-DZ),PSET
1020 PAINT (2,100),4:COLOR 3
1030 LINE -(255-DX*.6,191),PSET
:LINE-(255,191-DZ),PSET
1060 LINE (0,191-T*166/10)-(DX*
.6,191-T*166/10-DZ),PSET
2000 COLOR 4
2090 LINE(X,188)-(X+DX,Y),PSET,
BF
```

```
50 DZ=50/(N+1)
115 IF A(T)=0 THEN 160
120 D2=DX*1.4:IFDX>40THEND2=DX+
15
130 X=(T-1)*D2+18:Y=188-16*A(T)
*DY
1000 LINE(8,191)-(8,25),6:LINE
-(8+DX*.6,26-DZ),6
1010 LINE -(8+DX*.6,191-DZ),6:L
INE -(8,191),6
1020 PAINT (9,180),6
1030 LINE (8,191)-(255-DX*.6,19
1),15:LINE -(255,191-DZ),15
1040 LINE -(8+DX*.6,191-DZ),15:
LINE -(8,191),15:PAINT (100,190
),15
1050 FOR T=0 TO 10
1060 LINE(8,191-T*166/10)-(8+DX
*.6,191-T*166/10-DZ),15
```

```
50 DZ = 50 / (N + 1): IF DZ > 5
 THEN DZ = 5
1000  HCOLOR= 3: HPLOT 5,155 TO
 5,5 TO DX * .7,5 - DZ TO DX *
.7,150 - DZ TO 5,150
1010  HPLOT 5,150 TO 270 - DX *
 .7,150 TO 270,150 - DZ TO DX *
 .7,150 - DZ
```

Não execute ainda o programa, pois, como ele não está completo, você obterá apenas uma mensagem de erro. Digite mais estas linhas:

```
1010 DRAW 4,4: DRAW 0,-170
2050 PLOT 16+(a-1)*dx,a(a)*dy
2060 DRAW 4,4
2070 DRAW dx-4,0
2075 DRAW -4,-4: DRAW 4,4
2080 DRAW 0,-a(a)*dy
2090 DRAW -4,-4
```

```
50 DZ=50/(N+1)
115 IF A(T)=0 THEN 160
120 D2=DX*1.4:IF DX>40 THEN D2=
DX+15
1010 LINE -(DX*.6,191-DZ),PSET:
LINE-(0,191),PSET
1040 LINE -(DX*.6,191-DZ),PSET:
LINE -(0,191),PSET:PAINT(127,18
9),3
2010 LINE (X+DX,188)-(X+DX*1.6,
188-DZ),PSET:LINE-(X+DX*1.6,Y-D
Z),PSET
2020 LINE -(X+DX,Y),PSET:LINE -
(X+DX,188),PSET
2030 PAINT (X+DX+2,185),4
2040 COLOR 3
2050 LINE (X,Y)-(X+DX*.6,Y-DZ),
PSET:LINE-(X+DX*1.6,Y-DZ),PSET
2060 LINE -(X+DX,Y),PSET:LINE -
(X,Y),PSET
2070 PAINT (X+DX/2,Y-1),3
2080 COLOR 2
```

```
2000 COLOR12,5,5
2010 LINE (X+DX,188)-(X+DX*1.6,
188-DZ):LINE -(X+DX*1.6,Y-DZ)
2020 LINE -(X+DX,Y):LINE -(X+DX
,188)
2030 PAINT (X+DX+2,185)
2050 LINE (X,Y)-(X+DX*.6,Y-DZ):
LINE -(X+DX*1.6,Y-DZ)
2060 LINE -(X+DX,Y):LINE -(X,Y)
2070 PAINT (X+DX/2,Y-1)
2090 LINE (X,188)-(X+DX,Y),3,BF
```

```
2000  FOR S = 1 TO DX
2010  HCOLOR= 3: HPLOT X + S +
DX * .7,Y - DZ TO X + S + DX *
.7,150 - DZ: HCOLOR= 6
2020  HPLOT X + S,150 TO X + S,
Y
2030  HPLOT  TO X + S + DX * .7
,Y - DZ: REM  TO X + 1 + DX * .
7,Y - DZ TO X + 1,Y
2040  NEXT
2050  HPLOT  TO X + S + DX * .7
,150 - DZ
```

Você tem agora uma belíssima imagem tridimensional na tela. Como exercício, modifique os valores dos comandos de cores para selecionar aquelas que mais lhe agradam. Faça um registro das linhas que precisam ser mudadas para alterar a escala do gráfico, de modo que você tenha uma referência à mão quando for usar o programa mais tarde. Uma maneira simples de fazer esse registro consiste em utilizar linhas no programa com a declaração **REM**. Ela indica que seu conteúdo é exclusivamente informativo, não devendo, portanto, ser interpretado como instrução ou comando ao computador.

GRÁFICO DE SEGMENTOS

Outra forma interessante de apresentação de dados é o gráfico de segmentos — um tipo de gráfico circular que requer apenas uma coordenada (coordenada polar) para cada informação, representando por meio de um ângulo o tamanho de cada dado.

Estruturalmente, o programa para desenhar um gráfico de segmentos é simples. Para ver como funciona, entre e execute as linhas seguintes.

```
 10 DIM a(12): LET n=0
 20 CLS
 40 PRINT AT 5,14;"MENU"
 50 PRINT AT 8,10;"1- Introduz
 ir dados"
 60 PRINT AT 10,10;"2-Grafico"
 70 PRINT AT 12,10;"3- Saida"
 80 LET a$=INKEY$: IF a$<"1"
 OR a$>"3" THEN  GOTO 80
 90 GOSUB VAL a$*200
100 GOTO 20
200 CLS : LET n=1
210 PRINT "Item No. ";n,:
INPUT  LINE a$: PRINT a$: IF
a$="" THEN  RETURN
220 LET a(n)=VAL a$: LET n=n+1
230 IF n<13 THEN  GOTO 210
240 LET n=n-1: RETURN
400 IF n=0 THEN  RETURN
410 CLS : LET tt=0: FOR t=1 TO
n: LET tt=tt+a(t): NEXT t
420 LET f=(2*PI)/tt
430 CIRCLE 127,86,60
440 LET a=0: FOR k=1 TO n
450 LET m=a+a(k)*f
460 PLOT 127,86: DRAW 60*SIN m
,60*COS m
470 LET a=m
480 NEXT k
490 PAUSE 0: RETURN
```

```
10 DIM A(31),P(31)
15 PMODE 3,1
20 CLS
40 PRINT @45,"MENU"
50 PRINT @169,"1- INTRODUZIR DA
DOS"
60 PRINT @201,"2- GRAFICO"
70 PRINT @233,"3- SAIDA"
80 A$=INKEY$:IF A$<"1" OR A$>"3
" THEN 80
90 ON VAL(A$) GOSUB 200,400,600
100 GOTO 20
200 CLS:N=0
210 PRINT"ITEM NO. ";N+1;:INPUT
 A$:IF A$="" THEN RETURN
220 N=N+1:A(N)=VAL(A$)
230 IF N<31 THEN 210
240 RETURN
400 IF N=0 THEN RETURN
410 PCLS:SCREEN 1,0:TT=0:FOR T=
1 TO N:TT=TT+A(T):NEXT
420 FOR T=1 TO N:P(T)=A(T)*810*
ATN(1)/TT:NEXT
430 J=0:P(0)=-1
440 FOR T=1 TO  N
450 IF T=N AND N-3*INT(N/3)=1 T
HEN COLOR 4 ELSE COLOR T-3*INT(
T/3)+2
460 FOR K=1 TO P(T)
470 X=X+.01:Y=Y+.01
480 LINE (127,95)-(127+60*SIN(X
),95-60*COS(Y)),PSET
490 NEXT K,T
500 IF INKEY$="" THEN 500
510 RETURN
600 CLS
```

```
10 DIM A(31),P(31)
20 CLS
30 LOCATE 18:PRINT"MENU"
40 LOCATE 10,8:PRINT"1 - Entrar
 dados"
50 LOCATE 10,11:PRINT"2 - Ver g
ráfico"
60 LOCATE 10,14:PRINT"3 - Fim"
80 A$=INKEY$:IF A$<"1" OR A$>"3
" THEN 80
90 ON VAL(A$) GOSUB 200,400,600
100 GOTO 20
200 CLS:N=0
210 PRINT"Item nº ";N+1;:INPUT
A(N+1):IF A(N+1)=0 THEN RETURN
220 N=N+1
230 IF N<31 THEN 210
240 RETURN
400 IF N=0 THEN RETURN
410 COLOR 15,15,15:SCREEN 2:TT=
0:FOR T=1 TO N:TT=TT+A(T):NEXT
420 FOR T=1 TO N:P(T)=A(T)*810*
```

```
ATN(1)/TT:NEXT
430 J=1:P(0)=-1
440 FOR T=1 TO N
450 IF T=NTHEN COLOR 13 ELSE J=
J+1:COLOR J:IF J=12 THEN J=0
460 FOR K=1 TO P(T)
470 X=X+.01:Y=Y+.01
480 LINE (127,95)-(127+60*SIN(X
),95-60*COS(Y))
490 NEXT:NEXT
500 IF INKEY$=""THEN500
510 RETURN
600 CLS
3000 DATA 8,9,10
```

```
10  DIM A(31),P(31)
20  HOME
40  HTAB 18: PRINT "MENU"
50  HTAB 10: VTAB 10: PRINT "1
- ENTRAR DADOS"
60  HTAB 10: VTAB 13: PRINT "2
- VER GRAFICO"
70  HTAB 10: VTAB 16: PRINT "3
- FIM "
80  GET A$: IF A$ < "1" OR A$ >
 "3" THEN 80
90  ON  VAL (A$) GOSUB 200,400,
600
100  GOTO 20
200  HOME :N = 0
210  PRINT "ITEM NO. ";N + 1;:
INPUT A$: IF A$ = "" THEN  RETU
RN
220 N = N + 1:A(N) =  VAL (A$)
230  IF N < 31 THEN 210
240  RETURN
400  IF N = 0 THEN  RETURN
410  HGR :TT = 0: FOR T = 1 TO
N:TT = TT + A(T): NEXT
420  FOR T = 1 TO N:P(T) = A(T)
 * 810 *  ATN (1) / TT: NEXT
430 J = 0:P(0) =  - 1
440  FOR T = 1 TO N
450 J = J + 1: IF J = 4 THEN J
= 1
455  HCOLOR= J: IF T = N THEN
HCOLOR= 5
460  FOR K = 1 TO P(T)
470 X = X + .01:Y = Y +  01
480  HPLOT 127,95 TO 127 + 60 *
  SIN (X),95 - 60 *  COS (Y)
490  NEXT : NEXT
500  GET A$
510  TEXT : RETURN
600  HOME
```

Ao executar o programa, um menu será apresentado na tela, oferecendo três opções: entrar dados, ver o gráfico ou terminar o programa. A linha 80 faz com que o computador espere até que uma tecla seja pressionada. Você pode pressionar qualquer uma, mas, devido à condição imposta pela declaração **IF...THEN** da mesma linha, somente as teclas 1, 2 ou 3 serão aceitas.

Uma condição parecida faz com que o programa volte ao menu, caso você tecle 2 sem ter nenhum dado na memória. A tecla 3, por sua vez, termina o programa. É necessário o comando **RUN** para reiniciá-lo.

No começo da execução, a escolha óbvia é a tecla 1, que lhe possibilita entrar alguns dados. A linha 90 indica para qual ponto do programa a execução será desviada; no caso, ela irá para a linha 200. Cabe a esta linha estabelecer uma variável (**N**) para o número de dados. A entrada de dados se dá na linha 210. Eles permanecerão armazenados numa matriz (**A(.)**), na linha 220. Esta matriz foi dimensionada anteriormente, na linha 10.

Para entrar os dados, digite o número seguido de **ENTER** ou **RETURN**. A linha 230 verifica se o espaço reservado para os dados já não está preenchido. Se ainda há lugar, o programa volta à linha 210 para mais uma entrada. Não é obrigatório ocupar todo o espaço reservado para dados. Quando você quiser encerrar a entrada de números, simplesmente tecle as instruções **ENTER** ou **RETURN** sem ter digitado nenhum número. O programa volta, então, a apresentar o menu principal, por ação da linha 210. Se, por outro lado, você preencheu todo o espaço disponível, a linha 240 encerrará automaticamente a entrada de dados.

CONSTRUÇÃO DO GRÁFICO

Se, agora, você pressionar 2, o programa será desviado para a linha 400, encarregada de iniciar a rotina que desenha o gráfico. A linha 410 lê todos os valores armazenados e calcula o seu total. O restante da rotina calcula uma escala para os dados e desenha os gráficos. Como isso é feito de maneira um pouco diferente para cada computador, a descrição será individual. Para entender melhor o funcionamento das funções circulares no computador, veja o artigo da página 334.

A linha 420 divide o círculo completo (um ângulo de 360 graus ou 2 pi radianos) pelo valor total dos dados para obter um fator de escala. A linha 430 encarrega-se de desenhar o círculo (de raio 60). As linhas 440 e 450 passam por todos os valores, calculando um subtotal para cada um e multiplicando este valor pelo fator de escala (f). O valor encontrado (m) é o valor de pontos na circunferência do círculo. Este é dividido por raios desenhados pela linha 460.

MICRO DICAS

APERFEIÇOE OS PROGRAMAS

Agora que você já completou e testou os programas deste artigo, que tal tentar melhorar a aparência dos gráficos desenhados na tela? Aqui vão algumas sugestões.

Um recurso que enriquece muito o aspecto final de um gráfico — e que aumenta a sua utilidade, sobretudo quando se trata de ilustrar apresentações ou artigos — é a *titulação*. Você pode acrescentá-la ao programa, por meio de linhas **DATA** ou **INPUT**, sob três formas:

- Rótulos para os eixos X e Y: identificam o valor ou nome das divisões dos eixos (exemplo: os nomes dos meses em um gráfico de barras).
- Legendas: indicam o significado de cada barra ou "fatia" do gráfico de segmentos. No gráfico de barras, podem ser colocadas na parte de baixo ou no canto superior direito; no de segmentos, devem aparecer ao lado de cada "fatia".
- Títulos propriamente ditos: especificam o assunto do gráfico — por exemplo, "VENDAS DA COMPANHIA XYZ. Resultados de 1985". Costumam ter de uma a três linhas, geralmente centradas e posicionadas na parte superior ou inferior da tela.

Também é interessante salientar uma barra ou segmento, deslocando-o ligeiramente de sua posição — como se uma fatia do gráfico de segmentos fosse retirada. Este recurso, conhecido como *explosão*, pode ser facilmente programado nos micros que possuem o comando **CIRCLE**.

A linha 410 totaliza os dados da memória. Em seguida, a linha 420 passa por todos os dados, calculando um subtotal para cada um e dividindo este subtotal pelo total. Esses valores, já em escala, são armazenados na matriz **P(.)**, que foi dimensionada na linha 10. A linha 430 define variáveis que serão utilizadas na seqüência de cores. O laço **FOR...NEXT** iniciado na linha 440 escolhe a cor para cada um dos segmentos do círculo, e o que começa na linha 460 traça os raios que preencherão os segmentos com a cor predeterminada. A linha 500 faz com que o gráfico fique visível até que alguma tecla seja pressionada, momento em que o menu novamente é apresentado.

GRÁFICOS DA ROM NO SPECTRUM (1)

- O QUE SÃO CARACTERES GRÁFICOS PRÉ-PROGRAMADOS
- COMO ENTRAR CARACTERES GRÁFICOS PELO TECLADO DO MICRO

Os computadores da linha ZX Spectrum dispõem de caracteres gráficos que podem ser usados dentro de um programa ou entrados diretamente pelo teclado. Aprenda a utilizá-los.

Os micros da linha Spectrum (como o TK-90X) são muito versáteis do ponto de vista da programação gráfica. Em artigos anteriores, vimos como a tela gráfica pode ser programada por meio de instruções extremamente poderosas, em alta resolução, tais como **LINE**, **CIRCLE**, **PAINT** etc.

Entretanto, o Spectrum tem um recurso gráfico adicional que é pouco explorado: os *caracteres gráficos* de baixa resolução. Estes podem tanto ser entrados diretamente pelo teclado quanto inseridos por intermédio da função **CHR$**, do BASIC.

O que são caracteres gráficos? Como você já sabe, os caracteres que aparecem no vídeo têm códigos numéricos inteiros, que, teoricamente, podem variar entre 0 e 255. Cada caractere corresponde, portanto, a um byte da memória de vídeo. Parte dessa codificação é convencionada internacionalmente pelo chamado código ASCII, que vai de 32 a 126. Os códigos de 0 a 31 são normalmente utilizados para funções de controle do vídeo, e dependem do tipo de computador. O mesmo ocorre com os códigos de 127 a 255. Nesta faixa, cada fabricante utilizou os códigos de uma maneira, geralmente para acomodar caracteres gráficos.

O Spectrum possui apenas dezesseis caracteres gráficos de teclado, os quais podem ser descritos como combinações dos quatro quadradinhos que formam um caractere (pixels maiores dos que os utilizados na programação gráfica de alta resolução). A matriz básica de um caractere gráfico é:

"Colorindo" individualmente os quadradinhos, em todas as combinações possíveis, obtemos os diferentes gráficos. Por exemplo:

O caractere gráfico com o código 128 é um espaço em branco, o que não o torna menos necessário. Como os demais caracteres, ele pode ser invertido e colorido com **INK** e **PAPER**.

GRÁFICOS DA ROM

Os caracteres gráficos são também conhecidos como *gráficos da ROM*, pois já vêm pré-programados. No Spectrum, eles ocupam a faixa da tabela de caracteres que vai de 128 a 143 e, em geral, são utilizados para a elaboração de desenhos em baixa resolução ou para a composição de bordas e outros tipos de linhas retas. Na formatação de tabelas ou formulários de entrada — seu emprego mais freqüente — apresentam a vantagem de não complicar a programação, misturando tela gráfica com texto. Possibilitam, dessa maneira, o uso de comandos bem mais simples, como **PRINT AT** e **TAB**.

Outra vantagem dos caracteres gráficos pré-programados é que eles se comportam exatamente como qualquer caractere alfanumérico, para fins de impressão na tela. Assim, valem para eles os comandos **INK**, **INVERSE** e **PAPER**, quando se quer mudar as cores de frente e fundo. É divertido tentar compor desenhos de baixa resolução, usando os três comandos acima mencionados e os caracteres gráficos. E, não havendo necessidade de alta resolução, será sempre mais fácil desenhar com eles do que com os comandos gráficos.

ENTRADA PELO TECLADO

Da mesma forma que os caracteres convencionais do ASCII, os caracteres gráficos podem ser digitados pelo teclado. Para isso, pressione simultaneamente as teclas <**CAPS LOCK**> e <**9**> (a tecla 9 é identificada pelo rótulo **GRAPH**, na parte de cima). Desse modo, coloca-se o teclado em modo gráfico, o que é indicado na tela pelo aparecimento do cursor com a letra G. Em seguida, pressione simultaneamente a tecla <**SHIFT**> e uma das teclas numéricas (de 1 a 8). Você obterá, então, um dos símbolos gráficos marcados nas respectivas telas. Se não acionarmos o <**SHIFT**>, aparecerão os símbolos gráficos complementares, ou invertidos, em relação aos marcados nas teclas. Com isso, teremos acesso aos caracteres desejados, que aparecerão diretamente na tela — por exemplo, dentro de uma cadeia alfanumérica.

Quando terminar de digitar os caracteres gráficos em uma linha, pressione novamente <**CAPS LOCK**> e <**9**>, para sair do modo gráfico.

O programa seguinte desenha uma tabela simples, usando blocos gráficos. Depois, coloca os nomes digitados pelo usuário nas linhas da tabela.

```
200 CLS
210 PRINT "▗▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▖"
220 PRINT "▌ NO.  ▌ NOME                ▐"
230 PRINT "▝▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▘"
240 FOR i=1 TO 10
250 PRINT "▌      ▌                     ▐"
260 NEXT i
270 PRINT "▝▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▘"
280 FOR i=1 TO 10
285 PRINT AT 19,0;"NOME:"
290 INPUT n$
300 PRINT AT i+3,4;i
310 PRINT AT i+3,10;n$
320 NEXT i
```

Nas linhas 220 e 250, os espaços em branco e as letras podem ser digitados normalmente, sem ser necessário sair do modo gráfico. Nesse caso, as letras aparecerão apenas em maiúsculo. Para aparecerem em minúsculo, será preciso "desligar" o modo gráfico.

Existem duas desvantagens na entrada de caracteres gráficos pelo teclado: primeiro, nem todos os símbolos disponíveis estão marcados nas teclas, o que torna o processo mais trabalhoso e sujeito a erros; segundo, o programa não pode ser listado em uma impressora não gráfica, ou que não seja própria para o Spectrum.

No próximo artigo desta série, veremos como utilizar caracteres gráficos dentro de programas.

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

PROGRAMAÇÃO BASIC
Você já teve um primeiro contato com os caracteres gráficos do TRS-80. Aprenda a utilizá-los em desenhos mais complexos.

PERIFÉRICOS
Veja como fazer o melhor uso de sua impressora.

PROGRAMAÇÃO BASIC
No micro, as leis da perspectiva podem ser aplicadas por meio de apenas algumas teclas. Saiba como.

PROGRAMAÇÃO DE JOGOS
Simulador de vôo: complete seu programa.

CÓDIGO DE MÁQUINA
Crie na tela um relógio digital.