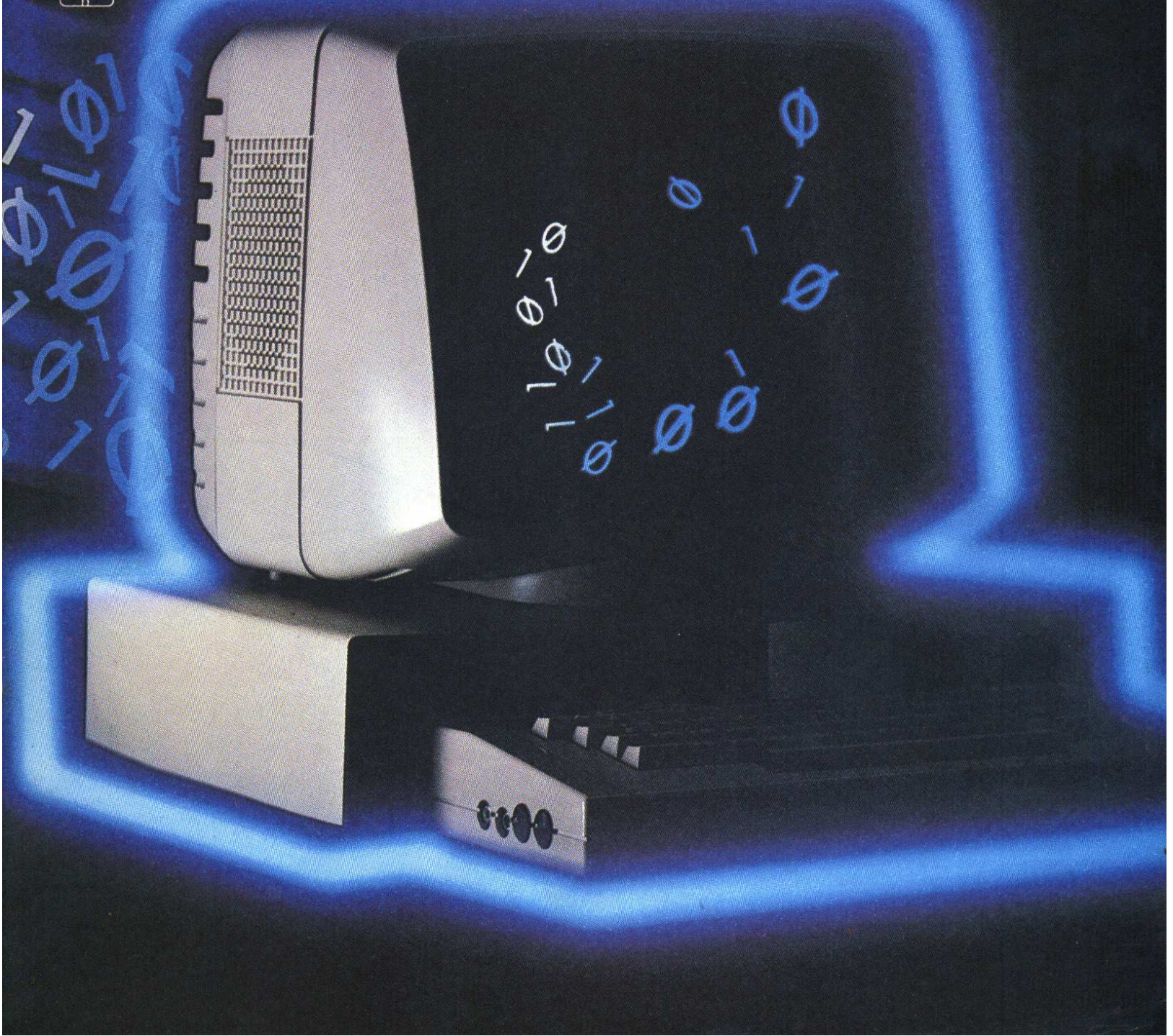
CURSO PRÁTICO 37 DE PROGRAMAÇÃO DE COMPUTADORES

INPUT

PROGRAMAÇÃO BASIC - PROGRAMAÇÃO DE JOGOS - CÓDIGO DE MÁQUINA

NOVA CULTURAL

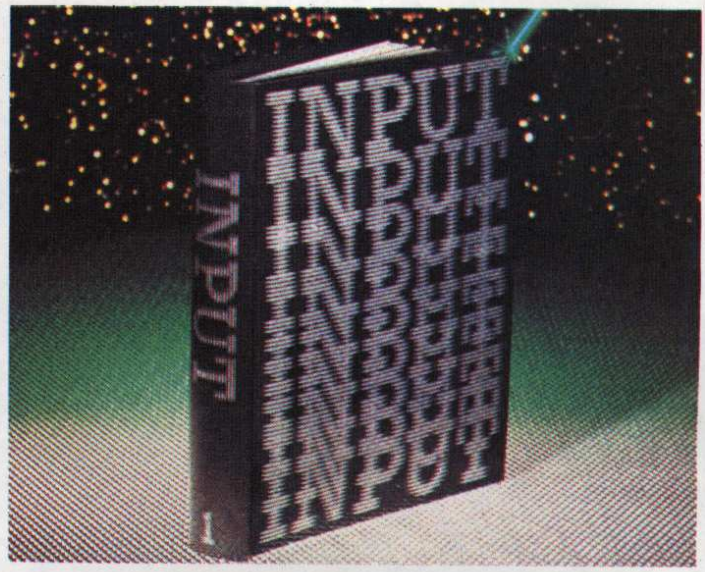
Cz$ 20,00



INPUT

Vol. 3 Nº 37

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.

COMPLETE SUA COLEÇÃO

Exemplares atrasados, até seis meses após o encerramento da coleção, poderão ser comprados, a preços atualizados, da seguinte forma: 1. PESSOALMENTE — Por meio de seu jornaleiro ou dirigindo-se ao distribuidor local, cujo endereço poderá ser facilmente conseguido junto a qualquer jornaleiro de sua cidade. Em **São Paulo**, os endereços são: rua Brigadeiro Tobias, 773, Centro; av. Industrial, 117, Santo André; e no **Rio de Janeiro**: rua da Passagem, 93, Botafogo. 2. POR CARTA — Poderão ser solicitados exemplares atrasados também por carta, que deve ser enviada para DINAP — Distribuidora Nacional de Publicações — Números Atrasados — Estrada Velha de Osasco, 132, Jardim Teresa — CEP 06000 — Osasco — SP. Não envie pagamento antecipado. O atendimento será feito pelo reembolso postal e o pagamento, incluindo as despesas postais, deverá ser efetuado ao se retirar a encomenda na agência do Correio. 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 — 2 685, Camarate — Lisboa; Apartado 57 — Telex 43 069 JARLIS P.
Atenção: Após seis meses do encerramento da coleção, os pedidos serão atendidos dependendo da disponibilidade do estoque.
Obs.: Quando pedir livros, mencione sempre 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 9442, São Paulo — SP.

Editor
VICTOR CIVITA

REDAÇÃO
Diretora Editorial: Iara Rodrigues

Editor Executivo: Antonio José Filho

Editor Chefe: Paulo de Almeida
Editor de Texto: Cláudio A. V. Cavalcanti
Chefe de Arte: Carlos Luiz Batista
Assistentes de Arte: Ailton Oliveira Lopes, Dilvacy M. Santos, Grace Alonso Arruda, José Maria de Oliveira, Monica Lenardon Corradi
Secretária de Redação/Coordenadora: Stefania Crema
Secretários de Redação: Beatriz Hagström, José Benedito de Oliveira Damião, Maria de Lourdes Carvalho, Marisa Soares de Andrade, Mauro de Queiroz

COLABORADORES
Consultor Editorial Responsável: Dr. Renato M. E. Sabbatini (Diretor do Núcleo de Informática Biomédica da Universidade Estadual de Campinas)
Execução Editorial: DATAQUEST Assessoria em Informática Ltda., Campinas, SP
Tradução: Reinaldo Cúrcio
Tradução, adaptação, programação e redação: Abílio Pedro Neto, Aluísio J. Dornellas de Barros, Marcelo R. Pires Therezo, Raul Neder Porrelli
Coordenação Geral: Rejane Felizatti Sabbatini
Editora de Texto: Ana Lúcia B. de Lucena
Assistente de Arte: Dagmar Bastos Sampaio

COMERCIAL
Diretor Comercial: Roberto Martins Silveira
Gerente Comercial: Flávio Maculan
Gerente de Circulação: Denise Maria Mozol

PRODUÇÃO
Gerente de Produção: João Stungis
Coordenador de Impressão: Atilio Roberto Bonon

Preparador de Texto/Coordenador: Eliel Silveira Cunha
Preparadores de Texto: Alzira Moreira Braz, Ana Maria Dilguerian, Karina Ap. V. Grechi, Levon Yacubian, Luciano Tasca, Maria Teresa Galluzzi, Maria Teresa Martins Lopes, Paulo Felipe Mendrone
Revisor/Coordenador: José Maria de Assis
Revisoras: Conceição Aparecida Gabriel, Isabel Leite de Camargo, Ligia Aparecida Ricetto, Maria de Fátima Cardoso, Nair Lúcia de Brito
Paste-up: Anastase Potaris, Balduino F. Leite, Edson Donato

Edição organizada pela Editora Nova Cultural Ltda.
Av. Brigadeiro Faria Lima, nº 2000 - 3º andar
CEP 01452 - São Paulo - SP - Brasil
(Artigo 15 da Lei 5 988, de 14/12/1973).
Esta obra foi composta na AM Produções Gráficas Ltda. e impressa na Divisão Gráfica da Editora Abril S.A.

MÚSICA EM SEU MICRO

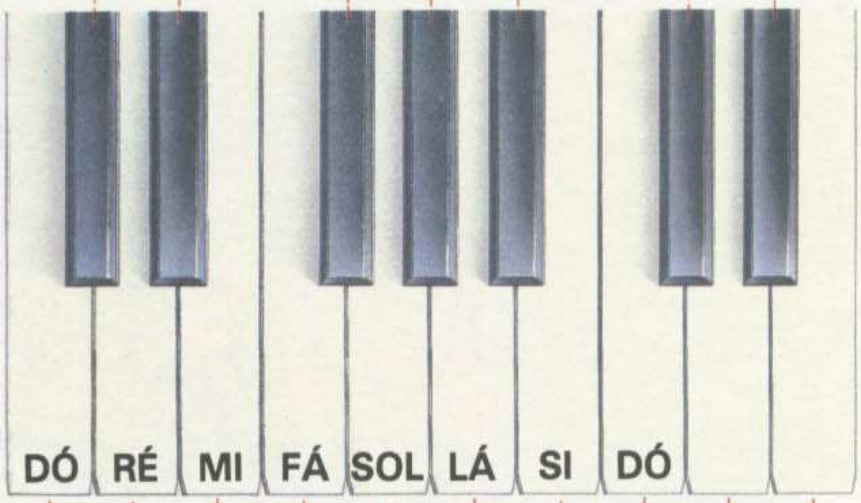
- NOTAS E ESCALAS
- TONS E SEMITONS
- PEQUENAS MELODIAS
- TRANSFORME O COMPUTADOR NUM ÓRGÃO MUSICAL

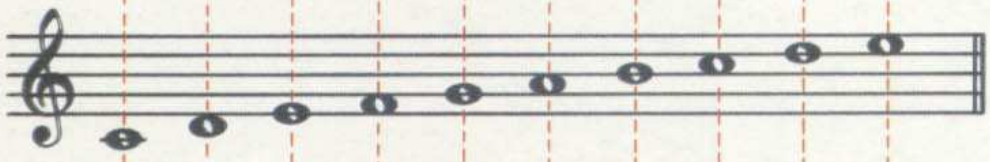
Bach ou Beethoven? Pink Floyd ou The Cure? Seja qual for a sua escolha musical, você pode tornar-se um ótimo instrumentista, dedilhando o teclado de um microcomputador.

Com exceção do Apple, todos os microcomputadores mencionados neste artigo contam com algum tipo de comando capaz de produzir sons. O funcionamento de comandos sonoros, aliás, já foi explicado anteriormente em *Quebre a Barreira do Som* (página 168). Um artigo posterior — *Apple e TK-2000: Efeitos Sonoros* — abordou o funcionamento de uma rotina em código de máquina, que será utilizada no programa deste artigo; ela permite que os micros dessas linhas produzam sons.

De um modo geral, é possível selecionar tanto a tonalidade como a duração da nota emitida. No MSX e no TRS-Color, outros parâmetros também podem ser modificados.

A forma de produzir sons por meio de um programa em BASIC varia de computador para computador. No MSX, podemos tocar até três notas ao mesmo tempo, o que possibilita a execução de acordes musicais. Os demais produzem só uma nota por vez. O MSX permite que controlemos outros parâmetros essenciais para certas aplicações.





Nenhuma dessas diferenças é relevante no momento. De fato, melodias simples podem ser executadas em qualquer um dos cinco computadores abordados neste artigo, que pretende ser apenas um guia para principiantes. Começaremos com algumas noções de teoria musical, necessárias à compreensão do programa, que transforma parte do teclado de seu micro em um instrumento simples. Em artigo posterior, veremos de que maneira tirar maior proveito dos recursos musicais do computador.

Antes de iniciar, devemos explicar o significado de dois termos fundamentais. O primeiro deles é a expressão "altura", que aqui quer dizer o mesmo que tom ou tonalidade. Ele permite classificar as notas musicais em graves e agudas. Assim, uma nota mais alta que outra será mais aguda que ela; uma nota mais baixa será mais grave. O segundo termo, "intervalo", indica a distância musical entre duas notas de tonalidade diferente.

UMA ESCALA SIMPLES

Uma escala é uma série ascendente de notas musicais, cada uma mais alta que a anterior. Embora em computação seja tecnicamente melhor recorrer a letras do alfabeto — de A a G — para designar as notas (como acontece na Europa e nos Estados Unidos), o sistema adotado no Brasil emprega os nomes tradicionais — dó, ré, mi, fá, sol, lá, si. Assim, uma escala começa com dó e vai "subindo" até o próximo dó, num total de oito notas.

Essa escala "dó-ré-mi" é chamada de "escala maior". Ela é definida por uma relação musical entre as notas. Assim, qualquer seqüência de notas com o mesmo padrão de intervalos musicais entre si é uma "escala maior".

E AS TECLAS PRETAS?

As teclas brancas de um piano correspondem às notas (letras) A, B, C, D, E, F e G, num ciclo que se repete pelo teclado afora — C é dó, D é ré, F é mi etc. A "escala maior" corresponde a oito teclas, começando num C e terminando no próximo C. Essa escala é denominada escala de "C maior", por motivos óbvios. Como dissemos antes, uma "escala maior" pode iniciar-se em qualquer tonalidade, mas a escala de "C maior" deve começar necessariamente num C, que tem uma altura específica. A escolha dessa letra como primeira nota fixa a tonalidade da escala.

Para que servem então as teclas pretas do teclado? Acontece que a "escala maior" não contém todas as notas possíveis entre a primeira e a última letras. Entre dó e ré, por exemplo, existe uma nota que não pertence à "escala maior"; o mesmo ocorre com outros pares de notas. As teclas pretas correspondem a esse tipo de som intermediário. Assim, a tecla preta entre C e D (dó e ré em "C maior") é chamada de C sustenido ou D bemol ("sustenido" significa mais alto e "bemol", mais baixo), conforme usemos a nota acima ou abaixo dela como referência.

A nota entre D e E é D sustenido ou E bemol, e assim por diante. Note que não há tecla preta entre E e F (mi e fá em "C maior") nem entre B e C (si e dó em "C maior").

O intervalo existente entre duas teclas vizinhas (mesmo que elas tenham cores diferentes) é chamado de semitom, de maneira que os intervalos entre C e C sustenido, C sustenido e D e entre E e F são todos semitons. Um intervalo que contenha dois semitons é denominado tom "inteiro".

O arranjo de tons e semitons que definem uma escala maior é:

dois semitons = um tom entre dó e ré (C e D em "C maior").

dois semitons = um tom entre ré e mi (D e E em "C maior")

um semitom entre mi e fá (E e F em "C maior").

dois semitons = um tom entre fá e sol (F e G em "C maior").

dois semitons = um tom entre sol e lá (G e A em "C maior").

dois semitons = um tom entre lá e si (A e B em "C maior").

um semitom entre si e dó (B e C em "C maior").

OUTRAS ESCALAS MAIORES

Segundo o que foi exposto, existem doze semitons entre o primeiro e o segundo dó de uma "escala maior" (total de oito notas).

Suponhamos agora que precisamos de uma "escala maior" em outra tonalidade — começando em G, por exemplo. Nela, G seria dó e A seria ré; mas, se tocássemos as seis teclas seguintes, o arranjo de intervalos não seria o de uma "escala maior": os intervalos entre lá e si, e si e dó não estariam corretos. Para obtermos uma "escala maior" verdadeira, F deve ser substituído por F sustenido (ou si, em nossa nova escala), restabelecendo o padrão correto de intervalos. Para entender melhor, estude o diagrama da página anterior ou use um teclado de verdade.

Uma "escala maior" começando em F seria então: F, G, A, B bemol, C, D, E e F. (B deve ser bemol, caso contrário a nota fá e os intervalos mi-fá e fá-sol estarão errados).

Podemos obter uma "escala maior" começando em qualquer nota, desde que respeitemos o arranjo de intervalos. Mas qualquer escala não iniciada em C deverá usar uma ou mais teclas pretas: apenas a escala "C maior" usa só teclas brancas. Fica explicado porque ela é tão popular: é mais fácil tocar usando só as teclas brancas.

Muitas músicas simples — como cantigas de ninar, músicas folclóricas e alguns hinos — podem ser tocadas na escala maior. A canção *Três Ratinhos Cegos*, por exemplo, começa assim:

mi, ré, dó
mi, ré, dó
sol, fá, fá, mi
sol, fá, fá, mi...

e as primeiras notas de *Atirei o Pau no Gato* são:

sol, lá, mi, ré, mi, fá, sol, sol,
sol, fá, sol, lá, lá, lá,
sol, lá, mi, mi, mi...

Escrevemos aqui apenas as tonalidades; o ritmo fica por conta do leitor. Só com notação musical — um tema que não abordaremos agora — poderíamos transmitir a informação completa a respeito da melodia — altura, ritmo, intensidade, entre outras coisas.

Músicas mais complexas não poderão ser executadas somente com as teclas brancas. Geralmente, começamos em uma escala e, com o desenvolvimento da música, passamos temporariamente para outras escalas, usando notas que não aparecem na primeira.

Veja, por exemplo, o caso de **Yesterday**, música composta por John Lennon e Paul McCartney que se tornou um dos grandes sucessos dos Beatles:

ré, dó, dó, mi, fá, fá sustenido, sol sustenido, lá, si, dó, si, lá, lá...

Essas notas, estranhas à escala original, são um requinte; elas dão um toque especial a muitas melodias.

BEMÓIS OU SUSTENIDOS?

Por que se usa o termo "sustenido" (em F sustenido) na escala "G maior", e "bemol" (em B bemol) na escala "F maior"? Afinal, F sustenido e G bemol correspondem à mesma tecla preta. Por que não usar só um dos termos, deixando o outro de lado? Quem tentar criar novas "escalas maiores" — começando em notas diferentes de C — terá a resposta. Na verdade, os dois termos são necessários para garantir que, em qualquer escala, cada nota seja representada por uma letra diferente. Assim, a substituição do B bemol da escala "F maior" por A sustenido, resultaria em: F, G, A, A sustenido, C, D, E e F. Em tal seqüência de notas aparecem dois A e nenhum B, o que pode gerar confusão.

Da mesma maneira, a substituição de F sustenido por G bemol em "G maior", resultaria em uma escala com dois G e nenhum F. Já o emprego de bemóis no lugar de sustenidos, ou viceversa, é tão deselegante como erros de ortografia em um texto. Programas para compor música que usem só bemóis — ou sustenidos —, com a justificativa de que C sustenido é igual a D bemol, deixam muito a desejar.

FREQÜÊNCIA E INTERVALOS

Som é o efeito produzido pelas vibrações no ar. Quanto mais rapidamente estas se repetirem, ou quanto maior for sua freqüência, mais aguda será a nota correspondente. A unidade usada para exprimir freqüência é cps (ciclos por segundo) ou Hz (hertz, que significa a mesma coisa).

Se tomarmos a nota C, cuja freqüência é igual a 256 Hz, e dobrarmos esse valor, obteremos outra nota C uma oitava acima da primeira — mais aguda, portanto (uma oitava é o intervalo entre o dó mais baixo e o dó mais alto). Dobrando novamente o valor da freqüência, ele passará de 512 para 1024 Hz, que corresponde à próxima nota C — uma oitava acima da anterior. Desse modo, quando *multiplicamos* a freqüência de uma nota, *adicionamos* a ela um intervalo musical.

Já vimos que há doze semitons em uma oitava (treze teclas, oito brancas e cinco pretas, com doze semitons *entre* si). Se, dobrando a freqüência, subimos uma oitava, por que fator devemos multiplicar a freqüência de uma nota para subir um semitom? No caso de uma oitava acima, temos a equação:

freqüência da nota x 2 = freqüência da nota uma oitava mais alta.

Chamamos de X o fator necessário para subir um semitom. Teremos então:

freqüência da nota
xXxXxXxXxXxXxXxXxXxXxX =
freqüência da nota uma oitava mais alta.
*X*X*X*X*X*X*X*X*X*X*X*X =

O número que divide a razão 2:1 em doze partes iguais é a décima segunda raiz de 2 (para calcular esse valor em BASIC, use a expressão **2↑(1/12)**). Se multiplicarmos 256 por esse valor, obteremos a nota um semitom mais alta. Uma nova multiplicação adiciona mais um semitom. Se repetirmos o processo, após doze multiplicações consecutivas, chegaremos à freqüência da oitava seguinte. Por tudo isso, a décima segunda raiz de dois é uma constante fundamental da música.

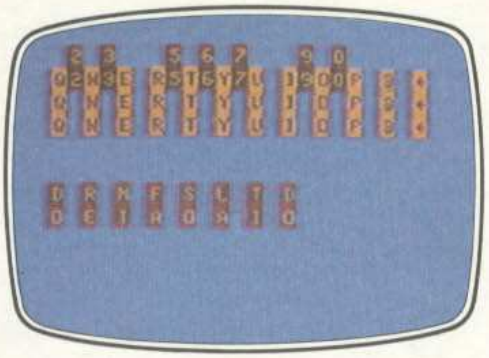
MELODIAS NA "ESCALA MAIOR"

Se quisermos tocar uma música usando o programa do final deste artigo, teremos que adivinhar qual nota é dó, qual é ré, e assim por diante. A questão é: como tocar "de ouvido"?

Existirá algum método para fazer com que a música possa ser tocada somente nas teclas brancas, facilitando sua execução? Neste caso também, um caminho possível é descobrir qual nota da música é um dó, deduzindo, a partir daí, as posições das demais notas. Muitas melodias começam ou terminam com essa nota; além disso, o dó acaba funcionando como o "centro de gravidade" de certas músicas simples. Uma vez encontrada, essa nota nos servirá de guia, e, com um pouco de sorte, descobriremos as outras. Tente tocar a música desde o início, nota por nota; preste muita atenção no que estiver ouvindo, procurando perceber se a tonalidade está subindo ou descendo naquele ponto e se a nota seguinte se encontra na tecla vizinha, ou mais além.

Se nos enganarmos sobre qual é a nota dó, algumas notas parecerão muito graves ou muito agudas e teremos de procurar o dó "certo". Com paciência e perseverança, porém, qualquer um pode desenvolver uma espécie de "intuição musical" e alguns serão capazes de tocar músicas inteiras no teclado de um microcomputador.

Para facilitar o trabalho dos principiantes, contudo, este artigo menciona as notas de algumas melodias mais ou menos conhecidas.



É assim que o diagrama do teclado aparece na tela dos micros da linha TRS-Color.

UM TECLADO MUSICAL

Os programas a seguir transformam a parte superior do teclado do seu computador em um pequeno órgão. As letras Q, W, E, R, T, Y, U, I serão as notas dó, ré, mi etc. As teclas da direita avançam na oitava seguinte. As teclas 2, 3, 5, 6, 7 funcionam como as teclas pretas do piano. Um diagrama de seu teclado fica mais ou menos assim:

```
2 3   5 6 7 etc.
Q W E R T Y U I etc.

d r m f s l s d
ó é i á o á i ó
```

O programa coloca na tela um diagrama semelhante para orientar o tecladista. Além disso, quando forem tocadas as notas da primeira oitava da "escala maior", seus nomes aparecerão num canto do vídeo.

Digite a seguir o programa específico para o seu micro e siga as instruções que o acompanham para se tornar um "virtuose".

S

O programa do Spectrum transforma as duas linhas superiores do teclado em um órgão. Para ouvir uma nota, basta pressionar a tecla correspondente.

A listagem começa estabelecendo os valores das variáveis que correspondem à duração das notas, ao endereço do laço principal (guardado na variável **loop**, para economizar memória), e à coordenada x do comando **PRINT AT**, usado para impressão das notas na tela. Ela continua selecionando as cores da tela, e entra a seguir na porção responsável pela produção de sons.

A variável **loop** contém o número da linha para o qual o programa retornará após executar uma nota; ele aguardará então que você pressione a tecla seguinte. Quando isso acontecer, o código do caractere correspondente será colocado na variável **nota**, com o auxílio do comando **CODE**.

O computador irá então para a linha cujo número está em **nota**. Os números de linha correspondem aos códigos das teclas usadas (de 48 a 121). O comando **SOUND** em cada uma dessas linhas é ajustado para tocar a nota correspondente à tecla (Q, por exemplo, corresponde a dó). Se consultarmos o manual — ou o artigo *Quebre a Barreira do Som* —, veremos que o primeiro número após **SOUND** corresponde à duração da nota (aqui representada pela variável **d**) e o segundo, à tonalidade.

Quando a tecla apertada for de uma nota da "escala maior", o nome desta também será impresso na tela, pela mesma linha que a executa.

Depois de emitir o som, o computador retorna à linha 1000, onde espera uma nova tecla ser pressionada.

```
  1 GOTO 900
 47 GOTO loop
 48 SOUND d,15: GOTO loop
 50 SOUND d,1: GOTO loop
 51 SOUND d,3: GOTO loop
 53 SOUND d,6: GOTO loop
 54 SOUND d,8: GOTO loop
 55 SOUND d,10: GOTO loop
 57 SOUND d,13: GOTO loop
101 PRINT AT y,x;"MI": SOUND d
,4: GOTO loop
105 PRINT AT y,x;"DO": SOUND d
,12: GOTO loop
111 SOUND d,14: GOTO loop
112 SOUND d,16: GOTO loop
113 PRINT AT y,x;"DO": SOUND d
,0: GOTO loop
114 PRINT AT y,x;"FA": SOUND d
,5: GOTO loop
116 PRINT AT y,x;"SO": SOUND d
,7: GOTO loop
117 PRINT AT y,x;"SI": SOUND d
,11: GOTO loop
119 PRINT AT y,x;"RE": SOUND d
,2: GOTO loop
121 PRINT AT y,x;"LA": SOUND d
,9: GOTO loop
800 GOTO loop
900 LET D=.03: LET X=5: LET lo
op=1000
901 BORDER 4: PAPER 4: CLS
902 PRINT AT 8,6;"d r m f s l
s d"
903 PRINT AT 9,6;"o e i a o a
i o"
910 LET a$=" 2 3   5 6 7   9 0
"
920 FOR y=3 TO 4: GOSUB 990
930 NEXT y
940 LET a$="Q W E R T Y U I O
p"
941 PAPER 7: INK 0
950 FOR y=4 TO 6: GOSUB 990
```

```
960 NEXT y
980 LET x=15: LET y=15: GOTO 1
oop
990 FOR i=1 TO LEN a$
991 IF a$(i)<>CHR$ 32 THEN
PRINT AT y,x+i;a$(i);
992 NEXT i: RETURN
1000 PRINT AT y,x;CHR$ 32;CHR$
32
1005 LET a$=INKEY$: IF a$="" TH
EN  GOTO 1000
1100 LET nota=CODE a$: GOTO not
a
```

Aqueles que possuem micros da linha Spectrum importados devem substituir o comando **SOUND** pelo comando **BEEP**, que tem a mesma sintaxe.

Após iniciar a execução do programa, mantenha uma das teclas pressionadas para ouvir um som pulsante correspondente à auto-repetição da tecla. O som não é emitido continuamente, como num órgão, pois o comando **SOUND** produz sons com duração limitada. Assim, o que o programa realmente faz é tocar a mesma nota repetidas vezes.

MSX

Vejamos agora o que acontece com os modelos da linha MSX.

Os microcomputadores dessa linha contam com poderosos comandos sonoros graças a um microprocessador interno dedicado inteiramente à produção de sons, e capaz de funcionar simultaneamente com a unidade central. Para programá-lo em BASIC, precisamos recorrer a vários comandos **SOUND**, com o propósito de definir diversos parâmetros do som — ou ruído — que estamos querendo emitir. Uma introdução a esse assunto foi apresentada no artigo *Quebre a Barreira do Som*, onde abordamos vários sons não-musicais.

Contudo, para transformar a parte superior do teclado num programa semelhante ao dos outros micros, podemos usar o comando **PLAY**, que torna o programa mais simples e curto.

A primeira linha do programa estabelece os parâmetros iniciais das notas: volume, duração, velocidade de execução. As linhas 20 e 30 colocam todas as notas usadas pelo programa dentro da variável indexada **N$**, recorrendo ao comando **READ** e aos dados da linha **DATA**. A linha 40 cria um cordão contendo todas as teclas usadas e um outro que permite dar nome a algumas das notas (dó, ré, mi etc). As linhas que vão de 50 a 100 cuidam da organização do vídeo. A primeira delas seleciona a tela de texto com 32 colunas, escolhe as cores e desliga as teclas de função.

Nas três linhas seguintes — ou seja, 55, 60 e 65 —, o comando **VPOKE** é utilizado para modificar a tabela de cores — **BASE (6)** —, permitindo, assim, a impressão de letras coloridas. Se você não gostar das cores, pode mudar os códigos nos locais onde eles aparecem — para maiores informações, veja o artigo *Os Comandos PEEK e POKE* (página 261).

Na linha 70 , o comando **VPOKE** é novamente utilizado, desta vez para escrever na tela. O **FOR...NEXT** contido nessa linha imprime parte do diagrama do teclado na tela, obtendo com **READ** os códigos dos caracteres de cada uma delas na linha 75 e modificando a tabela de nomes — **BASE(5)**. O restante do diagrama é feito do mesmo modo pelas linhas 80 e 85.

As linhas 90 e 100 completam o quadro, imprimindo o nome das notas sob as teclas correspondentes.

A linha 120 espera que acionemos uma tecla. A 130 descobre qual das notas do cordão **M$** corresponde à tecla pressionada; se a tecla não pertencer ao conjunto predefinido, a linha 120 será repetida.

A linha 140 é a que realmente executa a nota, usando o **PLAY**. A altura da nota é selecionada pela variável **NT**, que é usada para escolher o elemento de **N$** que contém os operandos adequados do **PLAY**.

A seguir, um comando **ON GOTO** na linha 150 imprimirá o nome da nota no canto superior esquerdo da tela, se ela pertencer à primeira oitava de "C maior". O programa retorna então para verificar uma nova tecla.

```
10 PLAY "V15L64T255"
20 DIM N$(21):FOR K=0 TO 16:REA
D N$(K):NEXT
30 DATA C,C♯,D,D♯,E,F,F♯,G,G♯,A
,A♯,B,04C,04C♯,04D,04D♯,04E
40 M$="Q2W3ER5T6Y7UI900P":D$="Q
WERTYUIO"
50 SCREEN 1:COLOR 1,6,6:KEY OFF
55 FOR I=6 TO 7:VPOKE BASE(6)+I
,15*16+1:NEXT
60 FOR I=8 TO 11:VPOKE BASE(6)+
I,16+15:NEXT
65 FOR I=12 TO 15:VPOKE BASE(6)
+I,16+10:NEXT
70 FOR K=0 TO 6:READ A:N=2*A-20
*INT(A/10):VPOKE BASE(5)+71+N,A
:VPOKE BASE(5)+103+N,A:NEXT
75 DATA 50,51,53,54,55,57,48
80 FOR K=0 TO 9:READ A:VPOKE BA
SE(5)+102+K*2,A:VPOKE BASE(5)+1
34+K*2,A:VPOKE BASE(5)+166+K*2,
A:NEXT
85 DATA 81,87,69,82,84,89,85,73
,79,80
90 LOCATE 4,9:PRINT "d r m f s
l s d"
100 LOCATE 4,10:PRINT "o e i a
o a i o"
120 A$=INKEY$:IF A$="" THEN 120
130 NT=INSTR(M$,A$):IF NT=0 THE
N 120
140 PLAY "O3"+N$(NT-1)
150 LOCATE 0,0:D=INSTR(D$,A$):O
N D GOTO 170,180,190,200,210,22
0,230,170
160 PRINT "  ":GOTO 120
170 PRINT "DO":GOTO 120
180 PRINT "RE":GOTO 120
190 PRINT "MI":GOTO 120
200 PRINT "FA":GOTO 120
210 PRINT "SO":GOTO 120
220 PRINT "LA":GOTO 120
230 PRINT "SI":GOTO 120
```

UM NOVO TECLADO

O programa do Apple utiliza a rotina de produção de sons do artigo *Apple e TK-2000: Efeitos Sonoros*. O TK-2000 não precisa dela, já que dispõe do comando **SOUND**.

A linha 10 limpa o vídeo e ativa a tela gráfica de baixa resolução. A linha 20 transfere para a memória os códigos da rotina de geração de som, contida na linha 30.

A linha 35 cuida de colorir a tela, usando uma série de traços horizontais — verdes, no Apple — desenhados por comandos **HLIN**. As linhas 40 a 70 desenham um teclado de piano. As teclas brancas e pretas são feitas por comandos **VLIN**. As coordenadas para uso desses comandos ficam nas linhas **DATA** 50 e 70.

As linhas 80 e 90 imprimem as letras das teclas correspondentes ao desenho do piano, para orientar o usuário. Observe que há dois espaços entre cada par de letras. A linha 100, por sua vez, imprime o nome das notas (os tradicionais dó, ré, mi...).

A seguir, o programa espera que seja pressionada uma tecla, colocando seu caractere em **K$**.

Na linha 110, o comando **POKE 900,60** define a duração das notas — o endereço 900 é usado justamente para conter essa duração.

Depois disso, uma grande quantidade de linhas **IF...THEN** testa o conteúdo de **K$** para descobrir qual tecla foi pressionada, emitindo a nota correspondente em cada caso. O endereço 901 é usado para determinar a tonalidade. A rotina de produção de som é executada pelo comando **CALL 800**.

Detectada a tecla pressionada, o programa retorna à linha 110 para aguardar uma nova nota.

Por que as teclas musicais criadas pelo programa nos microcomputadores da linha TRS-Color não têm auto-repetição?

O programa do TRS-Color usa habitualmente o comando **INKEY$** para verificar o teclado.

É possível obtermos a auto-repetição das teclas por intermédio do comando **PEEK** (essa forma tem sido muito utilizada por nós em rotinas de movimentação do cursor). No entanto, o programa teria que acionar, neste caso, mais de vinte **PEEK** com diferentes números, um para cada tecla musical.

Como tais números não teriam nenhuma relação entre si (as teclas, neste caso, são QWE... e não ABC...), seria preciso colocá-los em linhas separadas ou armazená-los em uma linha **DATA**, tornando a execução do programa muito demorada e complicada. É mais rápido, cômodo e simples pressionar uma tecla de cada vez.

A duração de uma nota pode ser controlada pelo tempo em que a tecla se mantém sob pressão?

De um modo geral, a resposta é não. Na maioria dos computadores isso só seria possível com linguagem de máquina, já que o programa teria que fazer duas coisas ao mesmo tempo: tocar a nota e verificar se a tecla continua sendo pressionada.

Contudo, no micro MSX esse controle é possível; é que, nesse computador, o dispositivo de som funciona independentemente da unidade de processamento central.

A produção de acordes — emissão de mais de uma nota simultaneamente — para enriquecer uma melodia também é possível no MSX, que conta com três canais de som.

```
10  HOME : GR
20  FOR I = 0 TO 22: READ A: PO
KE 800 + I,A: NEXT
30  DATA  160,0,174,133,3,238,4
8,192,136,208,5,206,132,3,240,6
,202,208,245,76,34,3,96
35  COLOR= 12: FOR I = 0 TO 39:
 HLIN 0,39 AT I: NEXT
40  COLOR= 0: FOR I = 1 TO 7: R
EAD A: VLIN 20,30 AT A: NEXT
50  DATA  8,11,17,20,23,29,32
60  COLOR= 15: FOR I = 1 TO 20:
 READ A: VLIN 20,39 AT A: NEXT
70  DATA  6,7, 9,10, 12,13,15,1
6,18,19, 21,22,24,25,27,28,30,3
1,33,34
80  VTAB 21: HTAB 9: PRINT "2
3"; TAB( 18);"5  6  7"; TAB( 30
);"9  0"
90  VTAB 22: HTAB 8: PRINT "Q
W  E  R  T  Y  U  I  O  P"
100  VTAB 24: HTAB 7: PRINT "DO
 RE MI FA SO LA SI DO";
110  GET K$: POKE 900,60
120  IF K$ = "Q" THEN  POKE 901
,96: CALL 800: GOTO 110
130  IF K$ = "2" THEN  POKE 901
,90: CALL 800: GOTO 110
140  IF K$ = "W" THEN  POKE 901
,85: CALL 800: GOTO 110
150  IF K$ = "3" THEN  POKE 901
,80: CALL 800: GOTO 110
160  IF K$ = "E" THEN  POKE 901
,76: CALL 800: GOTO 110
170  IF K$ = "R" THEN  POKE 901
,72: CALL 800: GOTO 110
180  IF K$ = "5" THEN  POKE 901
,67: CALL 800: GOTO 110
190  IF K$ = "T" THEN  POKE 901
,64: CALL 800: GOTO 110
200  IF K$ = "6" THEN  POKE 901
,60: CALL 800: GOTO 110
210  IF K$ = "Y" THEN  POKE 901
,56: CALL 800: GOTO 110
220  IF K$ = "7" THEN  POKE 901
,53: CALL 800: GOTO 110
230  IF K$ = "U" THEN  POKE 901
,50: CALL 800: GOTO 110
240  IF K$ = "I" THEN  POKE 901
,47: CALL 800: GOTO 110
250  IF K$ = "9" THEN  POKE 901
,45: CALL 800: GOTO 110
260  IF K$ = "O" THEN  POKE 901
,42: CALL 800: GOTO 110
270  IF K$ = "0" THEN  POKE 901
,40: CALL 800: GOTO 110
280  IF K$ = "P" THEN  POKE 901
,37: CALL 800: GOTO 110
290  GOTO 110
```

O programa acima funciona normalmente nos micros da linha TK-2000. Contudo, muitos de seus usuários podem preferir usar o comando **SOUND**.

Para isto, basta eliminar as linhas 20 e 30 e modificar a linha 110, dando-lhe a seguinte forma:

```
110 GET K$: D = 60
```

A seguir, modifique todas as ocorrências de:

```
POKE 901,F:CALL 800
```

para

```
SOUND F,D
```

onde **F** não é uma variável, mas o mesmo número que ocorre após o **POKE 901**.

Os usuários do TK-2000 podem mudar as cores nas linhas 35, 40 e 60, já que as cores do Apple são diferentes.

O programa começa limpando a tela e colorindo-a de azul. A mesma linha também estabelece os parâmetros de volume, duração e velocidade das notas que serão executadas pelo comando **PLAY**. O artigo *Quebre a Barreira do Som* explica a sintaxe desse comando.

As linhas 20 e 30 posicionam as notas usadas pelo programa dentro da matriz **N$**. A linha 40 coloca num cordão as teclas que serão utilizadas e em outro os nomes correspondentes a algumas delas (dó, ré, mi etc).

As linhas 50 a 110 cuidam dos desenhos na tela, colorindo-os de laranja e preto. Depois disso, o computador espera que pressionemos uma tecla. Se esta não corresponder a uma nota, o computador voltará à linha 120 e esperará por outra.

A linha 140 executa a nota com o auxílio do comando **PLAY**. A tonalidade da nota é definida pela variável **NT**, que depende da tecla pressionada. **NT** é usada para selecionar o elemento da matriz **N$** que contém os parâmetros adequados da instrução **PLAY**.

Usando o comando **ON...GOTO**, o computador imprimirá na tela o nome da nota (se esta fizer parte da primeira oitava de " C maior"), retornando em seguida para esperar uma nova tecla.

```
10 CLS 3:PLAY"V31L4T8":B$=CHR$(
175)
20 DIM N$(21):FOR K=0 TO 19:REA
D N$(K):NEXT
30 DATA C,C♯,D,D♯,E,F,F♯,G,G♯,A
,A♯,B,O4C,O4C♯,O4D,O4D♯,O4E,O4F
,O4F♯,O4G
40 M$="Q2W3ER5T6Y7UI900P@-"+CHR
$(8):D$="QWERTYUIO"
50 FOR K=0 TO 6:READ A:N=2*A-20
*INT(A/10):POKE 1093+N,A:POKE 1
125+N,A:NEXT
60 DATA 50,51,53,54,55,57,48
65 POKE 1113,45:POKE 1145,45
70 FOR K=0 TO 11:READ A:POKE 11
24+K*2,A:POKE 1156+K*2,A:POKE 1
188+K*2,A:NEXT
80 DATA 81,87,69,82,84,89,85,73
,79,80,64,95
90 PRINT @260,"d"B$"r"B$"m"B$"f
"B$"s"B$"l"B$"s"B$"d";
100 PRINT @292,"o"B$"e"B$"i"B$"
a"B$"o"B$"a"B$"i"B$"o";
110 SCREEN 0,1
120 A$=INKEY$:IF A$="" THEN 120
130 NT=INSTR(M$,A$):IF NT=0 THE
N 120
140 PLAY"O3"+N$(NT-1)
150 PRINT @0,;:D=INSTR(D$,A$):O
N D GOTO 170,180,190,200,210,22
0,230,170
160 PRINT B$;B$;:GOTO 110
170 PRINT"DO";:GOTO 110
180 PRINT"RE";:GOTO 110
```

```
190 PRINT"MI";:GOTO 110
200 PRINT"FA";:GOTO 110
210 PRINT"SO";:GOTO 110
220 PRINT"LA";:GOTO 110
230 PRINT"SI";:GOTO 110
```

UMA SINFONIA DE BEETHOVEN

O que podemos tocar nesse teclado musical que o programa acaba de criar? Que tal uma canção de ninar, por exemplo? Ela começa com dó, de forma que a primeira tecla é Q; você terá que descobrir o ritmo "de ouvido":

Q,W,R,E,T,E,Q
Q,E,W,R,E,Q
Q,E,W,R,E,T,E,Q
Y,W,R,E,Q

E, agora, esta passagem famosa da *Nona Sinfonia* de Beethoven, composta quando o artista já estava surdo:

E,E,R,T,T,R,E,W
Q,Q,W,E,E,W,W
E,E,R,T,T,R,E,W
Q,Q,W,E,W,Q,Q
W,E,Q,W,E,R,E,Q
W,E,R,E,W,Q,W,T
E,E,R,T,T,R,E,W
Q,Q,W,E,W,Q,Q

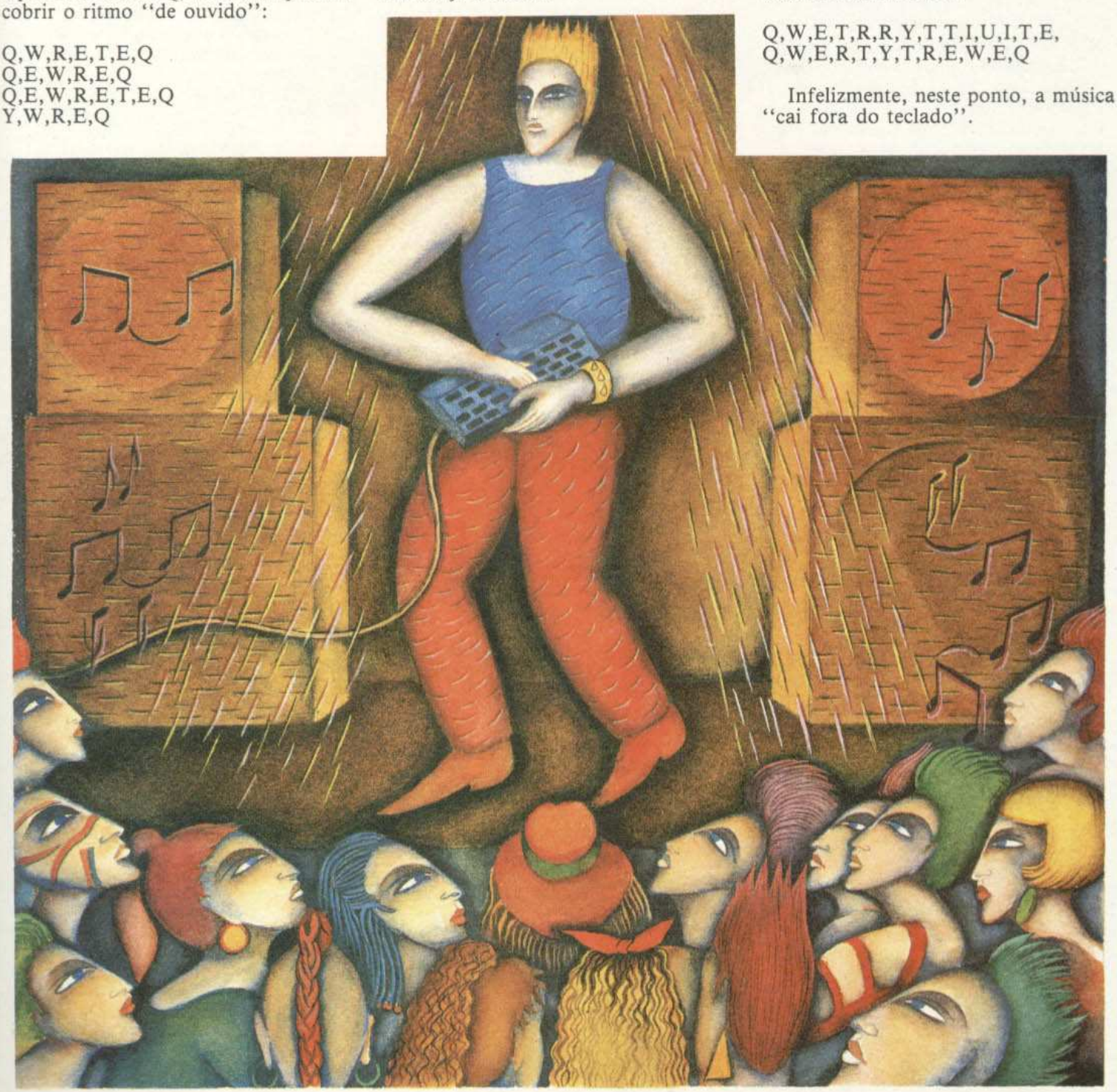
Executemos a seguir o tema do filme *A Noviça Rebelde*:

Q,W,E,Q,E,Q,E
W,E,R,R,E,W,R
E,R,T,E,T,E,T
R,T,Y,Y,T,R,Y
T,Q,W,E,R,T,Y
Y,W,E,5,T,Y,U
U,E,5,6,Y,U,I
I,U,Y,R,U,T,I

Nessa melodia existem apenas três notas fora da escala (perto do final). Aqui estão as teclas para tocar o início de *Jesus Alegria dos Homens*, de Johann Sebastian Bach:

Q,W,E,T,R,R,Y,T,T,I,U,I,T,E,
Q,W,E,R,T,Y,T,R,E,W,E,Q

Infelizmente, neste ponto, a música "cai fora do teclado".



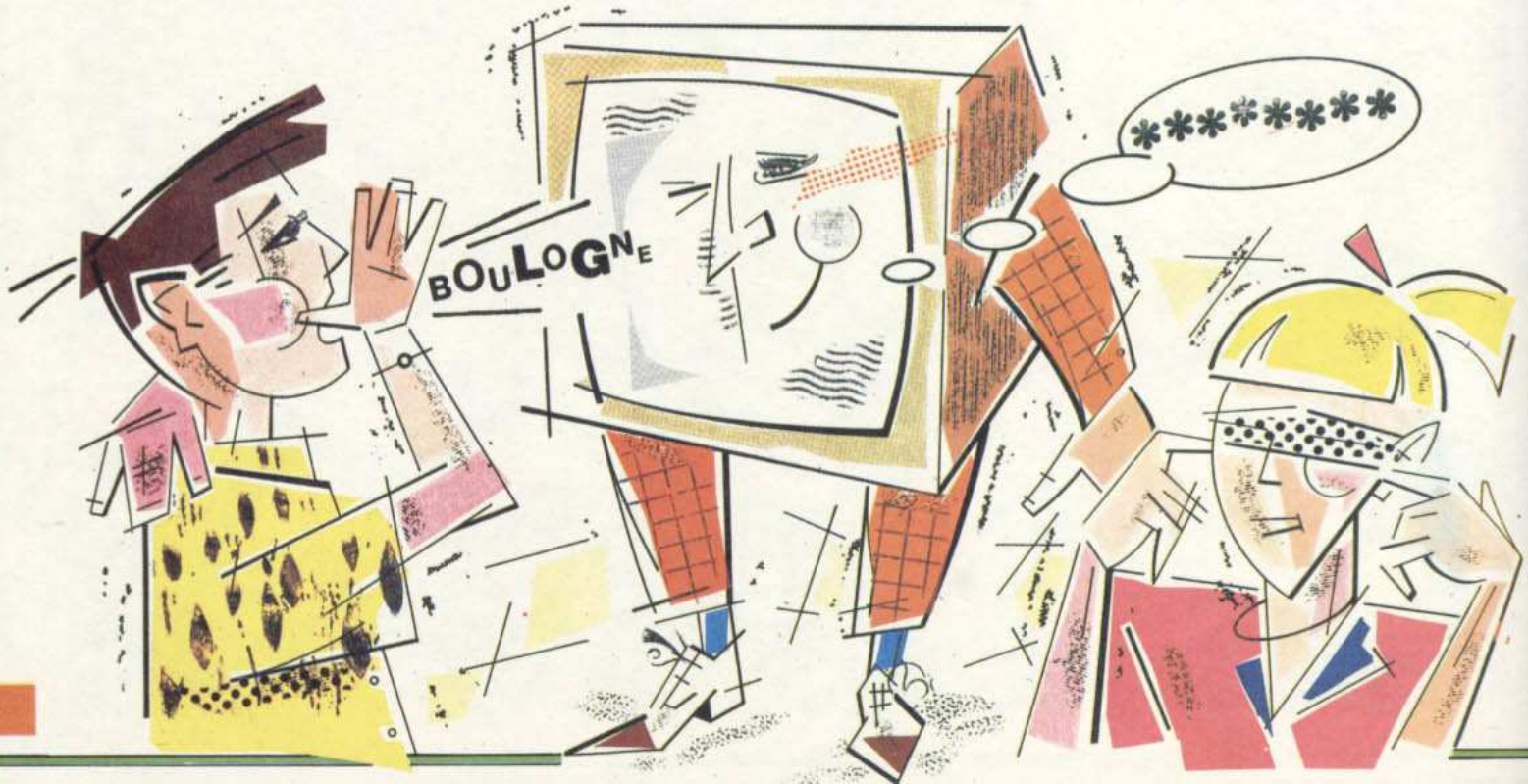
COMPLETE O JOGO DE PALAVRAS

Nonada. É com essa palavra misteriosa que tem início Grande Sertão: Veredas, de Guimarães Rosa. Surpreenda seus adversários com expressões como essa, jogando o jogo de palavras.

Neste artigo completaremos o jogo de palavras começado na lição anterior. Nele você encontrará tudo o que ainda é necessário para começar a jogar. Digite as linhas que faltam e veja algumas aplicações interessantes para os comandos de manipulação de cordões alfanuméricos do computador. Depois, desafie seus amigos para adivinhar palavras bem complicadas.

Aqui estão as rotinas para cada uma das opções do jogo: comprar letras, adivinhar uma letra em uma posição específica da frase, e adivinhar a frase inteira. O programa também faz a contagem de pontos assim como do número de tentativas e de jogadas.

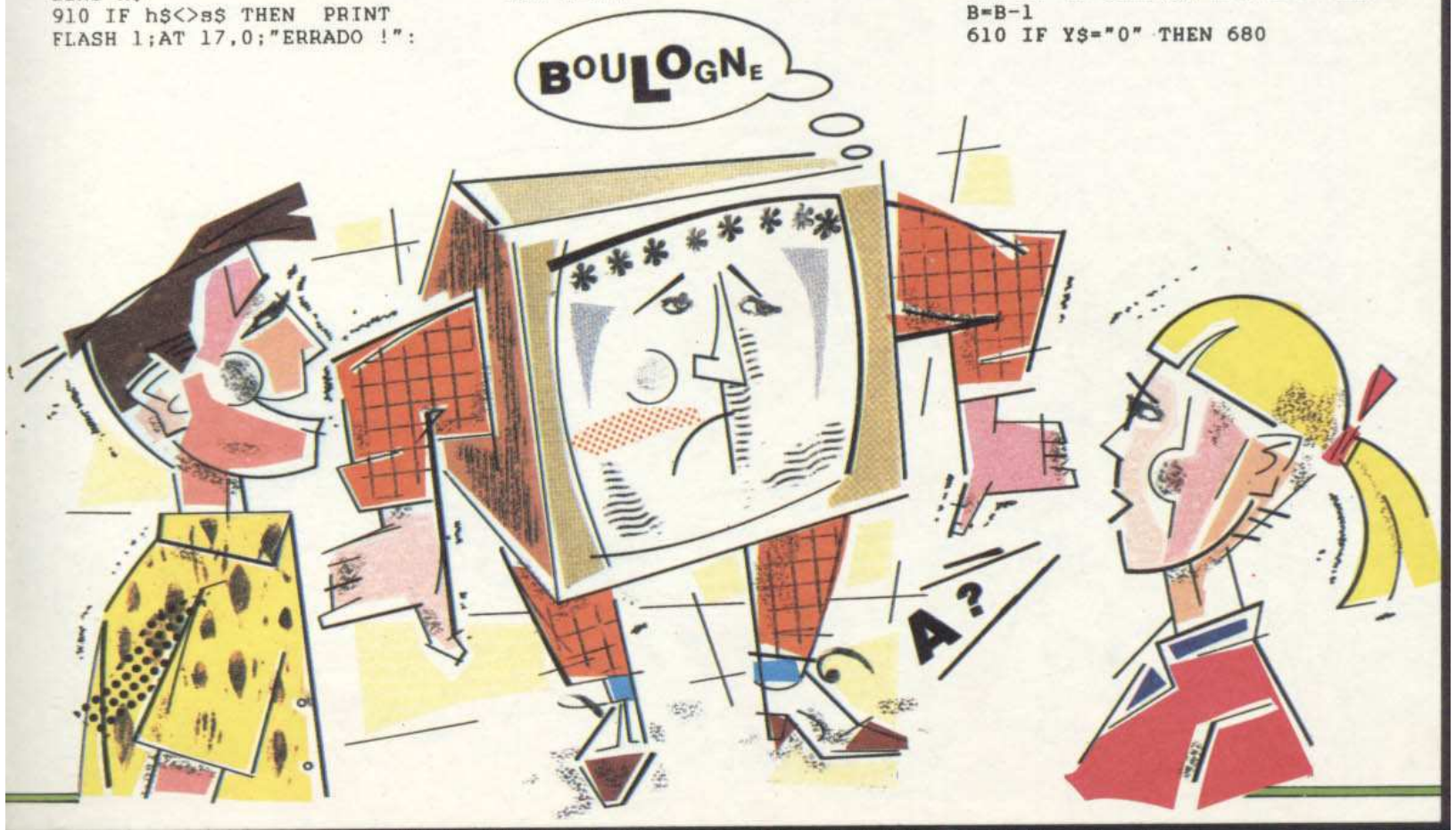
```
370 IF d$<>"XX" AND d$<>"ZZ"
AND LEN d$>1 THEN  GOTO 360
380 IF d$=CHR$ 32 THEN  GOTO
410
385 IF d$="ZZ" THEN  LET d$=""
: GOTO 900
390 IF CODE d$<65 OR CODE d$>
90 THEN  GOTO 360
400 IF d$="XX" THEN  LET d$=""
: GOTO 500
410 GOSUB 790
420 LET e=0
430 LET e=e+1
440 IF e=l+1 THEN  LET tb=tb-q
: LET q$(m TO m+7)="": PRINT
AT 4,0;q$: LET d$="": GOTO 470
450 IF s$(e)<>d$ THEN  GOTO
 430
460 IF s$(e)=d$ THEN  LET z$(e
)=d$: GOTO 430
470 PAUSE 100: PRINT AT 14,0;:
FOR r=1 TO 7: PRINT "
     ": NEXT r
480 PRINT  PAPER 2; INK 6;AT 1
,22;tb;CHR$ 32: PRINT  PAPER 2
; INK 6;AT 14,0;z$: PRINT " TE
NTATIVA ";f: LET f=f+1: IF s$=
z$ THEN  GOTO 730
490 GOTO 360
500 INPUT "QUAL CARACTER QUER
ADIVINHAR ?", LINE d$
510 IF LEN d$>1 THEN  GOTO 500
520 IF d$=CHR$ 32 THEN  GOSUB
790: GOTO 550
530 IF CODE d$<65 OR CODE d$>
90 THEN  GOTO 500
540 GOSUB 790
550 PRINT  PAPER 2; INK 6;AT
18,0;d$: PRINT AT 18,2;"EM QUA
L POSICAO ? - Use as      tecl
as  DIREITA e ESQUERDA      e
'0' para definir."
560 PRINT  PAPER 2; INK 6;AT
14,0;z$: PRINT  PAPER 6; INK 2
;AT 14,b;z$(b+1)
570 PAUSE 0: LET y$=INKEY$: IF
y$="" THEN  GOTO 570
590 IF y$="8" AND b<1-1 THEN
LET b=b+1
600 IF y$="5" AND b>0 THEN
LET b=b-1
610 IF y$="0" THEN  GOTO 680
640 IF b>=32 THEN  LET w=15:
LET v=b-32
650 IF b<32 THEN  LET w=14:
LET v=b
660 PRINT  PAPER 2; INK 6;AT
14,0;z$: PRINT  PAPER 6; INK 2
;AT w,v;z$(b+1)
670 GOTO 570
680 IF z$(b+1)<>"*" THEN  GOTO
570
690 IF s$(b+1)<>d$ THEN  LET
tb=tb-q/2: PRINT  FLASH 1;AT
17,0;"QUE AZAR!": PAUSE 50:
LET b=0: GOTO 470
700 IF s$(b+1)=d$ THEN  PRINT
FLASH 1;AT 17,0;"MUITO BEM !":
PAUSE 50: LET z$(b+1)=d$: LET
tb=tb+q: LET b=0
710 IF s$=z$ THEN  GOTO 730
```



- NOVAS ROTINAS PARA CONCLUIR O JOGO DE PALAVRAS
- COMPRE LETRAS
- VERIFICAÇÃO DO PALPITE DO JOGADOR
- COMO ADIVINHAR UMA LETRA EM POSIÇÃO ESPECÍFICA
- ADIVINHE A FRASE COMPLETA
- A INSTRUÇÃO **CALL** NO APPLE E NO TK-2000

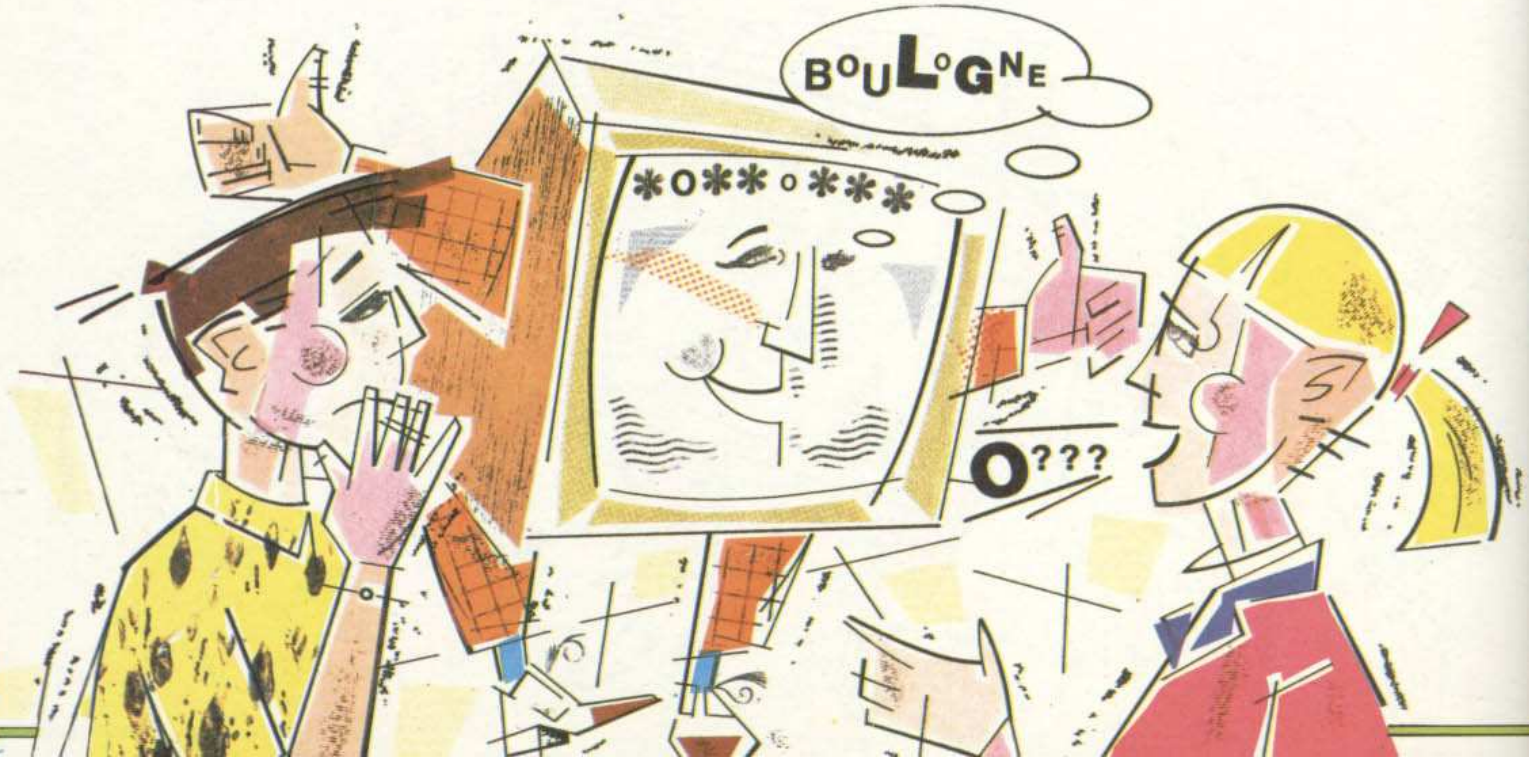
```
720 GOTO 470
730 PRINT  INK 6; PAPER 2;AT 1
,22;tb: PRINT AT 17,0;"PARABEN
S, ";b$;TAB 0;TAB 31;" ":
PAUSE 100: CLS
740 LET k=k+1: IF k=t*2 THEN
GOTO 880
750 LET c$=a$: LET a$=b$: LET
b$=c$
760 LET tc=ta: LET ta=tb: LET
tb=tc
770 LET q$="": LET d=0: LET
f=1
780 GOTO 160
790 LET m=(CODE d$-64)*8-7
800 IF m=-263 THEN  LET m=209
810 IF q$(m TO m+5)="" THEN
GOTO 360
820 LET q=VAL q$(m+2 TO m+3)
830 RETURN
880 IF ta>tb THEN  CLS : PRINT
a$;" VENCEU POR ";ta;" A ";tb
890 IF tb>ta THEN  CLS : PRINT
b$;" VENCEU POR ";tb;" A ";ta
892 IF ta=tb THEN  CLS : PRINT
" O RESULTADO FOI UM EMPATE !"
895 STOP
900 INPUT "INTRODUZA A FRASE",
LINE h$
910 IF h$<>s$ THEN  PRINT
FLASH 1;AT 17,0;"ERRADO !":
PAUSE 50: LET tb=tb-50: PRINT
INK 6; PAPER 2;AT 1,22;tb:
PRINT AT 15,0;"TENTATIVA ";f:
LET f=f+1: GOTO 360
920 FOR n=1 TO 1: LET d$=z$(n)
: IF d$<>"*" THEN  GOTO 950
930 LET m=(CODE s$(n)-64)*8-7:
IF m=-263 THEN  LET m=209
940 LET tb=tb+VAL q$(m+2 TO m+
3)
950 NEXT n: GOTO 730
```

```
360 PRINT @384,"";:LINE INPUT "
XX=ADIVINHAR LETRA
 ZZ=ADIVINHAR A FRASE
  A-Z=COMPRAR O CARACTER  ?";D$
370 IF D$<>"XX" AND D$<>"ZZ" AN
D LEN(D$)>1 THEN 360
380 IF D$=CHR$(32) THEN 410
385 IF D$="ZZ" THEN D$="":GOTO
1000
390 IF D$<"A" OR D$>"Z" THEN360
400 IF D$="XX" THEN D$="":GOTO
500
410 GOSUB 790
420 E=0
430 E=E+1
440 IF E=L+1 THEN TB=TB-G:MID$(
Q$,M,8)="        ":PRINT @96,Q$
:D$="":GOTO 470
450 IF MID$(S$,E,1)<>D$ THEN430
460 IF MID$(S$,E,1)=D$ THEN MID
$(Z$,E,1)=D$:GOTO 430
470 GOSUB 950:PRINT @448:PRINT
@416:PRINT @480,STRING$(31,32);
480 PRINT @54,TB:PRINT @352,Z$:
PRINT @320,"TENTATIVA";F:F=F+1:
IF S$=Z$ THEN 730
490 GOTO 360
500 PRINT @448:PRINT @416:PRINT
 @448,"";:LINE INPUT "QUER ADIV
INHAR QUAL CARACTER ?";D$
510 IF LEN(D$)>1 THEN 500
520 IF D$=CHR$(32) THEN GOSUB 7
90:GOTO 550
530 IF D$<"A" OR D$>"Z" THEN500
540 GOSUB 790
550 PRINT @448,"EM QUAL POSICAO
? - USE SETAS E  '0' PARA DEFIN
IR.";
560 PRINT @352,Z$:POKE 1024+352
+B,(ASC(MID$(Z$,B+1,1))AND 191)
570 Y$=INKEY$:IF Y$="" THEN 570
590 IF Y$=CHR$(9) AND B<L-1 THE
N B=B+1
600 IF Y$=CHR$(8) AND B>0 THEN
B=B-1
610 IF Y$="0" THEN 680
```



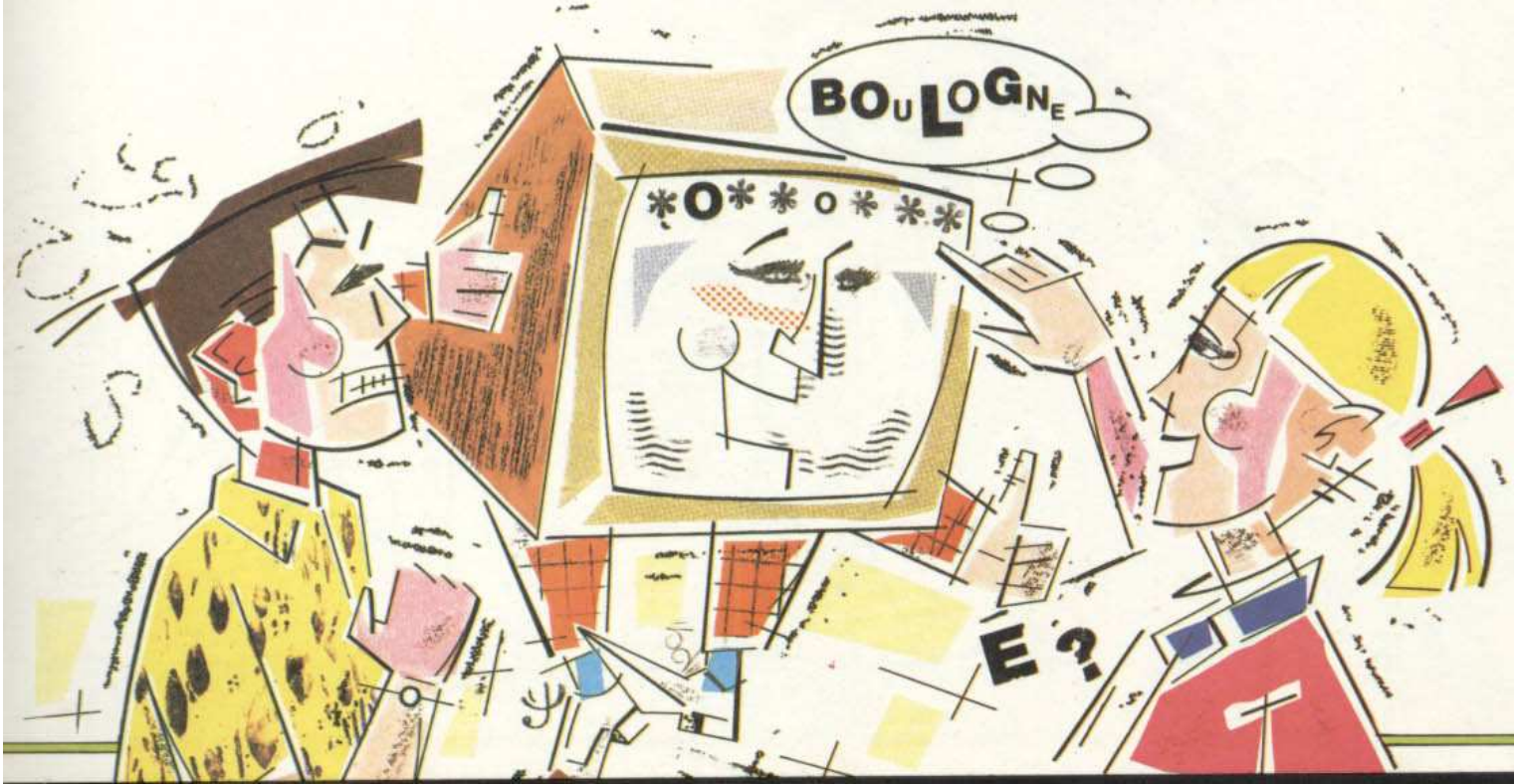
```
660 PRINT @352,Z$:POKE 352+1024
+B,(ASC(MID$(Z$,B+1,1))AND 191)
670 GOTO 570
680 PRINT @480,STRING$(31,32);:
IF MID$(Z$,B+1,1)=D$ THEN TB=TB
-G:PRINT @448,"TRAPACEIRO!":FOR
 DE=1 TO 100:NEXT:B=0:GOTO 470
690 IF MID$(S$,B+1,1)<>D$ THEN
TB=TB-G/2:PRINT @448,"QUE AZAR!
":FOR DE=1 TO 100:NEXT:B=0:GOTO
 470
700 IF MID$(S$,B+1,1)=D$ THEN P
RINT @448,"MUITO BEM!":FOR DE=1
 TO 100:NEXT:MID$(Z$,B+1,1)=D$:
TB=TB+G:B=0
710 IF S$=Z$ THEN 730
720 GOTO 470
730 PRINT @480,"PARABENS, ";B$;
:GOSUB 950:CLS
740 K=K+1:IF K=T*2 THEN 880
750 C$=A$:A$=B$:B$=C$
760 TC=TA:TA=TB:TB=TC
770 Q$="":D=0:F=1
780 GOTO 160
790 M=(ASC(D$)-64)*8-7
800 IF M=-263 THEN M=209
810 IF MID$(Q$,M,6)="      "THE
N 360
820 G=VAL(MID$(Q$,M+2,2))
830 RETURN
880 IF TA>TB THEN CLS:PRINT A$;
" VENCEU POR";TA;"A";TB
890 IF TB>TA THEN CLS:PRINT B$;
" VENCEU POR";TB;"A";TA
892 IF TA=TB THEN CLS:PRINT"O R
ESULTADO FOI UM EMPATE !"
895 END
1000 PRINT @448:PRINT @416:PRIN
T @416,"INTRODUZA A FRASE -":LI
NE INPUT GU$
1010 IF GU$<>S$ THEN PRINT @416
,"ERRADO !":TB=TB-50:PRINT @54,
TB:GOSUB 950:PRINT @320,"TENTAT
IVA";F;:F=F+1:GOTO 360
1020 FOR N=1 TO L:D$=MID$(Z$,N,
1):IF D$<>"*" THEN 1050
1030 M=(ASC(MID$(S$,N,1))-64)*8
-7:IF M=-263 THEN M=209
1040 TB=TB+VAL(MID$(Q$,M+2,2))
1050 NEXT N:GOTO 730
```

```
167 S$=""
360 LOCATE 0,21:PRINTSPC(77):LO
CATE 0,21:PRINT"XX-Adivinha let
ra    ZZ-Adivinha frase":INPUT"
          A-Z Compra letra ";D$
370 IF D$<>"XX" AND D$<>"ZZ" AN
D LEN(D$)>1 THEN 360
380 IF D$="" THEN D$="_":GOTO 4
10
385 IF D$="ZZ" THEN D$="":GOTO
1000
390 IF (D$<"A" OR D$>"Z") AND D
$<>"Ç" THEN 360
400 IF D$="XX" THEN D$="":GOTO
500
410 GOSUB 790
420 E=0
430 E=E+1
440 IF E=L+1 THEN TB=TB-G:MID$(
Q$,M,7+(N/5=INT(N/5)))=STRING$(
8,32):LOCATE 0,4:PRINTQ$:D$="":
GOTO 470
450 IF MID$(S$,E,1)<>D$ THEN 43
0
460 IF MID$(S$,E,1)=D$ THEN MID
$(Z$,E,1)=D$:GOTO 430
470 GOSUB 950
480 LOCATE25,1:PRINTTB;"pontos"
:LOCATE 0,12:PRINTZ$:LOCATE 0,1
5:PRINT"Tentativa ";F:F=F+1:IF
S$=Z$ THEN 730
490 GOTO 360
500 LOCATE 0,21:PRINTSPC(77):LO
CATE 0,21:INPUT"QUAL A LETRA";D
$
510 IF LEN(D$)>1 THEN 500
520 IF D$="" THEN D$=CHR$(32):G
OSUB 790:GOTO 550
530 IF (D$<"A" OR D$>"Z") AND D
$<>"Ç" THEN 500
540 GOSUB 790
550 LOCATE 0,21:PRINTSPC(77):LO
CATE 0,21:PRINT"INDIQUE A POSIÇ
ÃO USANDO AS SETAS        PARA
MARCAR TECLE <0>"
555 B=0:V=12:BB=0
560 LOCATE BB,V,1
570 Y$=INKEY$:IF Y$="" THEN 570
590 IF Y$=CHR$(28) AND B<L-1 TH
EN B=B+1
600 IF Y$=CHR$(29) AND B>0 THEN
 B=B-1
605 BB=B:V=12:IF B>39 THEN BB=B
-40:V=13LIST 600-
610 IF Y$="0" THEN 680
670 GOTO 560
680 IF MID$(Z$,B+1,1)=D$ THEN T
B=TB-G:LOCATE 0,21:PRINTSPC(77)
:LOCATE 0,21:PRINT"TRAPACEIRO!"
:GOSUB 950:GOTO 470
690 IF MID$(S$,B+1,1)<>D$ THEN
TB=TB-G/2:LOCATE 0,21:PRINTSPC(
77):LOCATE 0,21:PRINT"ERROU DES
TA VEZ...":GOSUB 950:GOTO 470
700 IF MID$(S$,B+1,1)=D$ THEN L
OCATE 0,21:PRINTSPC(77):LOCATE
0,21:PRINT"BOM PALPITE!":MID$(Z
$,B+1,1)=D$:GOSUB 950:TB=TB+G
710 IF S$=Z$ THEN 730
720 GOTO 470
730 LOCATE 0,21:PRINTSPC(77):LO
CATE 0,21:PRINT"PARABENS! ACERT
OU!":BEEP:GOSUB 950:BEEP:CLS
740 K=K+1:IF K=T*2 THEN 880
750 SWAP A$,B$
760 SWAP TA,TB
770 Q$="":D=0:F=1:S$=""
780 GOTO 160
790 M=INSTR(Q$,D$)
800 IF D$="_" THEN D$=CHR$(32)
810 IF MID$(Q$,M,5)=STRING$(5,3
2) THEN 360
```



```
820 G=VAL(MID$(Q$,M+2,2))
830 RETURN
880 IF TA>TB THEN CLS:PRINTA$;"
 GANHOU POR";TA;"PONTOS A";TB
890 IF TB>TA THEN CLS:PRINTB$;"
 GANHOU POR";TB;"PONTOS A";TA
892 IF TA=TB THEN CLS:PRINT"O J
OGO TERMINOU EMPATADO EM":PRINT
TA;"PONTOS"
895 END
950 FOR DE=1 TO 1000:NEXT:RETUR
N
1000 LOCATE 0,21:PRINTSPC(77):L
OCATE 0,21:PRINT"DIGITE A FRASE
 COMPLETA:":LINEINPUT GU$
1010 IF GU$<>S$ THEN LOCATE 0,2
1:PRINTSPC(7):LOCATE 0,21:PRINT
"ERRADO!":TB=TB-50:LOCATE 25,1:
PRINTTB;"pontos":GOSUB 950:LOCA
TE 0,15:PRINT"TENTATIVA";F:F=F+
1:GOTO 360
1020 FOR N=1 TO L:D$=MID$(Z$,N,
1):IF D$<>"*" THEN 1050
1030 IF D$=CHR$(32) THEN D$="_"
:M=INSTR(Q$,D$)
1040 TB=TB+VAL(MID$(Q$,M+2,2))
1050 NEXT:GOTO 730
```

```
370  IF D$ <  > "ZZ" AND D$ <
> "XX" AND  LEN (D$) > 1 THEN 3
60
380  IF D$ = "" THEN D$ =  CHR$
 (32): GOTO 410
385  IF D$ = "ZZ" THEN D$ = "":
 GOTO 1000
390  IF D$ < "A" OR D$ > "Z" TH
EN 360
400  IF D$ = "XX" THEN D$ = "":
 GOTO 500
410  GOSUB 790
420 E = 0
430 E = E + 1
440  IF E = L + 1 THEN TB = TB
- G:Q$ =  LEFT$ (Q$,M - 1) + "
      " +  MID$ (Q$,M + 8): VT
AB 5: PRINT Q$:D$ = "": GOTO 47
0
450  IF  MID$ (S$,E,1) <  > D$
THEN 430
460  IF  MID$ (S$,E,1) = D$ THE
N Z$ =  LEFT$ (Z$,E) + D$ +  MI
D$ (Z$,E + 2): GOTO 430
470  FOR DE = 1 TO 300: NEXT
480  VTAB 2: HTAB 25: PRINT TB;
" PONTOS  ": VTAB 12: HTAB 40:
PRINT Z$: VTAB 16: PRINT "TENTA
TIVA ";F:F = F + 1: IF S$ =  MI
D$ (Z$,2) THEN 710
490  GOTO 360
500  VTAB 22: CALL  - 958: INPU
T "QUAL A LETRA ";D$
510  IF  LEN (D$) > 1 THEN 500
520  IF D$ = "" THEN D$ = " ":
GOSUB 790: GOTO 550
530  IF D$ < "A" OR D$ > "Z" TH
EN 500
540  GOSUB 790
550  VTAB 22: CALL  - 958: PRIN
T "COLOQUE O MARCADOR SOB A POS
ICAO DESEJADA E TECLE [ENTER]";
:B = 1
560  VTAB 14: CALL  - 868: HTAB
 B: PRINT  CHR$ (94)
570  GET Y$: IF Y$ =  CHR$ (13)
 THEN 680
590  IF Y$ =  CHR$ (8) AND B >
1 THEN B = B - 1
600  IF Y$ =  CHR$ (21) AND B <
 L THEN B = B + 1
670  GOTO 560
680  VTAB 14: CALL  - 958: VTAB
 22: IF  MID$ (Z$,B,1) = D$ THE
N  PRINT "TRAPACEIRO!";:TB = TB
 - G: FOR DE = 1 TO 300: NEXT :
 GOTO 470
690  IF  MID$ (S$,B,1) <  > D$
THEN TB = TB - G / 2: PRINT "ER
ROU DESTA VEZ...";: FOR DE = 1
TO 300: NEXT : GOTO 470
700  IF  MID$ (S$,B,1) = D$ THE
N  PRINT "BOA! ";:TB = TB + G:Z
$ =  LEFT$ (Z$,B) + D$ +  MID$
(Z$,B + 2): FOR DE = 1 TO 300:
NEXT
710  IF S$ =  MID$ (Z$,2) THEN
 FLASH : VTAB 13: PRINT S$: CAL
L  - 868: NORMAL : GOTO 730
720  GOTO 470
730  VTAB 22: CALL  - 958: PRIN
T "PARABENS!": GOSUB 950: HOME
740 J = J + 1: IF J = T * 2 THE
N 880
750 C$ = A$:A$ = B$:B$ = C$
760 TC = TA:TA = TB:TB = TC
770 Q$ = "":D = 0:F = 1
780  GOTO 160
790 M = ( ASC (D$) - 64) * 8 -
6
800  IF D$ =  CHR$ (32) THEN M
= 218
805  IF D$ = "Z" THEN M = 210
810  IF  MID$ (Q$,M,2) = "  " T
HEN  POP : GOTO 360
820 G =  VAL ( MID$ (Q$,M + 2,2
))
830  RETURN
880  IF TA > TB THEN  HOME : PR
INT : PRINT A$;" GANHOU POR ";T
A;" PONTOS A ";TB
890  IF TA < TB THEN  HOME : PR
INT : PRINT B$;" GANHOU POR ";T
B;" PONTOS A ";TA
892  IF TA = TB THEN  HOME : PR
INT : PRINT "O JOGO TERMINOU EM
PATADO EM ": PRINT TA;" PONTOS!
"
895  END
950  FOR DE = 1 TO 3000: NEXT :
 RETURN
1000  VTAB 22: CALL  - 958: INP
UT "ENTRE A FRASE: ";GU$
1010  IF GU$ <  > S$ THEN  VTAB
 22: CALL  - 958: PRINT "ERRADO
```



Por que as linhas em que há a instrução CALL diferem nos programas do Apple e do TK-2000?

Porque a instrução **CALL** seguida de um certo número de linhas serve para acionar uma rotina em linguagem de máquina. No nosso caso, essas linhas constituem rotinas intrínsecas do computador, ou seja, são rotinas próprias da máquina.

No Apple, a instrução **CALL −958** faz com que a tela seja apagada desde o cursor até a última posição de vídeo, sem alteração na posição do cursor. No TK-2000, ela faz a mesma coisa, mas com uma diferença: agora, o cursor é colocado na primeira posição da tela. Assim, ele deve ser reposicionado após o comando.

Já a instrução **CALL −868** provoca, no Apple, o apagamento da linha em que está o cursor a partir da posição deste. A mesma instrução no TK-2000 não tem um efeito necessário para o nosso programa.

```
!":TB = TB - 50: VTAB 2: HTAB 2
5: PRINT TB;" PONTOS": VTAB 16:
 PRINT "TENTATIVA ";F:F = F + 1
: FOR DE = 1 TO 300: NEXT : GOT
O 360
1020  FOR N = 2 TO L + 1:D$ =
MID$ (Z$,N,1): IF D$ <  > "*" T
HEN 1050
1030 M = ( ASC ( MID$ (S$,N - 1
,1)) - 64) * 8 - 6: IF M =  - 2
62 THEN M = 218
1035  IF M = 202 THEN M = M + 8

1040 TB = TB +  VAL ( MID$ (Q$,
M + 2,2))
1050  NEXT :Z$ = " " + S$: GOTO
 710
```

Para executar o programa no TK-2000, faça as seguintes modificações:

```
500  VTAB 22: CALL  - 958: VTAB
 22: INPUT "QUAL A LETRA ";D$
550  VTAB 22: CALL  - 958: VTAB
 22: PRINT "COLOQUE O MARCADOR
SOB A POSICAO DESEJADA E TECLE
[ENTER]";:B = 1
560  VTAB 14: PRINT  SPC( 40):
VTAB 14: HTAB B: PRINT  CHR$ (9
4)
680  VTAB 14: CALL  - 958: VTAB
 22: IF  MID$ (Z$,B,1) = D$ THE
N  PRINT "TRAPACEIRO!";:TB = TB
 - G: FOR DE = 1 TO 300: NEXT :
 GOTO 470
710  IF S$ =  MID$ (Z$,2) THEN
 VTAB 13: PRINT S$: GOTO 730
730  VTAB 22: CALL  - 958: VTAB
 22: PRINT "PARABENS!": GOSUB 9
50: HOME
1000  VTAB 22: CALL  - 958: VTA
B 22: INPUT "ENTRE A FRASE: ";G
U$
1010  IF GU$ <  > S$ THEN  VTAB
 22: CALL  - 958: VTAB 22: PRIN
T "ERRADO!":TB = TB - 50: VTAB
2: HTAB 25: PRINT TB;" PONTOS":
 VTAB 16: PRINT "TENTATIVA ";F:
F = F + 1: FOR DE = 1 TO 300: N
EXT : GOTO 360
```

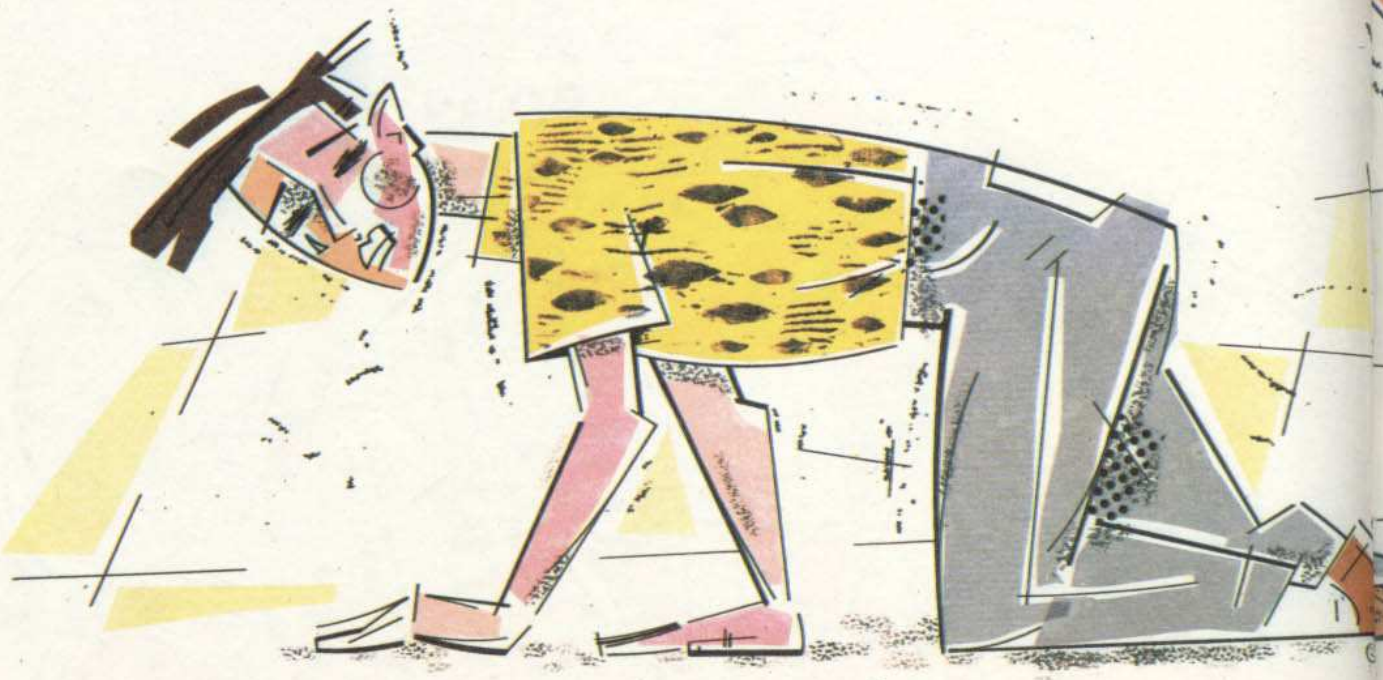
Como na parte anterior da listagem, existem pequenas variações entre as máquinas, mas, de maneira geral, os programas são muito parecidos.

As linhas 370 a 410 manipulam a opção do jogador — comprar letras, adivinhar letras ou mesmo descobrir a frase toda. A rotina identifica o tipo de palpite e evita que sejam feitas entradas ilegais.

COMO COMPRAR LETRAS

Se o jogador decidir comprar uma letra, o computador verificará imediatamente qual o valor dessa letra. Para isso, é acionada uma sub-rotina que começa na linha 790. Esta encontra o valor ASCII da letra para saber até que posição da tabela de valores deve ir. Se o computador encontrar um espaço em branco na posição, isso significa que a letra já foi comprada e o programa voltará à linha 360. Do contrário, será lido o valor da letra, por intermédio da função **VAL**. Esse valor é utilizado para calcular o novo número de pontos do jogador.

A rotina que cuida da compra de letras começa na linha 430. As linhas 430 a 460 passam pela frase procurando ocorrências da letra. A cadeia de aste-



riscos é atualizada, substituindo-se os asteriscos pela letra comprada, quando isso ocorrer. A nova cadeia é então mostrada na tela e o valor da letra subtraído do total de pontos. O número de tentativas é incrementado de 1. A linha 440 apaga a letra escolhida da tabela, indicando que ela não está mais disponível.

ADIVINHE LETRAS EM UMA POSIÇÃO

Se o jogador quiser tentar acertar uma letra em posição específica, deve primeiro selecionar XX. Esse procedimento faz o programa pular para a linha 500. O jogador deve então dizer que letra vai ser usada. Várias verificações evitam que se faça uma entrada ilegal. Em seguida, é preciso mover o marcador para a posição desejada e pressionar 0 (no Apple, pressione <**ENTER**>). O programa verifica se a letra está disponível e faz um teste para ver se o palpite é correto.

Caso o jogador erre o palpite, a linha 690 enviará uma mensagem, subtraindo metade do valor da letra do total de pontos. Se letra e posição coincidirem, a linha 700 emitirá uma mensagem de felicitações e somará o valor da letra ao total de pontos do jogador. Quando a frase é completada, a linha 710 envia o programa para a 730, que dá a boa notícia ao jogador.

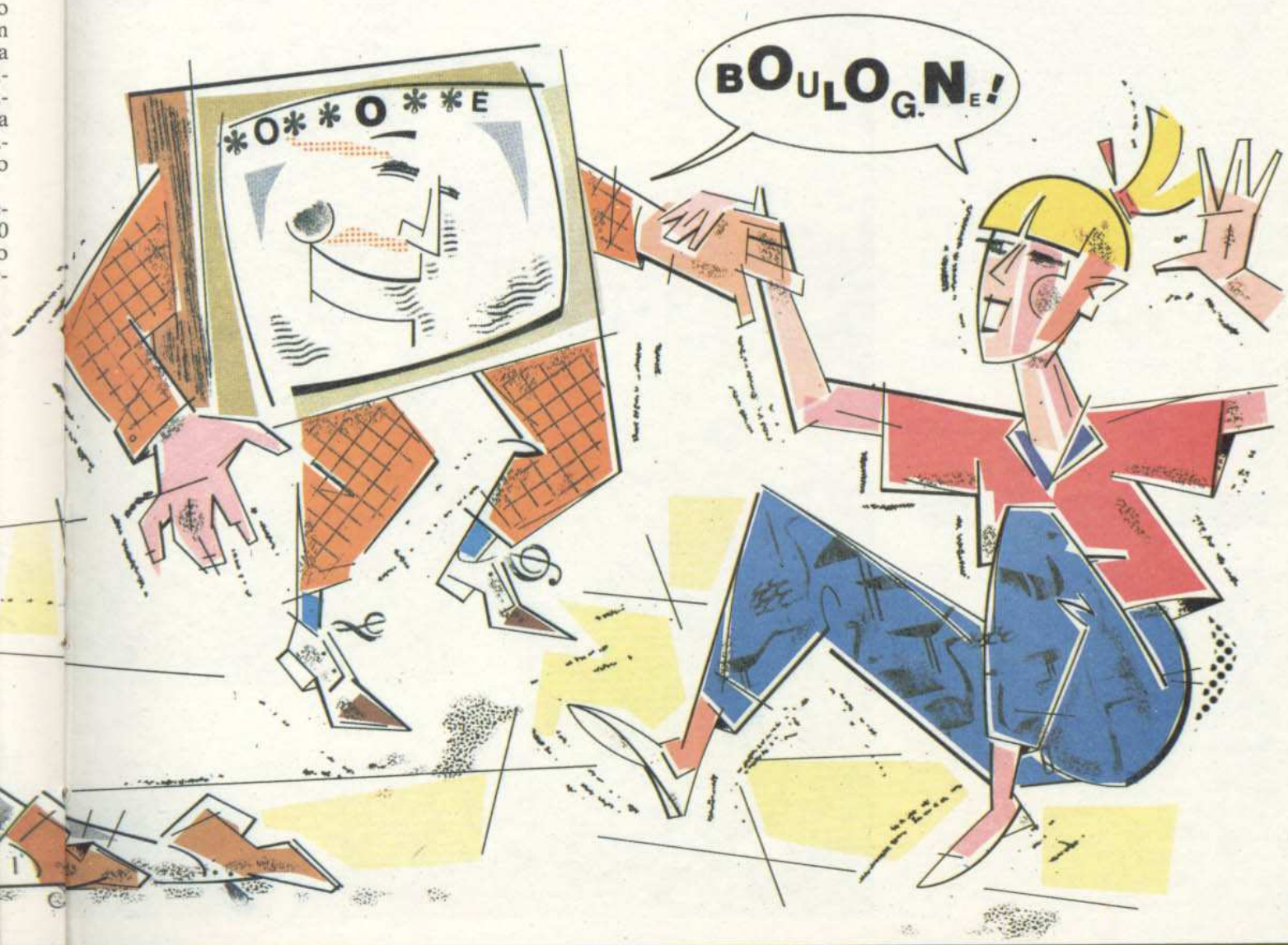
A FRASE COMPLETA

Se o jogador for mais ambicioso, pode querer adivinhar a frase inteira. Para isso, deve primeiro digitar ZZ. A linha 385 manda o programa para a linha 900 (no Spectrum) ou 1000 (nos outros micros). A rotina pede que a frase seja digitada e a compara com a original. Se elas não forem iguais, o jogador perderá cinqüenta pontos e o contador de tentativas será incrementado. Se o palpite for correto, o valor de todas as letras ainda não adivinhadas será somado ao total de pontos, e a linha 730 avisará o jogador que a frase está correta.

FIM DO JOGO

Adivinhada a frase, o programa verifica o número de jogadas realizadas. Este não pode ser maior que o determinado no início do jogo. Se o número ainda for menor, o próximo jogador será chamado, depois que uma nova frase tenha sido introduzida. Quando o jogo acaba, o programa passa à linha 880. Os pontos são comparados e o resultado final é apresentado.

O jogo termina aqui, mas você pode implementar uma rotina do tipo "joga novamente?" para deixá-lo completo.



SÍMBOLOS GRÁFICOS NO TK-2000

Os computadores da linha TK-2000 dispõem de um grande número de caracteres gráficos que podem ser entrados diretamente pelo teclado, ou usados em um programa.

As telas gráficas no computador (GR e HGR) podem ser programadas por meio de diversas instruções extremamente poderosas, em média e alta resolução, tais como **VLIN**, **HLIN**, **PLOT**, **HPLOT**, **DRAW** e outras.

Entretanto, o TK-2000 tem um recurso gráfico adicional, habitualmente pouco explorado, que os compatíveis com a linha Apple não têm. Trata-se dos *caracteres gráficos*, que estão disponíveis para o programador através de dois meios: entrada direta pelo teclado e inserção 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, convencionada internacionalmente, é o chamado código ASCII, que vai de 32 a 126. Os códigos de 0 a 31 são normalmente utilizados em funções de controle do vídeo e dependem do tipo de computador que está sendo usado.

O mesmo acontece com os códigos que vão de 127 a 255. Nessa faixa, os fabricantes utilizam geralmente os códigos para acomodar caracteres gráficos (que podem ser tipos especiais, como naipes de baralho, notas musicais etc., ou blocos gráficos formando linhas, ângulos e cantos).

GRÁFICOS DA ROM

Tais caracteres gráficos são também chamados de *gráficos da ROM*, pois já vêm pré-programados. Nos computadores da linha TK-2000, os caracteres especiais ocupam a faixa da tabela de caracteres que vai de 193 a 242.

Os sinais gráficos que mais nos interessarão neste artigo são os blocos geralmente utilizados na composição de desenhos formados por linhas retas, tais como tabelas ou formulários de entrada. A vantagem desses blocos consiste em simplificar a programação, tornando desnecessária a mistura da tela gráfica com o texto. Tal simplificação deixa aberto o caminho para o emprego de

comandos mais diretos, como **VTAB**, **HTAB**, **PRINT**, **INPUT** etc.

ENTRADA PELO TECLADO

Da mesma maneira que os caracteres convencionais do código ASCII (demarcados sobre as teclas do microcomputador), os sinais gráficos podem ser digitados pelo teclado. Para isso, é necessário, em primeiro lugar, pressionar simultaneamente as teclas **<CONTROL>** e **<B>**. Esse procedimento coloca o teclado em modo gráfico.

Em seguida, deve-se acionar simultaneamente a tecla **<SHIFT>**, e uma das teclas alfanuméricas, ou então as teclas **<SHIFT>**, **<CONTROL>** e uma terceira. Com isso, o caractere desejado aparece diretamente na tela (por exemplo, dentro de uma cadeia alfanumérica).

O manual de programação do TK-2000 exibe um desenho esquemático do teclado, no qual são assinalados todos os caracteres gráficos, em relação à disposição das teclas. Consulte também a tabela que apresentamos a seguir.

Ao se terminar de digitar os caracteres gráficos em uma linha, deve-se pressionar novamente **<CONTROL>** e **<B>**, para sair do modo gráfico.

Por exemplo, se quisermos digitar o símbolo tradicional do naipe de paus, devemos teclar **<CONTROL><B>** e, em seguida, **<SHIFT><R>**.

O programa abaixo demonstra como isso pode ser feito. Ele desenha uma tabela simples na tela, utilizando blocos gráficos e coloca depois os diversos nomes digitados nas linhas da tabela.

```
200 HOME
210 PRINT "┌──────┬──────┐"
220 PRINT "│ NO.  │ NOME │"
230 PRINT "├──────┼──────┤"
240 FOR I=1 TO 10
250 PRINT "│      │      │"
260 NEXT I
270 PRINT "└──────┴──────┘"
275 FOR I=1 TO 10
280 VTAB 20:HTAB 1
290 INPUT "NOME : ";N$
300 VTAB I+3:HTAB 4: PRINT I
310 VTAB I+3:HTAB 10: PRINT N$
320 NEXT I
330 VTAB 20:HTAB 1:STOP
```

Os traços devem ser digitados nesta seqüência:

```
Linha 210:
<SHIFT> A
<SHIFT><CONTROL> F (4 vezes)
<SHIFT> H
<SHIFT><CONTROL> F (11 vezes)
<SHIFT> S
Linha 220:
<SHIFT><CONTROL> C
Linha 230:
<SHIFT><CONTROL> H
<SHIFT><CONTROL> G (4 vezes)
<SHIFT><CONTROL> B
<SHIFT><CONTROL> G (11 vezes)
<SHIFT><CONTROL> N
Linha 250: como a linha 220
Linha 270:
<SHIFT> Z
<SHIFT><CONTROL> F (4 vezes)
<SHIFT> G
<SHIFT><CONTROL> F (11 vezes)
<SHIFT> X
```

Tanto na linha 220 quanto na linha 250, os espaços em branco e as letras podem ser digitados normalmente, sem precisar sair do modo gráfico. Em outras palavras: nas linhas que têm caracteres gráficos (210, 220, 230, 250 e 270), basta pressionar **<CONTROL><B>** uma vez, logo após digitar o sinal de abre aspas; em seguida, teclar os gráficos e/ou as letras, e novamente **<CONTROL><B>**. Só depois disso, digita-se o sinal de fecha aspas.

Existem duas desvantagens nessa forma de entrada de caracteres gráficos: primeiro, as teclas não têm qualquer marcação que auxilie o usuário a encontrar o gráfico correto. Torna-se necessário então consultar o manual, o que faz o processo bastante moroso. Em segundo lugar, o programa não pode ser listado em uma impressora não gráfica, ou que não seja específica para a linha TK-2000.

CARACTERES GRÁFICOS NO PROGRAMA

Existe ainda um outro truque para especificar caracteres gráficos dentro de um programa sem precisar digitá-los diretamente. A função que permite fazer isso é a utilíssima **CHR$**.

Os caracteres normais, especiais e gráficos com códigos na faixa de 32 a 255 podem ser impressos na tela a par-

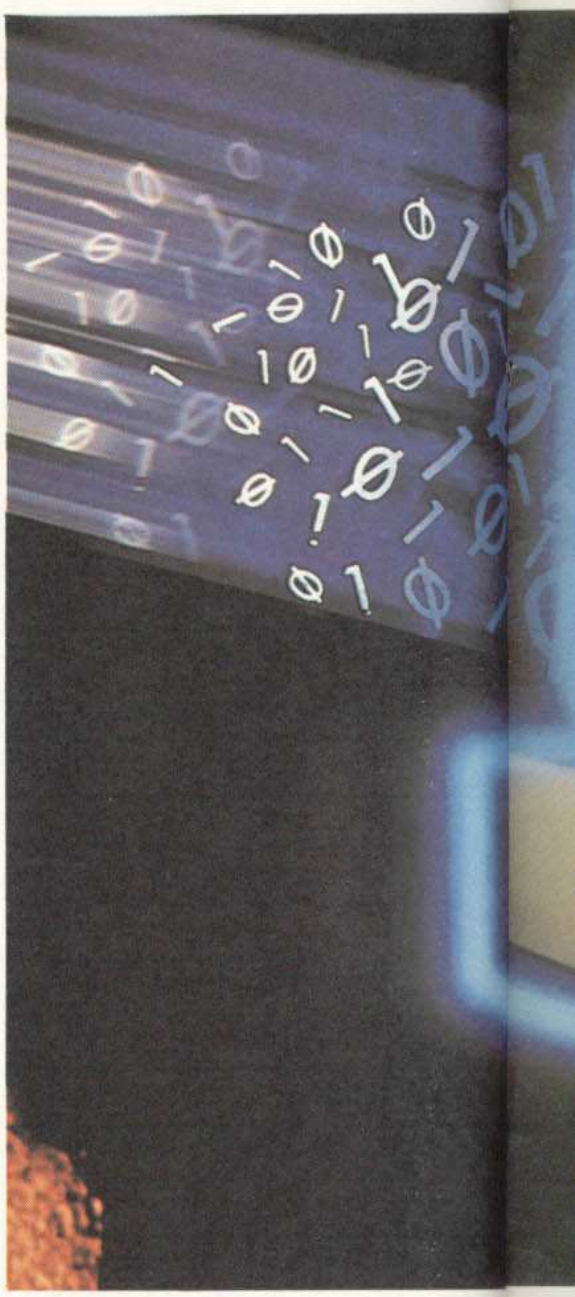
tir de um programa, contendo a **CHR$** acompanhada do número de código do caractere desejado. Por exemplo, para imprimir na tela o símbolo de paus (naipe de baralho), digitamos:

```
PRINT CHR$(242);CHR$(231)
```

Portanto, para combinar códigos gráficos com a função **CHR$**, é preciso "avisar" o computador que o código será usado como gráfico. Para isso, recorremos a dois bytes: **CHR$(242)**, seguido de **CHR$(n)**, onde **n** é o código do caractere gráfico na tabela. O próximo programa mostra a tabela de correspondência entre códigos numéricos e gráficos na tela do TK-2000:

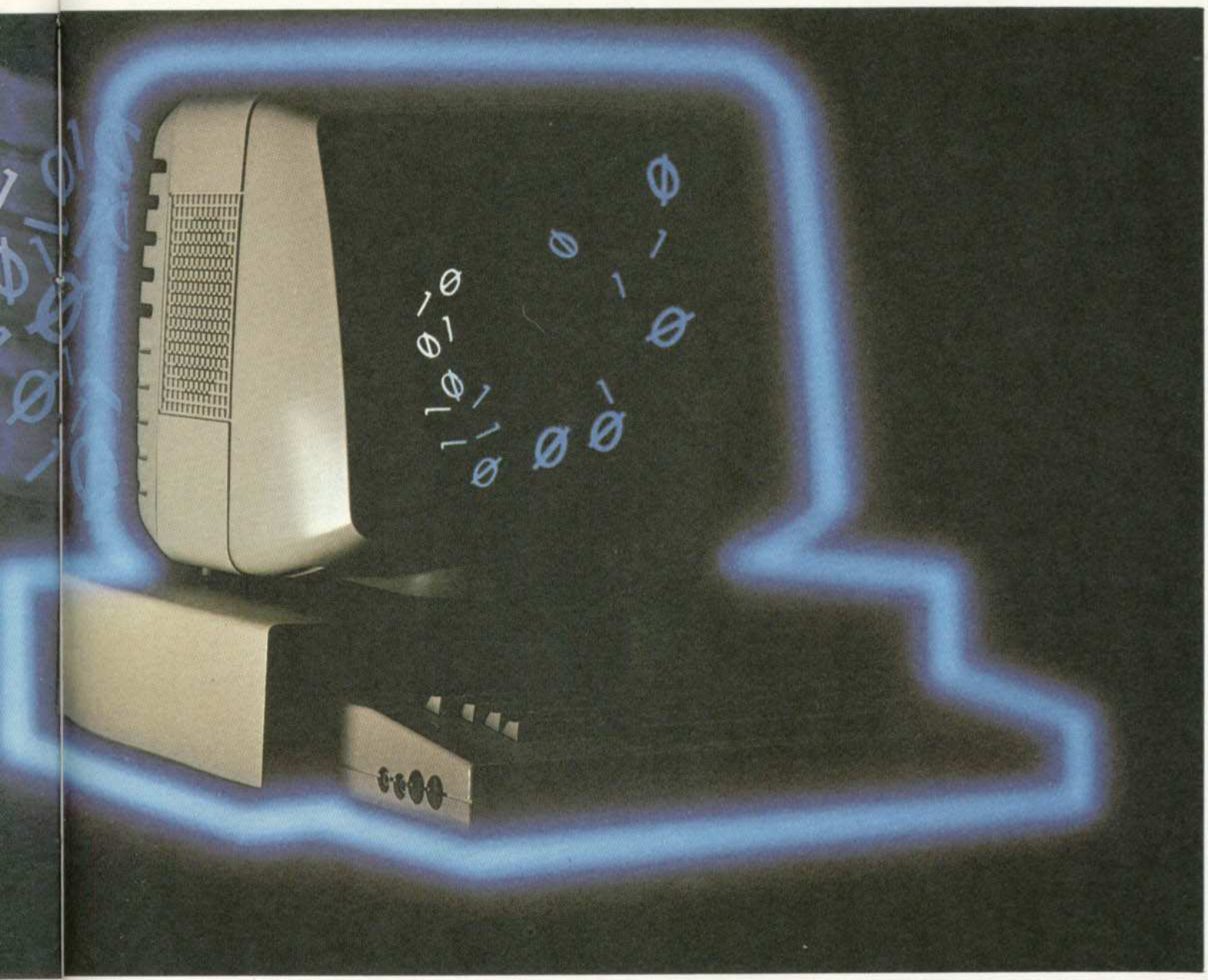
```
10 HOME
20 FOR J=193 TO 242 STEP 6
30 FOR I=J TO J+5
40 PRINT I;" ";CHR$(242);CHR$(
I);" ";
50 NEXT I
60 PRINT
70 NEXT J
80 VTAB 20:HTAB 1: PRINT
"PRESSIONE RETURN"
90 GET A$
```

Os caracteres são impressos ordenadamente em fileiras, por meio dos dois laços que começam nas linhas 20 e 30. O primeiro laço varia **J** de 193 a 242 (faixa de códigos correspondente aos caracteres gráficos), de 6 em 6. O laço seguinte percorre todos os valores entre **J** e **J+5**. O **PRINT** da linha 60 serve para encerrar uma fileira de seis códigos e suas representações gráficas, que são mostradas na linha 40.



CARACTERES GRÁFICOS PARA O TK-2000

Código	Teclas	Caractere	Código	Teclas	Caractere
193	CONTROL SHIFT 1	□	218	CONTROL SHIFT V	❒
194	CONTROL SHIFT 2	▭	219	CONTROL SHIFT C	❒
195	CONTROL SHIFT 3	▬	220	CONTROL SHIFT J	◪
196	CONTROL SHIFT 4	▬	221	CONTROL SHIFT M	◪
197	CONTROL SHIFT 5	▬	222	SHIFT T	⬮
198	CONTROL SHIFT 6	■	223	SHIFT J	⬯
199	CONTROL SHIFT 7	■	224	SHIFT G	⊞
200	CONTROL SHIFT Q	❒	225	SHIFT H	⊞
201	CONTROL SHIFT W	❒	226	SHIFT B	⊞
202	CONTROL SHIFT E	❒	227	SHIFT N	⊞
203	CONTROL SHIFT R	❒	228	SHIFT Q	♠
204	CONTROL SHIFT T	❒	229	SHIFT W	♥
205	CONTROL SHIFT Y	❒	230	SHIFT E	♦
206	CONTROL SHIFT U	■	231	SHIFT R	♣
208	CONTROL SHIFT G	▤	232	SHIFT D	◲
209	CONTROL SHIFT H	▣	233	SHIFT F	◱
210	CONTROL SHIFT B	⊞	234	SHIFT C	◳
211	CONTROL SHIFT N	⊞	235	SHIFT V	◰
212	CONTROL SHIFT A	◢	236	SHIFT A	◲
213	CONTROL SHIFT S	◣	237	SHIFT S	◱
214	CONTROL SHIFT Z	◥	238	SHIFT Z	◳
215	CONTROL SHIFT X	◤	239	SHIFT X	◰
216	CONTROL SHIFT D	❒	240	SHIFT U	⊠
217	CONTROL SHIFT F	▭	241	SHIFT Y	⊞
			242	SHIFT M	⊞



Se quiser usar caracteres gráficos com certa freqüência em um programa, deve armazenar o código de controle em uma variável alfanumérica, como no exemplo abaixo. Neste caso, os quatro símbolos dos naipes do baralho são armazenados em **N$** e seus nomes, em **E$**:

```
5 HOME
10 FOR I=1 TO 4
20 READ E$(I)
30 LET N$(I)=CHR$(242)+
CHR$(227+I)
40 PRINT N$(I),E$(I)
50 NEXT I
60 DATA ESPADAS,COPAS,OUROS,
PAUS
```

Assim, toda vez que precisarmos imprimir na tela um dos quatro símbolos, digitaremos **PRINT N$(N)**, onde **N** é o código do naipe (1 = espadas, 2 = copas, 3 = ouros e 4 = paus).

Do mesmo modo quando quisermos utilizar uma cadeia de caracteres gráficos em vários pontos de um programa, devemos armazená-lo em uma variável alfanumérica. Como o TK-2000 não tem a função **STRING$**, que permite fazer isso com um único comando, é preciso escrever um laço de acumulação:

```
10 HOME
20 INPUT "CODIGO GRAFICO
(193-242) : ";      C
30 INPUT "COMPRIMENTO (1-39) :
";L
40 PRINT:PRINT:PRINT
50 LET S$=""
60 FOR I=1 TO L
70 LET S$=S$+CHR$(242)+CHR$(C)
80 NEXT I
90 PRINT S$
100 GOTO 10
```

A cadeia alfanumérica **S$** tem **L** vezes dois códigos, pois é necessário entrar um código 242 antes de cada código gráfico C (não funciona colocar apenas um código 242 como primeiro caractere de **S$**). Se rodar esse programa com vários códigos ao acaso — ou entrando-os em seqüência para ver o resultado —, constatará que às vezes surge uma interferência na tela inteira. Se isso ocorrer, experimente interromper o programa, usando **<CONTROL><C>** e digitando **RUN** novamente.

MAIS TÉCNICAS DE ORDENAÇÃO

Veja como pequenos melhoramentos nas rotinas de ordenação mais comuns podem aumentar sua velocidade de operação. E conheça uma rotina que bate todas as outras.

Todas as técnicas usadas para colocar dados em ordem apontam falhas, seja por dispenderem muito tempo, seja por exigirem um grande espaço de memória. Dessa forma, em cada aplicação específica é preciso procurar o método de ordenação mais eficiente, pois não existe uma técnica ideal que responda a todas as necessidades.

Já estudamos alguns dos métodos mais populares de ordenação de dados; a seguir, mostraremos outros, com a ressalva de que alguns deles são apenas refinamentos dos que foram apresentados anteriormente. Como você verá, estes são bem mais eficientes do que as versões originais.

SUBSTITUIÇÃO RETARDADA

O principal defeito da ordenação tipo bolha — a lentidão — torna-se particularmente evidente quando é preciso organizar uma grande quantidade de itens. No entanto, com uma pequena modificação no algoritmo original, pode-se reduzir o tempo gasto praticamente à metade.

A ordenação tipo bolha usa boa parte do tempo em comparar os diversos números da linha, trocando-os de posição até encontrar um valor maior. Nesse processo, várias trocas são feitas com grande dispêndio de tempo.

A rotina de substituição retardada funciona de modo semelhante, mas difere no fato de que nenhuma troca é realizada até que toda a linha tenha sido comparada. Vejamos um exemplo que emprega uma seqüência similar àquela estudada no artigo *Ordenação pelo Método de Bolhas* (página 292):

```
início                              fim

67  35  72  19  47  38  11  86
```

O primeiro valor, 67, é comparado (mas não trocado) com 35, como aconteceria na rotina de bolha normal; o mesmo acontece em relação ao primeiro número maior que ele, 72. Este último é tomado como o novo maior número e comparado sucessivamente com 19, 47, 38 e 11, antes de encontrar o valor máximo da seqüência: 86. Este permanece na última posição. O quadro que se segue mostra como tudo acontece, passo a passo. Alguns números aparecem entre colchetes. O primeiro deles é 86, o maior valor da primeira seqüência. Nas próximas linhas, os colchetes assinalam o maior número incorretamente localizado e o valor que ocupa seu lugar. Esses números são trocados na linha seguinte. Assim, na segunda linha, 72 é identificado como o maior incorretamente posicionado, e trocado com o 11, e assim por diante.

```
início                              fim

 67  35  72  19  47  38  11[86]
 67  35 [72] 19  47  38 [11] 86
[67] 35  11  19  47 [38] 72  86
 38  35  11  19 [47] 67  72  86
[38] 35  11 [19] 47  67  72  86
 19 [35][11] 38  47  67  72  86
[19][11] 35  38  47  67  72  86
 11  19  35  38  47  67  72  86
```

O número de comparações feito é o mesmo da rotina tipo bolha, mas o de trocas é muito menor.

Uma parte do programa a seguir é igual a um dos programas que aparecem no artigo *Rotinas de Ordenação*. Para executá-lo, basta adicionar as linhas de 4000 em diante (cuidado para não confundir a letra I com o número 1, ao fazer a digitação). Os usuários do Apple e do MSX devem fazer as modificações no programa principal conforme o indicado para todas as máquinas.

```
 10 POKE 23658,8: LET T=0:
 INPUT "NUMERO DE ITENS ";AA:
 IF AA<2 THEN  GOTO 10
 15 DIM A(AA)
 20 PRINT : PRINT "TABELA DESO
 RDENADA": PRINT
 30 FOR Z=1 TO AA
 40 LET A(Z)=INT (RND*100)+1
 50 PRINT TAB T;A(Z);: LET T=T
 +4: IF T>30 THEN  LET T=0
 60 NEXT Z
 70 PRINT : PRINT : PRINT "PRE
 SSIONE 'O' PARA ORDENAR"
 80 LET K$=INKEY$: IF K$<>"O"
 THEN  GOTO 80
 90 GOSUB 4000
100 PRINT : PRINT "TABELA ORDE
NADA": PRINT
110 LET T=0: FOR Z=1 TO AA
```

```
 120 PRINT TAB T;A(Z);: LET T=T
 +4: IF T>30 THEN  LET T=0
 130 NEXT Z
 140 GOTO 10
3999  REM ORDENACAO POR SUBSTIT
UICAO RETARDADA
4000 FOR I=1 TO AA-1
4010 LET K=I
4020 FOR J=I+1 TO AA
4030 IF A(J)<A(K) THEN  LET K=J
4040 NEXT J
4050 IF I<>K THEN  LET T=A(K):
LET A(K)=A(I): LET A(I)=T
4060 NEXT I
4070 RETURN
```

```
10 PRINT:PRINT:INPUT "NUMERO DE
 ITENS";AA:IF AA<2 THEN 10
15 DIM A(AA)
20 PRINT:PRINT"TABELA DESORDENA
DA":PRINT
30 FOR Z=1 TO AA
40 A(Z)=RND(100)
50 PRINT A(Z);
60 NEXT Z
70 PRINT:PRINT:PRINT "PRESSIONE
 'O' PARA ORDENAR"
80 K$=INKEY$:IF K$<>"O" THEN 80
90 GOSUB 4000
100 PRINT:PRINT:PRINT "TABELA O
RDENADA"
110 PRINT:FOR Z=1 TO AA
```

- ORDENAÇÃO POR SUBSTITUIÇÃO RETARDADA
- ORDENAÇÃO POR ESPALHAMENTO: FAZ O QUE SE FARIA MANUALMENTE
- RÁPIDA E SIMPLES: A ORDENAÇÃO DO JOGADOR DE CARTAS OU POR INSERÇÃO
- ORDENAÇÃO INSTANTÂNEA: A MAIS RÁPIDA DE TODAS


```
120 PRINT A(Z);
130 NEXT Z
140 RUN
3999 ORDENACAO POR SUBSTITUICAO
 RETARDADA
4000 FOR I=1 TO AA-1
4010 K=I
4020 FOR J=I+1 TO AA
4030 IF A(J)<A(K) THEN K=J
4040 NEXT J
4050 IF I<>K THEN T=A(K):A(K)=A
(I):A(I)=T
4060 NEXT I
4070 RETURN
```

Para que o programa acima rode no MSX substitua as seguintes linhas:

```
5 R=RND(-TIME)
40 A(Z)=INT(RND(1)*100+1)
50 PRINTA(Z),
120 PRINTA(Z),
4050 IF I<>K THEN SWAP A(K),A(I
)
```

Para que o programa acima rode nos micros da linha Apple II e TK-2000, substitua as linhas a seguir:

```
40 A(Z)=INT(RND(1)*100+1)
50 PRINTA(Z),
80 GET K$:IF K$<>"0" THEN 80
120 PRINTA(Z),
```

ORDENAÇÃO POR ESPALHAMENTO

A ordenação por espalhamento é outra rotina cuja velocidade se aproxima à da ordenação tipo bolha quando ambas são usadas para listas parcialmente ordenadas. Neste caso, uma matriz secundária é criada para uma ordenação preliminar parcial. Os valores inicial e final são determinados no início, enquanto outros itens só podem ser adicionados à lista quando esta estiver totalmente ordenada.

Embora essa rotina seja relativamente rápida, o uso de uma matriz secundária para armazenar dados diminui a memória disponível do micro.

Note que se deve especificar o valor máximo, o que é feito por meio da linha 5010. No nosso caso, esse valor é 100, que é o maior número aleatório permitido na linha 40.

Adicione as linhas seguintes ao primeiro programa:

```
  90 GOSUB 5000
4999 REM ORDENACAO POR ESPALHAM
ENTO
5000 DIM B(1.2*AA+30)
5010 FOR J=1 TO AA: LET K=INT (
A(J)*AA/100)+1
5020 IF B(K)=0 THEN  LET B(K)=A
(J): NEXT J: GOTO 5040
5030 LET K=K+1: GOTO 5020
5040 LET J=1: FOR K=1 TO 1.2*AA
+30: IF B(K)=0 THEN  NEXT K: GO
TO 5060
5050 LET A(J)=B(K): LET J=J+1:
NEXT K
5060 FOR J=AA-1 TO 1 STEP -1: L
ET F=-1
5070 FOR K=1 TO J
5080 IF A(K)>A(K+1) THEN  LET F
=0: LET T=A(K): LET A(K)=A(K+1)
: LET A(K+1)=T
5090 NEXT K: IF F=0 THEN  NEXT
J: RETURN
```

```
90 GOSUB 5000
```

```
4999 REM **ORDENACAO POR ESPALH
AMENTO**
5000 DIM B(1.2*AA+30)
5010 FOR J=1 TO AA:K=INT(A(J)*A
A/100)+1
5020 IF B(K)=0 THEN B(K)=A(J):N
EXT:GOTO 5040
5030 K=K+1:GOTO 5020
5040 J=1:FOR K=1 TO 1.2*AA+30:I
F B(K)=0 THEN NEXT:GOTO 5060
5050 A(J)=B(K):J=J+1:NEXT
5060 FOR J=AA-2 TO 1 STEP -1:F=
-1
5070 FOR K=1 TO J+1
5080 IF A(K)>A(K+1) THEN F=0:T=
A(K):A(K)=A(K+1):A(K+1)=T
5090 NEXT:IF F=0 THEN NEXT
5100 RETURN
```

Os usuários do MSX podem trocar a segunda metade da linha 5080, que faz a troca de valores entre as variáveis **A(K)** e **A(K + 1)** para **SWAP A(K), A(K + 1)**. O valor 1.2 na linha 5000 pode ser ajustado para fornecer o espaço necessário à matriz. Essa rotina de ordenação imita, de certa forma, a maneira normalmente usada para ordenar um grupo de informações: uma vez misturados os dados, e determinados o maior e o menor valor, tudo vai sendo posicionado conforme as prioridades.

ORDENAÇÃO POR INSERÇÃO

Uma opção para muitas aplicações é a rotina de ordenação por inserção. Supondo que os números abaixo sejam cartas de baralho, disponha-os na ordem dada, da esquerda para a direita:

```
esq.                    dir.
  9    4    5    7    2
```

O processo começa a partir da esquerda (ou pelo primeiro valor da lista a ser ordenada). Ele procura pela primeira ocorrência de um número menor fora de ordem. Um rápido exame de linha mostra o 4 fora de lugar. Ele é então reposicionado (inserido) antes do 9, formando a nova seqüência:

```
esq.                    dir.
  4    9    5    7    2
```

As cartas 4 e 9 estão agora na seqüência correta, mas, quando incluímos os outros números (5, 7 e 2) em nossa ob-

servação, a ordem desaparece. A cada passada, contudo, as cartas vão sendo reordenadas, até que alcancemos a seqüência correta:

```
esq.                    dir.

 4    9    5    7    2
 4    5    9    7    2
 4    5    7    9    2
 2    4    5    7    9
```

Como se pode ver, não há aqui separação de grupos ou comparações par a par de valores, como na maioria dos métodos já estudados: as cartas (números) são alinhadas da esquerda para a direita em ordem crescente, depois de algumas trocas de posição.

Esse processo é muito semelhante ao usado por um jogador de baralho ao analisar e ordenar uma mão de cartas. De fato, a ordenação por inserção é também conhecida como ordenação do jogador de cartas.

Vejamos agora um grupo de números um pouco maior, num estágio em que alguns valores já foram ordenados:

```
grupo não ordenado      g. ordenado
                              34
                              47
                              59
                              87
      102>>>>>>>>>
       26                    144
       73                    167
      193
```

O número 102 é colocado na sua posição correta e a rotina prossegue deslocando os valores remanescentes:

```
grupo não ordenado      g. ordenado
       26>>>>>>>>>
       73                     34
      193                     47
                              59
                              87
                             102
                             144
                             167
```

Em seguida, o número 73 é inserido na sua posição; resta agora, como se pode observar, apenas um valor não ordenado:

```
grupo não ordenado      g. ordenado
                              26
                              34
                              47
                              59
       73>>>>>>>>>
      193                     87
                             102
                             144
                             167
```

E, para completar:

```
                              26
                              34
                              47
                              59
                              73
                              87
                             102
                             144
                             167
      193>>>>>>>>>
```

Adicione as linhas abaixo ao programa de demonstração para observar a velocidade dessa rotina de ordenação:

```
 90 GOSUB 6000
5999 REM ORDENACAO POR INSERCAO
6000 FOR I=1 TO AA-1
6010 LET K=A(I+1)
6020 FOR J=I TO 1 STEP -1
6030 IF K>=A(J) THEN  GOTO 6070
6040 LET A(J+1)=A(J)
6050 NEXT J
6060 LET J=0
6070 LET A(J+1)=K
6080 NEXT I: RETURN
```

```
90 GOSUB 6000
5999 REM **ORDENACAO POR INSERCAO**
6000 FOR I=1 TO AA-1
6010 K=A(I+1)
6020 FOR J=I TO 1 STEP -1
6030 IF K>=A(J) THEN 6070
6040 A(J+1)=A(J)
6050 NEXT J
6060 J=0
6070 A(J+1)=K
6080 NEXT I:RETURN
```

O valor da variável **K** é o próximo número da lista não ordenada a ser comparado. O laço externo que começa na linha 6000 "varre" a lista ordenada, passando dos menores para os maiores valores, até encontrar a posição para **K**. Então, a rotina das linhas 6020 a 6050 expande a lista ordenada para que haja lugar para o novo item. O programa continua até que todos os valores da lista não ordenada tenham sido colocados nos seus lugares.

ORDENAÇÃO INSTANTÂNEA

Embora seja considerada bastante rápida, a rotina de ordenação por inserção parece lenta quando comparada com a rotina de ordenação instantânea (ou *quicksort*, se você preferir). Esta última, contudo, além de ser complexa, exige muito espaço de memória. Assim, ela não é encontrada com grande freqüência em programas especialmente desenhados para as limitadas memórias dos micros domésticos.

Seja como for, a ordenação instantânea é, sem dúvida, a mais veloz de todas as rotinas que já examinamos até agora. Além disso, ela vai mais longe do que uma simples rotina de comparação e troca. E, embora seu algoritmo seja bastante complexo (a ponto de não podermos explicá-lo aqui), vale a pena conhecê-la e usá-la!

```
 90 GOSUB 7000
6999 REM ORDENACAO INSTANTANEA
7000 LET K=0: LET I=0: DIM S(AA
)
7010 LET S(I+1)=1: LET S(I+2)=A
A
7020 LET K=K+1
7030 IF K=0 THEN  RETURN
7040 LET K=K-1: LET I=K+K
7050 LET A=S(I+1): LET B=S(I+2)
7060 LET Z=A(A): LET U=A: LET L
=B+1
7070 LET L=L-1
7080 IF L=U THEN  GOTO 7150
7090 IF Z<=A(L) THEN  GOTO 7070
7100 LET A(U)=A(L)
7110 LET U=U+1
7120 IF L=U THEN  GOTO 7150
7130 IF Z>=A(U) THEN  GOTO 7110
7140 LET A(L)=A(U): GOTO 7070
7150 LET A(U)=Z
7160 IF B-U>=2 THEN  LET I=K+K:
 LET S(I+1)=U+1: LET S(I+2)=B:
LET K=K+1
7170 IF L-A>=2 THEN  LET I=K+K:
 LET S(I+1)=A: LET S(I+2)=L-1:
LET K=K+1
7180 GOTO 7030
```

```
90 GOSUB 7000
6999 REM **ORDENACAO INSTANTANEA**
7000 K=0:I=0:DIM S(AA)
7010 S(I+1)=1:S(I+2)=AA
7020 K=K+1
7030 IF K=0 THEN RETURN
7040 K=K-1:I=K+K
7050 A=S(I+1):B=S(I+2)
7060 Z=A(A):U=A:L=B+1
7070 L=L-1
7080 IF L=U THEN 7150
7090 IF Z<=A(L) THEN 7070
7100 A(U)=A(L)
7110 U=U+1
7120 IF L=U THEN 7150
7130 IF Z>=A(U) THEN 7110
7140 A(L)=A(U):GOTO 7070
7150 A(U)=Z
7160 IF B-U>=2 THEN I=K+K:S(I+1
)=U+1:S(I+2)=B:K=K+1
7170 IF L-A>=2 THEN I=K+K:S(I+1
)=A:S(I+2)=L-1:K=K+1
7180 GOTO 7030
```

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 DE JOGOS
Programar o jogo do Otelo é relativamente fácil. Difícil é preparar uma estratégia para derrotar o computador.

PROGRAMAÇÃO BASIC
Aprenda a converter partituras musicais em linhas de programas BASIC e ouça as suas canções preferidas.

CÓDIGO DE MÁQUINA
Comece agora a programar um videogame completo: participe do jogo *Avalanche* e aprenda mais sobre código de máquina.