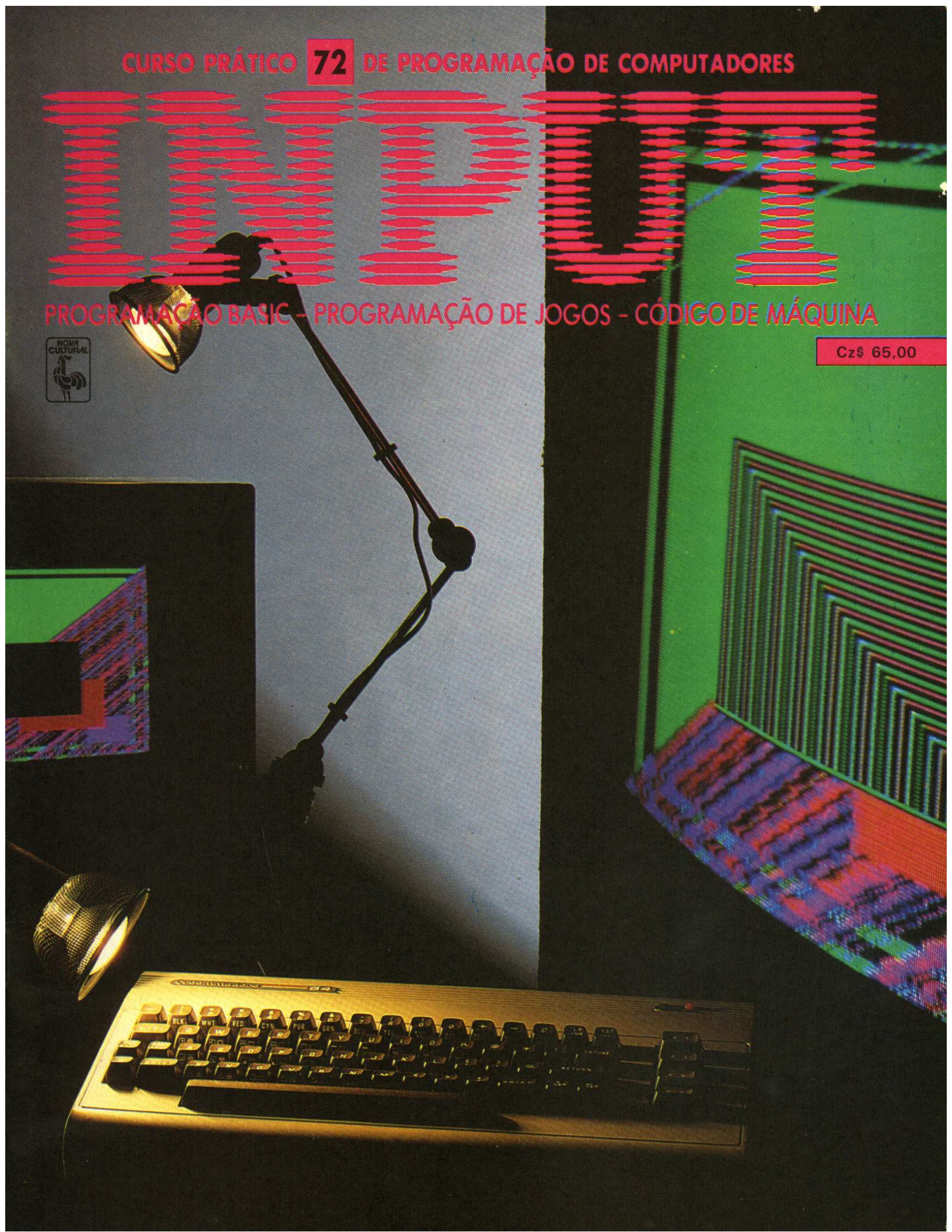
CURSO PRÁTICO 72 DE PROGRAMAÇÃO DE COMPUTADORES

# BASIC

PROGRAMAÇÃO BASIC – PROGRAMAÇÃO DE JOGOS – CÓDIGO DE MÁQUINA

NOVA CULTURAL

Cz$ 65,00

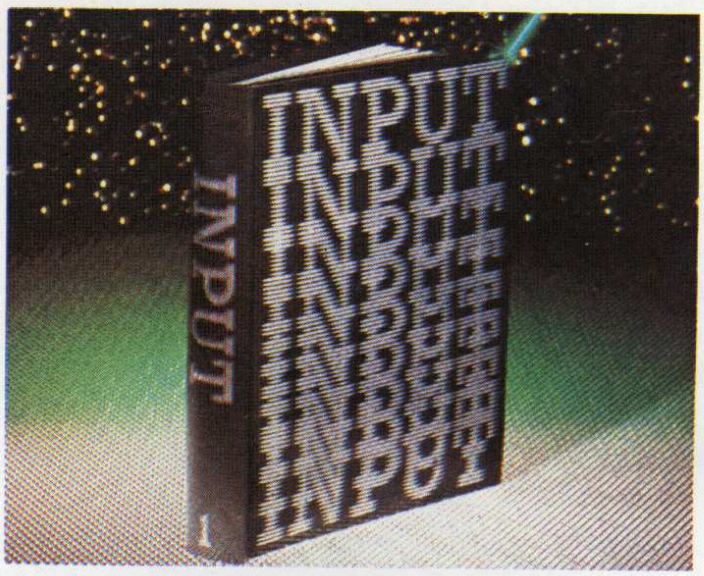


# INPUT

Vol. 5 N.º 72

## 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. CEP 01051.


EDITOR
RICHARD CIVITA

**NOVA CULTURAL**

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

**REDAÇÃO**
**Diretor Editorial:** Carmo Chagas
**Editores Executivos:**
Antonio José Filho, Berta Sztark Amar
**Editor Chefe:** Paulo de Almeida
**Editoras Assistentes:** Ana Lúcia B. de Lucena,
Marisa Soares de Andrade
**Chefe de Arte:** Carlos Luiz Batista

**Assistentes de Arte:** Dagmar Bastos Sampaio,
Grace Alonso Arruda, Monica Lenardon Corradi

**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, 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:** Flávio Maculan
**Gerente de Circulação:** Denise Mozol
**Gerente de Propaganda e Publicidade:** José Carlos Madio
**Gerente de Pesquisa e Análise de Mercado:**
Wagner M. P. Nabuco de Araújo

CLC

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




Edição organizada pela Editora Nova Cultural Ltda.
Av. Brigadeiro Faria Lima, n.º 2000 - 3.º 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.

# UM EDITOR MUSICAL (3)

- INSTRUÇÕES
- MENU
- EXECUÇÃO DA MELODIA
- EDIÇÃO
- GRAVAÇÃO E LEITURA

**Complete a listagem do seu editor musical e entregue-se ao prazer de compor e executar melodias. As instruções dadas a seguir mostram como utilizar cada função do programa.**

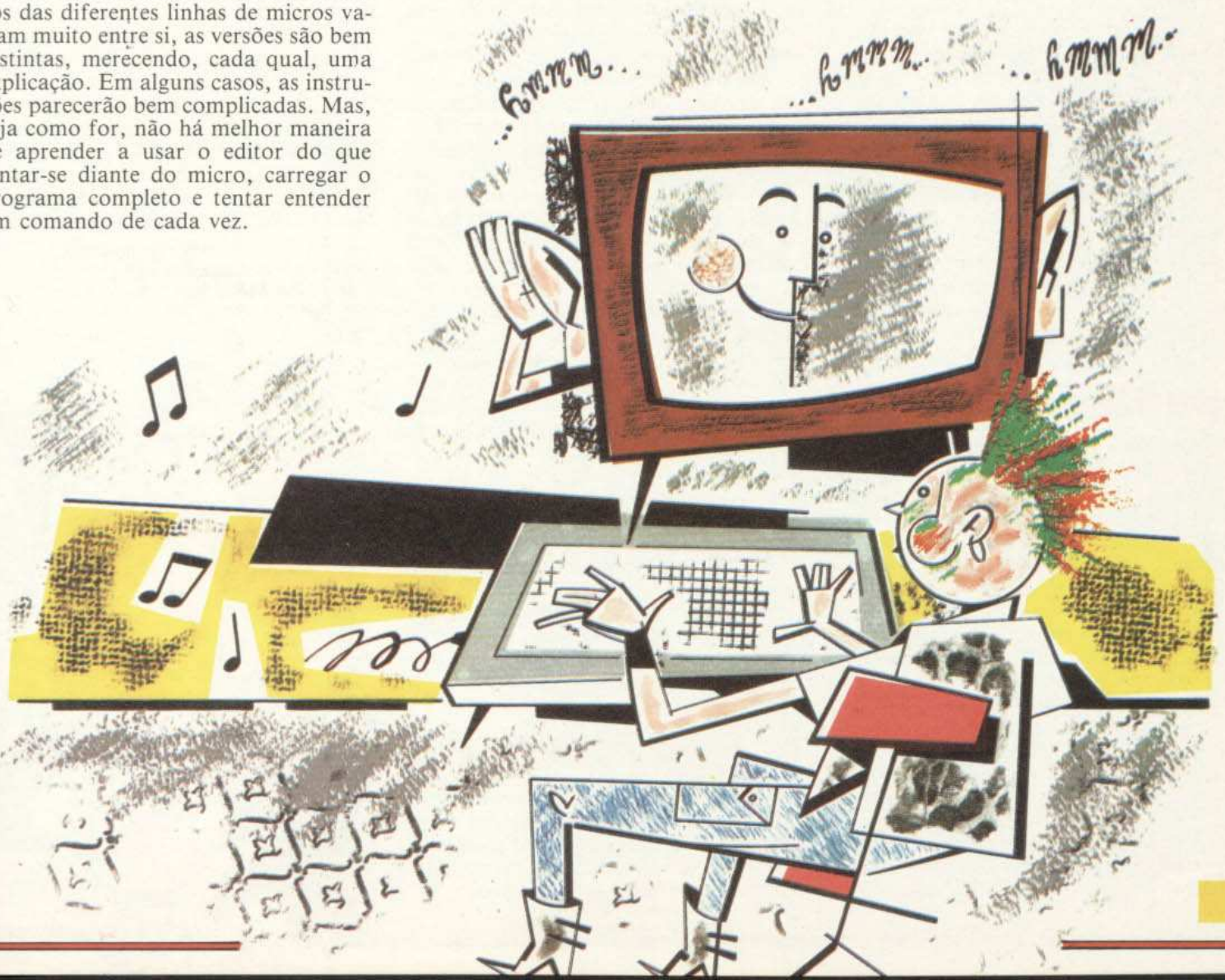
Apresentamos aqui a última parte do editor musical. Uma vez adicionada ao restante do programa, você poderá se dedicar à composição e execução de peças musicais.

O programa foi feito de modo a obter o máximo de cada máquina, em termos de efeitos sonoros. Como os recursos das diferentes linhas de micros variam muito entre si, as versões são bem distintas, merecendo, cada qual, uma explicação. Em alguns casos, as instruções parecerão bem complicadas. Mas, seja como for, não há melhor maneira de aprender a usar o editor do que sentar-se diante do micro, carregar o programa completo e tentar entender um comando de cada vez.

O menu do programa do computador Spectrum oferece sete opções. Selecione a opção 1, que transforma o teclado do micro num piano.

A disposição das notas é igual à que vimos no artigo da página 721. O dó mais grave corresponde à tecla Q; o dó médio, à tecla I, e o dó mais agudo, à tecla B. O usuário deve determinar se irá acrescentar notas à melodia anterior, ou iniciar uma nova. Como nada foi composto ainda, escolha 'S' para começar. Defina também a duração de cada nota, usando <**SYMBOL**> <**SHIFT**> seguido de um número de 1 a 5 (semicolcheia, colcheia, semínima, mínima, semibreve, respectivamente — quanto maior o número, mais demorada é a nota). Pode-se estabelecer a duração de cada nota antes de digitá-la e, também, corrigi-la após digitar a melodia completa, selecionando a opção 4.

O programa emite os sons à medida que as teclas são pressionadas, armazenando as notas musicais correspondentes na memória. Depois, pode-se executar a melodia completa sempre que se desejar através da opção 3. Para modificar o andamento da melodia, usa-se um número entre 1 e 15 — quanto maior



o número, mais rápido o andamento. A opção 2 possibilita a digitação das notas por meio de um código simples. As doze notas possíveis — C (dó), C # (dó sustenido), D (ré), até o B (si) — são numeradas de 0 a 11, e a pausa recebe o código −4. A oitava da melodia é especificada por um valor compreendido entre 1 e 7, e o número que indica a duração de cada nota corresponde à sua duração teórica. Assim, uma semicolcheia vale 1; uma colcheia, 2; uma semínima, 4; uma mínima, 8 e uma semibreve, 16. Essas durações valem, igualmente, para a pausa.

O código é muito fácil de usar. Por exemplo, a nota dó, segunda oitava, semicolcheia, tem o código 1024; a nota si, quarta oitava, semibreve, é 11416. Para pausas, −404 é o intervalo de uma semínima. Não se esqueça de que cada código tem dois dígitos; assim, uma colcheia vale 02, e não 2.

Após digitar algumas notas, pressione <**RETURN**> para voltar ao menu; selecione a opção 3 para tocar a melodia introduzida até o momento.

Se você quiser alterar, apagar ou acrescentar algumas notas, selecione a opção 4 do menu. Inicialmente, pressione **D**, para selecionar as notas que serão modificadas. Depois, entre o número das notas e tecle **E** para editá-las. Para mudar uma nota, digite seu número, <**RETURN**> e seu código. Para inserir uma nota, aperte **X** e entre seu número. Caso queira apagar tudo e reiniciar o trabalho, selecione a opção 5.

Quando estiver satisfeito com a melodia, grave-a através da opção 6.

A opção 7 permite recuperar uma música da fita cassete.

## T

Ao executar o programa, um menu com nove opções será exibido na tela. Comece com a opção 3, para tocar música diretamente no teclado.

As notas estão dispostas em duas fileiras, como vimos no artigo da página 721. O dó menor corresponde à tecla Q; o dó médio, à I, e o dó maior, à V. A barra de espaço indica uma pausa com a duração desejada. O computador registrará cada nota que você tocar, mas não em tempo real — assim, não importa que você demore mais ou menos para teclar uma nota. Também não haverá problema se algum erro for cometido: é fácil corrigir as notas depois.

Se você quiser mudar uma oitava antes de executar uma determinada nota, recorra às teclas com setas pará cima e para baixo. Para alterar a duração da nota, pressione as teclas com setas para a esquerda e a direita. A oitava e a duração correntes estarão impressas no topo da tela. Enquanto você toca, as notas vão sendo exibidas com a duração e oitava respectivas.

Depois de experimentar essa maneira de entrar com as notas, selecione a opção 4 (as notas que você tocou permanecem na memória). Ela permite a entrada das notas por um processo que em certos casos pode ser bem útil. Primeiro, digite o nome da nota, de "**a**" até "**f**", para as notas naturais; **A, C, D, F, G**, para acidentes (bemóis são colocados como sustenidos; logo, Bb= A # etc.), e "**p**", para uma pausa. Em seguida, entre a oitava — um número de 1 a 6 — e a duração da nota — letras **w, h, q, e, s,** correspondentes a semibreve, mínima, semínima, colcheia e semicolcheia. Para uma nota pontuada, simplesmente adicione um ponto no final. Por exemplo: A2e. significa nota A (lá sustenido), oitava 2, colcheia pontuada; c3w significa nota C (dó), oitava 3, semibreve.

Entre as notas na ordem que quiser e ouça a música quando desejar, selecionando a opção 7 do menu principal.

Outras opções do menu são a mudança no tempo de execução e a mudança geral de oitava (com efeito retroativo) — para executá-las, basta que teclemos um novo valor. O tempo de execução varia de 0 a 255 e indica a velocidade com que as notas são tocadas. Utilizando as opções Superior e Inferior, você pode subir ou descer a oitava de qualquer grupo de notas.

Para editar a melodia, selecione a opção 6. Com ela, você lista as notas e identifica as que quer alterar. É possível escolher entre **apagar, inserir, alterar** ou **continuar**. Experimente cada uma dessas opções. Para apagar notas, entre o número inicial e o número de notas que deseja eliminar. Para inserir, digite o número da nota que está antes do ponto onde se iniciará a inserção e entre uma nota por vez. Para alterar uma nota, tecle seu número e entre o novo conteúdo.

Finalmente, quando estiver satisfeito com a melodia, armazene-a em fita com a opção 2 e recupere-a quando quiser com a opção 1.

## MSX

Ao iniciar a execução do programa, um menu com nove opções será exibido na tela. Comece com a opção 3 para tocar música diretamente no teclado. Teremos três oitavas disponíveis.

As notas estão dispostas no teclado em duas fileiras: **Q** corresponde ao dó da oitava mais baixa; **I**, ao dó da oitava média, e **V**, ao dó da oitava mais alta. A nota A # (A ou lá sustenido) não pode ser obtida com a tecla "^", já que esta não gera nenhum código ASCII que identifique o seu pressionamento. Assim, A # foi atribuída à tecla "[", deixando uma pequena descontinuidade no teclado. A barra de espaço indica uma pausa com a duração especificada.

O computador registrará a nota que você tocar, mas não imediatamente, devido ao tempo necessário para o processamento das instruções. Portanto, não tem nenhuma importância que você demore mais ou menos para teclar uma nota. Também não haverá problema se algum erro for cometido: será fácil corrigir as notas mais tarde.

Caso você queira mudar a oitava antes de executar uma nota, use as teclas com setas para cima e para baixo. Para alterar a duração da nota, pressione as teclas com setas para a esquerda e a direita. A oitava mais baixa disponível no teclado estará impressa no topo da tela, bem como a duração das notas, que vão sendo exibidas à medida que você as toca. Para obter uma duração pontuada, selecione as teclas <**INS**> e <**DEL**>; para apagar a última nota executada, a tecla <**CLS**>.

Tendo experimentado essa forma de composição, use a tecla <**ENTER**> e volte ao menu principal. Selecione a opção 4 (as notas que você tocou permanecem na memória). Ela permite a entrada das notas por um processo que, em certos casos, pode ser muito útil. Primeiro, digite o nome da nota, de

"a" até "f", para as notas naturais; **A, C, D, F, G,** para acidentes (bemóis são colocados como sustenidos; logo, Bb = A # etc.), e "**p**", para uma pausa. Em seguida, entre a oitava — um número de 1 a 8 — e a duração da nota — letras **w, h, q, e, s, t, u**, correspondentes à semibreve, mínima, semínima, colcheia, semicolcheia, fusa e semifusa. Para uma nota pontuada, adicione um ponto no final. Por exemplo: c8t. significa C (dó), oitava 8, fusa pontuada; A4u significa A (lá sustenido), oitava 4, semifusa.

Entre as notas na ordem que quiser e ouça a música quando desejar, selecionando a opção 7 do menu principal.

Outras opções que temos são a mudança no tempo de execução das notas e a mudança geral da oitava. Esta última tem efeito retroativo, alterando a oitava de notas na memória. Para ambas as opções, basta teclar o novo valor. O tempo de execução varia de 32 a 255 e indica a velocidade com que as notas são tocadas. Com as opções Superior e Inferior você pode subir ou descer a oitava de um grupo de notas.

Para editar a melodia, selecione a opção 6. Com ela, você lista as notas e verifica o que deseja alterar. É possível escolher entre **apagar, inserir, alterar** ou **continuar**. Experimente cada uma dessas opções. Para apagar algumas notas, entre o número da primeira nota e o número de notas que deseja eliminar. Para inserir, digite o número da nota imediatamente anterior ao ponto onde pretende iniciar a inserção e entre uma nota de cada. Para alterar uma nota, tecle seu número e entre o novo conteúdo.

Finalmente, quando estiver satisfeito com a melodia, armazene-a em fita com a opção 2 e recupere-a quando quiser com a opção 1.


```
4452 INPUT "Entre com numero da
 nota -  ";NN
4454 IF (NN<1) OR NN>ct THEN  G
OTO 4000
4460 PRINT : PRINT "Re-entrando
 nota ";NN
4470 PRINT : PRINT
4480 INPUT "Entre com nova nota
 -  ";N$
4490 IF N$="" THEN  GOTO 4300
4500 FOR i=1 TO LEN (N$): IF (N
$(i)<"0" OR N$(i)>"9") AND (N$(
i)<>"-") THEN  GOTO 4000
4510 LET N=VAL (N$)
4520 IF INT (N/1000)>11 THEN  G
OTO 4000
4530 IF N<0 THEN  GOTO 4590
4540 LET M=INT (N/100): LET D=N
-M*100
4550 LET O=M-INT (M/10)*10: IF
O<1 OR O>7 THEN  GOTO 4000
4560 LET M=INT (M/10)+(O-1)*12-
36
4570 LET t(2*NN-1)=D: LET t(2*N
N)=M
4580 GOTO 4000
4590 LET M=INT (N/100)+1: LET D
=0-(N-M*100)
4600 IF M<>-4 THEN  GOTO 4000
4610 GOTO 4570
4700 CLS
4705 PRINT "Entre com numero da
 nota ANTES"'"da nova nota a se
r inserida."'"(0 para sair)"
4730 INPUT is
4735 IF is=0 THEN  GOTO 4000
4740 IF is>ct+1 THEN  GOTO 4700
4745 CLS
4750 GOSUB 2500
4755 PRINT
4760 INPUT "Entre com nova Nota
 -  ";N$
4765 IF LEN (N$)=0 THEN  GOTO 4
700
4770 FOR i=1 TO LEN (N$): IF (N
$(i)<"0" OR N$(i)>"9") AND (N$(
i)<>"-") THEN  GOTO 4700
4772 LET N=VAL (N$)
4774 IF INT (N/1000)>11 THEN  G
OTO 4700
4776 IF N<0 THEN  GOTO 4792
4778 LET M=INT (N/100): LET D=N
-M*100
4782 LET O=M-INT (M/10)*10: IF
O<1 OR O>7 THEN  GOTO 4700
4784 LET M=INT (M/10)+(O-1)*12-
36
4786 FOR i=ct TO is STEP -1: LE
T t(2*(i+1))=t(2*i): LET t(2*(i
+1)-1)=t(2*i-1): NEXT i
4788 LET t(2*is-1)=D: LET t(2*i
s)=M
4790 LET ct=ct+1: GOTO 4000
4792 LET M=INT (N/100)+1: LET D
=0-(N-M*100)
4794 IF M<>-4 THEN  GOTO 4700
4796 GOTO 4786
4800 CLS
4805 PRINT "Entre com numero da
 nota a ser  apagada."'"(0 par
a sair"'
4830 INPUT de
4835 IF de=0 THEN  GOTO 4000
4840 IF de>ct THEN  GOTO 4800
4845 FOR i=de TO ct: LET t(2*i)
=t(2*(i+1)): LET t(2*i-1)=t(2*(
i+1)-1): NEXT i
4850 LET ct=ct-1
4855 GOTO 4000
4900 DATA 3,0,-11,-9,0,-6,-4,-2
```

## MICRO DICAS

### TRANSCRIÇÃO DE PARTITURAS

Nem todos conseguem executar música simplesmente sentando-se diante do micro e usando o editor musical como um processador de textos. Embora a opção de *playback* (execução da melodia entrada com o auxílio do editor) ajude bastante, nosso programa não permite representar na tela uma partitura com os detalhes de uma aplicação mais profissional.

Assim, a solução é recorrer a partituras de músicas de fácil execução para violão, flauta, em órgão eletrônico ou piano, e transcrevê-las usando o nosso editor musical.

```
,0,1
4910 DATA 12,9,8,-8,10,0,13,0,1
5,0
4920 DATA 0,16,14,2,4,-12,-7,6,
-5,-1
4930 DATA 11,-10,7,-3,5
5000 CLS
5010 INPUT "DIGITE NOME DO ARQU
IVO ";F$: LET T(maxnotes+1)=ct:
 SAVE F$ DATA T(): RETURN
6000 INPUT "DIGITE NOME DO ARQU
IVO ";F$: LOAD F$ DATA T(): LET
 ct=T(MAXNOTES+1): RETURN
```

```
1440 FOR I=ST TO NN
1450 PRINT#C,USING"###   ";I;
1460 A$=N$(I):B$=LEFT$(A$,1)
1470 IF B$>="a"AND B$<="g"THEN
C$=CHR$(ASC(B$)-32)+" "ELSE C$=
B$+"#"
1480 IF B$="p" THEN C$="- "
1490 PRINT#C,C$;
1500 IF B$<>"p"THEN PRINT#C," O
ITAVA#";MID$(A$,2,1);ELSE PRINT
#C,STRING$(11,32);
1510 PRINT#C,"  ";LE$(INSTR(R1$
,MID$(A$,3,1)));
1520 IF MID$(A$,4,1)="."THEN PR
INT#C,"." ELSE PRINT#C
1530 LP=LP+1:IF LP=13AND C=0 TH
ENLP=0:GOSUB940:CLS
1540 NEXT:GOSUB940
1550 PRINT@448,STRING$(63,32);:
PRINT@480,"PRESSIONE QUALQUER T
ECLA/MENU";:EXEC 36038:RETURN
1560 CLS
1570 PRINT@8,"MODO TOCAR NOTAS"
1580 IF NN=0 THEN RETURN
1590 PRINT:INPUT"INICIO NA NOTA
 (ENTER=1)";ST
1600 IF ST<=0 THEN ST=1 ELSE IF
 ST>NN THEN ST=NN
1610 PRINT"TEMPO=";TE
1620 PLAY"V31;T"+STR$(TE)
1630 EXEC 46481:FOR I=ST TO NN
1640 PRINT@256,"TOCANDO NOTA NU
MERO";I
1650 A$=N$(I)
1660 B$=LEFT$(A$,1):IF B$="p"TH
EN 1690 ELSE IF B$>="a" AND B$<
="g"THEN P$=CHR$(ASC(B$)-32)+"
" ELSE P$=B$+"#"
1670 PLAY"O"+MID$(A$,2,1)+"L"+L
2$(INSTR(R1$,MID$(A$,3,1)))+MID
$(A$,4,1)+P$
1680 GOTO 1700
1690 PLAY"P"+L2$(INSTR(R1$,MID$
(A$,3,1)))
1700 NEXTI
1710 RETURN
1720 CLS
1730 IF NN=0 THEN RETURN
1740 PRINT@5,"MUDANCA GERAL DE
OITAVA"
1750 PRINT@64,"OITAVA SUPERIOR
OU INFERIOR(S/I)"
1760 POKE 282,245:INPUT A$
1770 IF A$="" THEN RETURN
1780 IF A$<>"S" AND A$<>"I"THEN
1750
1790 PRINT"INICIO EM(ENTER=TUDO
)";:INPUT ST
1800 IF ST<=0THEN ST=1:EN=NN:GO
TO1840
1810 INPUT "FINAL  EM(ENTER=FIM
 )";EN
1820 IF EN=0 OR EN>NN THEN EN=N
N
1830 IF STT>EN THEN ST=EN
1840 FOR I=1TO NN
1850 B$=MID$(N$(I),2,1)
1860 IF A$="I"THEN C$=CHR$(ASC(
B$)-1):IF C$="0"THEN C$="5"
1870 IF A$="S"THEN C$=CHR$(ASC(
B$)+1):IF C$="6"THEN C$="1"
1880 MID$(N$(I),2,1)=C$
1890 NEXT:RETURN
1900 CLS
1910 PRINT@4,"CARREGAR MUSICA D
A FITA"
1920 PRINT:PRINT"ESTA OPCAO IRA
 APAGAR            QUALQUER MUSI
CA NA MEMORIA       -VOCE QUER C
ONTINUAR (S/N)";
1930 POKE 282,255:INPUT A$
1940 IF A$<>"S" THEN RETURN
1950 PRINT:LINE INPUT"NOMEARQ:"
;A$
1960 OPEN "I",#-1,A$
1970 INPUT#-1,T$,N$
1980 NN=VAL(N$):TE=VAL(T$)
1990 FOR I=1 TO NN
2000 INPUT#-1,N$(I)
2010 NEXT:CLOSE#-1:RETURN
2020 CLS
2030 IF NN=0  THEN RETURN
2040 PRINT@5,"SALVAR MUSICA EM
FITA"
2050 PRINT:LINE INPUT"NOMEARQ:"
;A$
2060 OPEN "O",#-1,A$
2070 PRINT#-1,STR$(NN),STR$(TE)
2080 FOR I=1TO NN
2090 PRINT#-1,N$(I)
2100 NEXT:CLOSE#-1:RETURN
```

```
1610 FOR I=SL TO EL
1620 A$=N$(I):B$=LEFT$(A$,1)
```

```
1950 INPUT A$:IF A$="" THEN RET
URN
1960 IF A$<>"S" AND A$<>"I"THEN
 1940
1970 LOCATE 4,9:INPUT"INICIO EM
 (ENTER=  1 )";ST
1980 IF ST<=0 THEN ST=1:EN=NN
1990 LOCATE 4,10:INPUT"FINAL  E
M (ENTER=FIM )";EN
2000 IF EN=0 OR EN>NN THEN EN=N
N
2010 IF ST>EN THEN ST=EN
2020 FOR I=ST TO EN
2030 B$=MID$(N$(I),2,1)
2040 IF A$="I" THEN C$=CHR$(ASC
1630 IF B$>="a"AND B$<="g"THEN
C$=CHR$(ASC(B$)-32)+"   "ELSE C
$=B$+"♯  "
1640 IF B$="p" THEN C$="-   "
1650 C$=" "+C$
1660 IF B$<>"p"THEN C$=C$+"OITA
VA ♯"+MID$(A$,2,1)ELSE C$=C$+ST
RING$(9,32)
1670 C$=C$+" "+LE$(INSTR(R1$,MI
D$(A$,3,1)))
1680 IF MID$(A$,4,1)="."THEN C$
=C$+"."ELSE C$=C$+" "
1690 IF C=1 THEN LOCATE 5,LP+5:
PRINT USING"♯♯♯";I;:PRINT C$ELS
E LPRINT USING"♯♯♯";I;:LPRINT C
$
1700 IF RT<>0 THEN RETURN
1710 LP=LP+1:IF LP=13ANDC=1 THE
N LP=0 :GOSUB 1010:CLS
1720 NEXT:GOSUB 1010
1730 LOCATE 2,18:PRINT"APERTE Q
UALQUER TECLA-MENU PRINCIPAL"
1740 Q$=INKEY$:IF Q$="" THEN 17
40
1750 RETURN
1760 CLS:COLOR 1,10:LOCATE 8,1:
PRINT"MODO -TOCAR NOTAS"
1770 ST=1:IF NN=0 THEN RETURN
1780 LOCATE 4,5:INPUT"INICIO NA
 NOTA (ENTER=1)";ST
1790 IF ST<=0 THEN ST=1 ELSE IF
 ST>NN THEN ST=NN
1800 LOCATE 4,7:PRINT"TEMPO DE
EXECUÇÃO=";TE
1810 PLAY"V15T"+STR$(TE)
1820 FOR I=ST TO NN
1830 LOCATE 6,12:PRINT"TOCANDO
NOTA:";I
1840 A$=N$(I)
1850 B$=LEFT$(A$,1):IF B$="p"TH
EN 1880 ELSE IF B$>="a"ANDB$<="
g"THEN P$=CHR$(ASC(B$)-32)+" "E
LSE P$=B$+"♯"
1860 PLAY"O"+MID$(A$,2,1)+"L"+L
2$(INSTR(R1$,MID$(A$,3,1)))+P$+
MID$(A$,4,1)
1870 GOTO 1890
1880 PLAY"R"+L2$(INSTR(R1$,MID$
(A$,3,1)))+MID$(A$,4,1)
1890 NEXT I
1900 RETURN
1910 CLS:COLOR 1,3:ST=1:EN=NN
1920 IF NN=0 THEN RETURN
1930 LOCATE 7,1:PRINT"MUDANÇA G
ERAL DE OITAVA"
1940 LOCATE 0,6:PRINT"OITAVA SU
PERIOR OU INFERIOR(S/I)";
(B$)-1):IF C$="0" THEN C$="1"
2050 IF A$="S" THEN C$=CHR$(ASC
(B$)+1):IF C$="9" THEN C$="8"
2060 MID$(N$(I),2,1)=C$
2070 NEXT:RETURN
2080 CLS:COLOR 15,6
2090 LOCATE 5,1:PRINT"CARREGAR
MUSICA DO GRAVADOR";
2100 LOCATE12,3:PRINT"ATENÇÃO!!
"
2110 LOCATE 4,4:PRINT"  ESTA OP
ÇÃO IRÁ APAGAR                Q
UALQUER MUSICA NA MEMÓRIA"
2120 LOCATE 4,7:INPUT"VOCÊ QUER
 CONTINUAR?(S/N)";A$
2130 IF A$<>"S" THEN RETURN
2140 LOCATE 4,10:PRINT"aperte L
OAD no gravador"
2150 LOCATE 4,12:LINE INPUT"NOM
EARQ:";A$
2160 OPEN "CAS:A$" FOR INPUT AS
#1
2170 INPUT #1,NN,TE
2180 FOR I= 1 TO NN
2190 INPUT #1,N$(I)
2200 NEXT:CLOSE#1:RETURN
2210 CLS:COLOR 15,6
2220 LOCATE 8,1:PRINT"SALVAR MU
SICA EM FITA"
2230 LOCATE 4,6:PRINT"aperte SA
VE no gravador"
2240 LOCATE 4,8:LINE INPUT"NOME
ARQ:";A$
2250 OPEN "CAS:A$" FOR OUTPUT A
S#1
2260 PRINT #1,NN,TE
2270 FOR I= 1 TO NN
2280 PRINT #1,N$(I)
2290 NEXT:CLOSE #1:RETURN
```

# SPRITES EM LOGO PARA O MSX

**Algumas versões mais poderosas do LOGO permitem a programação de sprites. Seguindo as instruções deste artigo, os usuários do MSX poderão gerar e controlar quarenta figuras diferentes.**

Em artigos anteriores sobre a linguagem LOGO, vimos como desenhar na tela com a tartaruga, um cursor gráfico que pode ser ativado ou desativado por meio de comandos primitivos.

Como você deve ter notado, a elaboração de uma figura mais complexa demora um tempo razoavelmente longo, o que torna impossível, por exemplo, desenvolvermos um videogame com animação gráfica — tarefa simples em BASIC ou outra linguagem imperativa. Algumas versões do LOGO, porém, permitem a geração e utilização de *sprites*.

Nos artigos das páginas 188 e 808, vimos como trabalhar com sprites no MSX, única máquina nacional que possui esse recurso implementado em hardware. O Spectrum, o TRS-Color, o Apple e o TK-2000 têm comandos especiais para definir blocos gráficos (UDG) e usá-los em animações rápidas. Mas esses micros não dispõem de sprites "verdadeiros" — ou seja, não contam com controle de hardware (pelo *Video Display Processor*) que garanta a movimentação dos sprites na tela em diferentes planos, destacando-os ou ocultando-os conforme os critérios que determinam a prioridade de uns sobre os outros.

Aproveitando os recursos do MSX, o HotLOGO, versão lançada no Brasil pela Sharp para computadores dessa linha, inclui entre suas propriedades a programação de sprites verdadeiros.

Algumas versões de LOGO para o Apple e o TRS-Color também fazem uso de sprites. No exterior, a linha Commodore 64 tem poderosos instrumentos para a programação de sprites multicoloridos. Mas, em todas essas máquinas, os sprites não são controlados por hardware, como nos micros da linha MSX.

## SPRITES PRÉ-PROGRAMADOS

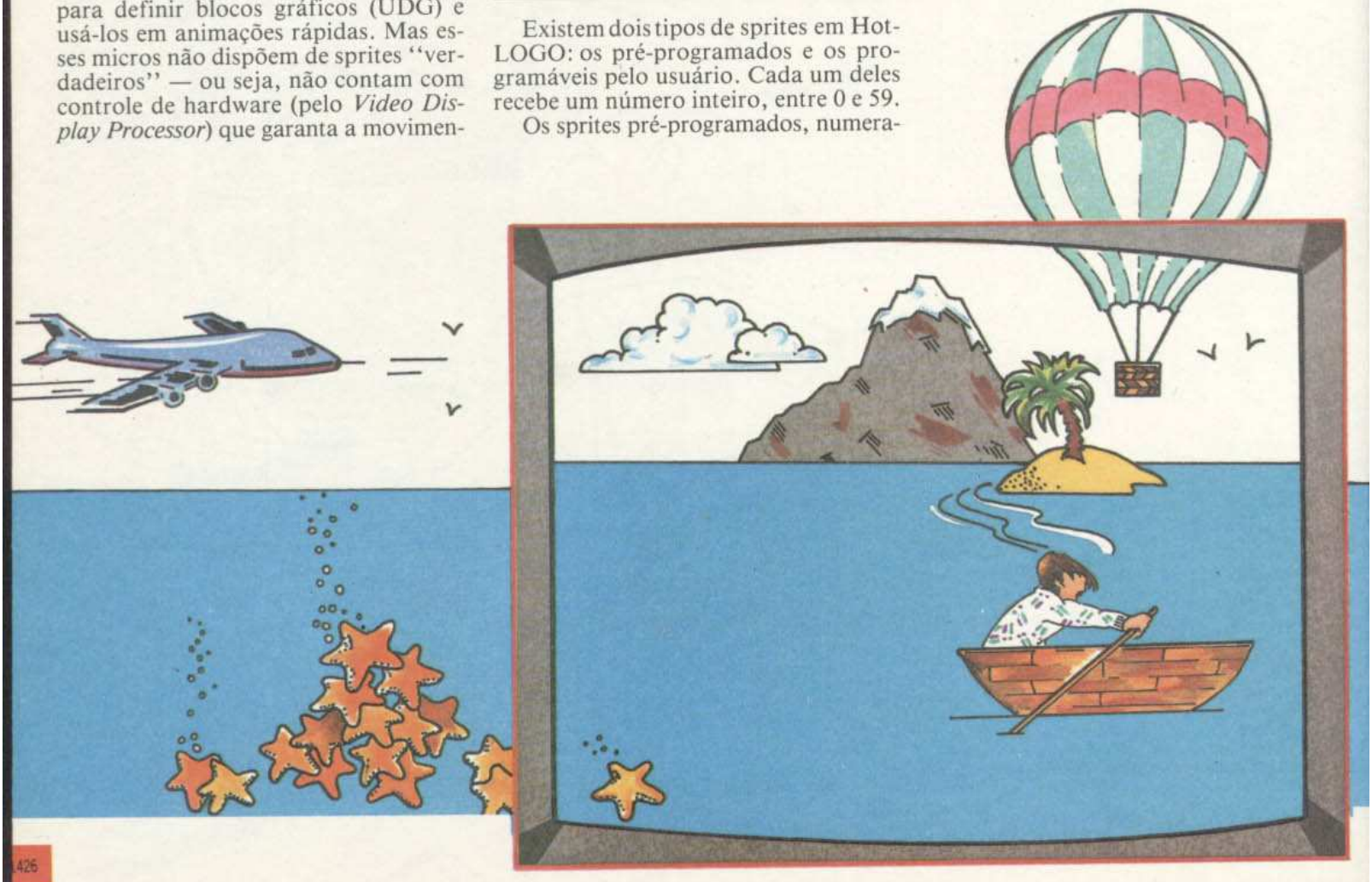
Existem dois tipos de sprites em HotLOGO: os pré-programados e os programáveis pelo usuário. Cada um deles recebe um número inteiro, entre 0 e 59.

Os sprites pré-programados, numerados de 0 a 9 e de 36 a 59, são matrizes de 16 x 16 pixels contendo figuras que o usuário não pode mudar. O acesso a eles é dado pelo comando **MUDEFIG**, seguido do número do sprite.

As figuras de 0 a 9 são:

| | |
|---|---|
| 0 círculo | 5 foguete |
| 1 coração | 6 tijolo |
| 2 gato | 7 helicóptero |
| 3 cachorro | 8 locomotiva |
| 4 caminhão | 9 vagão |

Os sprites numerados de 36 a 59 representam tartarugas em diversos ângulos e posições. Todos aceitam os mesmos comandos e podem ser movimentados simultaneamente na tela. Os spri-



tes fixos (0 a 9), porém, não mudam de orientação, como ocorre com as tartarugas ao encontrarem um comando **PARADIREITA** ou **PARAESQUERDA**.

## SPRITES PROGRAMÁVEIS

Os sprites numerados de 10 a 35 podem ser programados pelo usuário.

O primitivo utilizado para desenhar um sprite novo na tela é muito simples: chama-se **EDFIG** (edita figura), e deve indicar o número do sprite a ser criado ou modificado. Quando digitamos esse comando, a tela fica em branco e aparece um cursor piscando no canto superior esquerdo. Podemos movimentá-lo em qualquer direção, por meio das teclas de controle. Para acender um pixel (bit correspondente, na matriz do sprite), pressiona-se a tecla de espaço no ponto onde estiver o cursor. Para apagá-lo, pressiona-se a mesma tecla sobre esse ponto.

Se você quiser sobrepor uma grade à tela em branco, a fim de visualizar melhor o padrão do desenho, pressione as teclas <**CTRL**> e <**K**>.

Depois de criar ou modificar a figura a seu gosto, tecle <**ESC**>: ela será armazenada na memória, sob o número indicado no comando **EDFIG**. Se não quiser guardar a figura que está editando, tecle <**CTRL**> e <**STOP**>.

## ANIMAÇÃO GRÁFICA

Na produção de animações gráficas com sprites, usam-se os comandos **PARAFRENTE, PARATRÁS, PARADIREITA** e **PARAESQUERDA** para mover a figura de um lado a outro. Por exemplo:

```
RG
MUDEFIG 7
USENADA
PARAFRENTE 20
ESPERE 300
PARADIREITA 90
PARAFRENTE 90
ESPERE 300
PARATRAS 30
PARADIREITA 90 PARAFRENTE 50
```

O helicóptero (sprite pré-programado 7) parecerá levantar vôo, esperar um momento, voar para a direita, subir mais um pouco e, por fim, aterrissar.

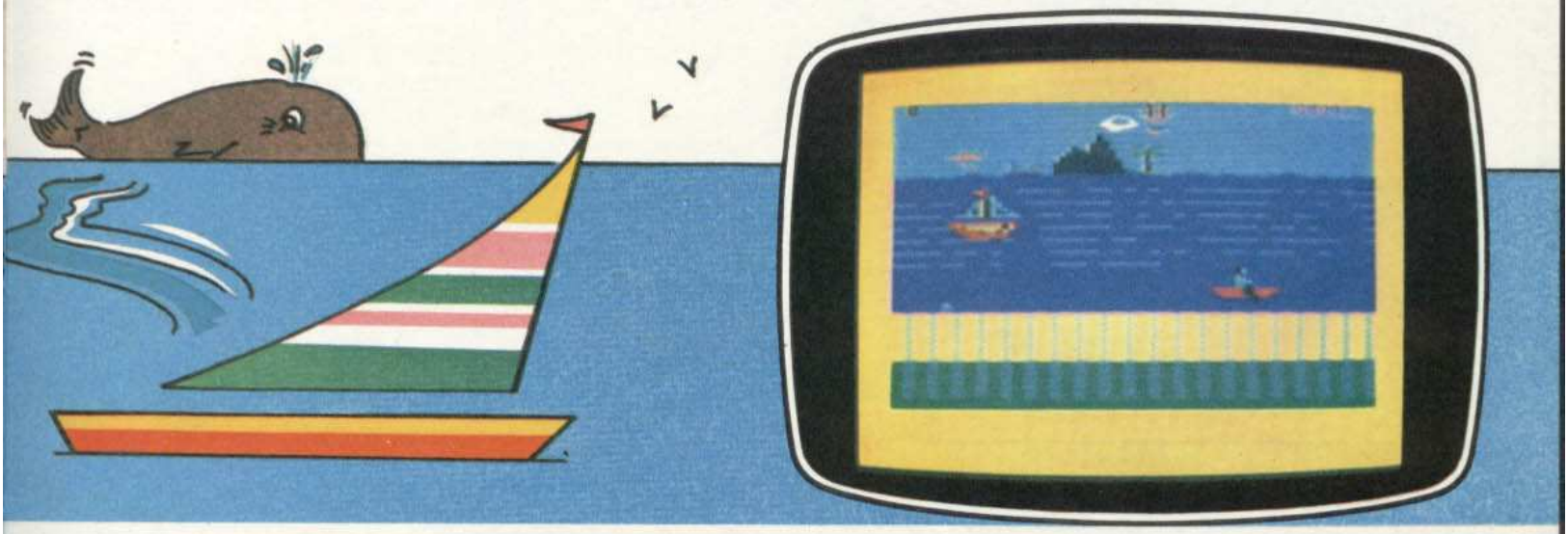
Para simular certas ações — como um homem correndo —, podemos usar dois ou mais sprites parecidos, mostrando a figura em diferentes estágios do movimento. O comando **MUDEFIG** se encarregará de alterná-los, enquanto se provoca um deslocamento na tela.


**Como funciona um sprite?**

A característica fundamental de um sprite é a sua velocidade. Aliás, é exatamente esta a origem do nome: *sprite*, em inglês, designa um pequeno duende que, segundo a lenda, corre como o vento.

Outra característica do sprite é a sua capacidade de interpenetração, ou seja, se duas figuras estiverem ocupando a mesma área de vídeo, aquela que for de menor prioridade dará a impressão de estar passando "por trás" da outra.

Teoricamente, pode-se reproduzir uma "família" de sprites através de software, mas eles não serão suficientemente velozes. Um sprite baseado em hardware, no entanto, consegue ter velocidade, pois, como é o caso dos micros da linha MSX, o vídeo dispõe de um rapidíssimo processador, que cuida de todos os detalhes do funcionamento das figuras, sem onerar o processador central.



# COMPRESSÃO DE TEXTOS (3)

**Agora que você já sabe como comprimir e descomprimir textos, ponha o programa para funcionar e observe seu desempenho no desenvolvimento de um jogo de aventura.**

Nos artigos das páginas 1332 e 1404, analisamos diversos algoritmos destinados à compressão de textos. Não explicamos, porém, como utilizar esses programas no contexto de um jogo de aventura. Como você deve se lembrar, nosso objetivo inicial era reduzir o espaço de memória ocupado por uma aventura desse tipo, permitindo a programação de um jogo mais complexo e extenso.

Neste artigo, examinaremos as técnicas de compressão e descompressão de textos aplicadas ao desenvolvimento de um jogo de aventura. Usaremos o algoritmo que demonstrou a maior eficiência de compressão, com a máxima simplicidade de programação: a técnica de compressão de dois códigos por byte, baseada na freqüência dos caracteres no texto. Como vimos, esse algoritmo pode ser programado em BASIC sem dificuldades, e é razoavelmente rápido.

## DESENVOLVIMENTO DE AVENTURAS

Em uma série de artigos de *Programação de Jogos* (páginas 208, 226, 270, 306 e 394), tratamos das principais técnicas de desenvolvimento de aventuras em BASIC. Com um número mínimo de alterações no programa então fornecido, você poderá empregar as rotinas de compressão e descompressão de textos, economizando uma quantidade razoável de memória.

Se você não armazenou o programa mencionado, digite-o agora, acompanhando as listagens dadas nos sucessivos artigos. Entretanto, cada vez que encontrar uma mensagem tal como:

```
345 PRINT "TIJOLOS SAO MUITO
PESADOS, SEU BRAÇO DEVE ESTAR
DOENDO."
```

substitua-a por outra mais curta, apenas para lembrar-se do que deve ser definido ou impresso ali. Por exemplo:

```
345 PRINT "TIJOLO PESADO"
```

Procedendo assim, você poderá testar se o programa completo de aventuras está funcionando bem, sem gastar muito tempo digitando mensagens extensas, que depois serão retiradas do programa principal.

Quais são as mensagens e frases que admitem compressão? Em um jogo de aventuras, utilizam-se doze tipos principais de texto alfanumérico:

- nomes de lugares (por exemplo: HALL DE ENTRADA);
- mensagens de localização (VOCE ESTÁ NO HALL DE ENTRADA);
- verbos (NADAR);
- nomes de objetos (REVOLVER);
- mensagens de identificação de objeto (VOCE PEGOU UM REVOLVER);
- mensagens de advertência (ESTÁ MUITO ESCURO PARA VER AS SAÍDAS);
- mensagens de erro (ISTO NÃO PODE SER ESVAZIADO);
- conseqüências de ações do jogador ou do programa (exemplo: AS BOLINHAS SE ESPALHARAM PELO CHÃO);
- perguntas ao jogador (QUER JOGAR NOVAMENTE?);
- frases comuns, em conjunto com outras mensagens (VOCE PEGOU); mensagens de ajuda (VERIFIQUE SE ALGUMA PORTA ESTÁ ABERTA); instruções do jogo.

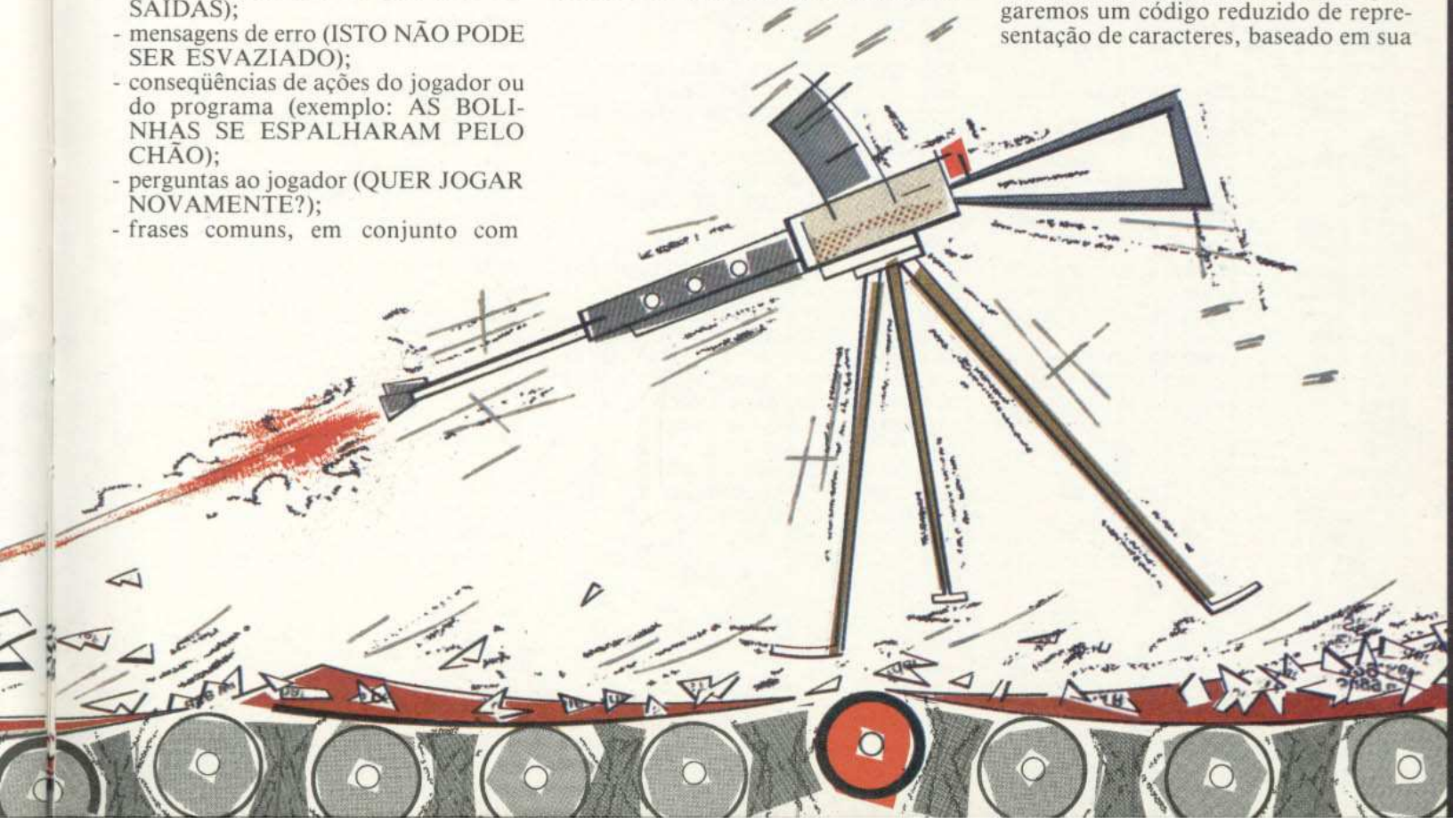
Os jogos de aventuras dificilmente contêm todos esses tipos de texto. Depois de verificar quais deles estão presentes, você deverá decidir se vale a pena comprimi-los. Listas de nomes curtos (objetos e locais, por exemplo) não devem ser comprimidas: o número de bytes economizado possivelmente não compensará os bytes de código de programa gastos para acessar o texto comprimido. Todos os outros tipos de texto (sobretudo as instruções) podem e devem ser comprimidos, desde que tenham duas ou mais palavras: isso levará a um ganho real de espaço.

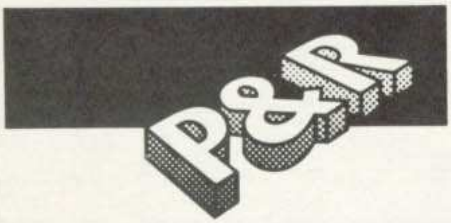
Para localizar as mensagens que serão comprimidas, o melhor é tirar uma listagem em impressora e marcar todos os pontos em que elas ocorrem. Em seguida, faça uma lista das mensagens, indicando os números das linhas do programa onde elas ocorrem. Agrupe as mensagens segundo o seu tipo: primeiro as mensagens de localização, depois as de identificação de objetos encontrados etc. Deixe as instruções para o fim. Isso facilitará a reprogramação do jogo de aventura e tornará menos cansativa a "caçada" aos erros cometidos.

## A LISTA-MESTRE

Tendo todos esses dados, digite no computador um programa em BASIC constituído apenas de linhas **DATA**, com as mensagens que serão comprimidas. Cada mensagem deve ficar em uma linha **DATA** separada.

Preparamos essa listagem para o jogo de aventura publicado em *INPUT*. Ela serve para todos os micros, e começa na linha 5000. Em algumas máquinas é possível digitar o texto em letras minúsculas e a acentuação correta, mas não o faça. Isso é importante, pois, como vimos nos artigos anteriores, empregaremos um código reduzido de representação de caracteres, baseado em sua






**É possível comprimir sons e melodias em jogos de aventuras?**

Existem técnicas bastante eficientes para a compressão de melodias, mas os algoritmos usados são extremamente especializados. Uma técnica muito simples pode ser empregada, entretanto, pelos usuários dos computadores TRS-Color e MSX: ela consiste em comprimir a seqüência de comandos usados para programar uma melodia como se fosse um texto (variável *string*), e depois descomprimi-la no momento de tocar a melodia.

freqüência no texto — diferente, portanto, do ASCII. Como a eficiência de compressão só é obtida se usarmos um máximo de 30-35 caracteres nesse código, trabalharemos apenas com as letras maiúsculas e poucos caracteres de pontuação.

```
5000 DATA "VOCE ESTA DO LADO DE
 FORA DE UM GRANDE PREDIO"
5010 DATA "VOCE ESTA A BEIRA DE
 UM GRANDE RIO"
5020 DATA "VOCE ESTA NUMA FLORE
STA PETRIFICADA"
5030 DATA "VOCE ESTA NUMA SALA
EMPOEIRADA"
5040 DATA "VOCE ESTA NUMA SALA
ESCURA"
5050 DATA "VOCE ESTA EM UM ATAL
HO ENLAMEADO"
5060 DATA "VOCE ESTA NA ENTRADA
 DA CIDADE OCULTA"
5070 DATA "VOCE ESTA NO HALL DE
 ENTRADA"
5080 DATA "VOCE ESTA NO PATIO"
5090 DATA "VOCE ESTA NO JARDIM"
5100 DATA "VOCE ESTA NO GUARDA-
LOUCAS"
5110 DATA "VOCE ESTA NA SALA DO
 TRONO"
5120 DATA "ESTA MUITO ESCURO PA
RA VER AS SAIDAS"
5130 DATA "COMO VOCE NAO TEM NA
DA QUE POSSA SER CONFISCADO, EL
E O PRENDE NUMA MASMORRA IMUNDA
"
5140 DATA "DESCULPE, VOCE NAO P
ODE SEGUIR NESSA DIRECAO"
5150 DATA "VOCE NAO PODE LARGAR
 O QUE NAO TEM"
5160 DATA "DESCULPE, NAO POSSO
AJUDAR AGORA"
5170 DATA "TIJOLOS SAO MUITO PE
SADOS, SEU BRACO DEVE ESTAR DOE
NDO"
5180 DATA "JA ESTA ACESA"
5190 DATA "ISTO NAO PODE SER ES
VAZIADO"
5200 DATA "AS BOLINHAS SE ESPAL
HARAM PELO CHAO"
5210 DATA "NADAR AONDE ?"
5220 DATA "QUE VERGONHA, VOCE S
E AFOGOU !"
5230 DATA "VOCE SE MOLHOU TODO"
5240 DATA "VOCE ACHOU UM REVOLV
ER"
5250 DATA "NADA ACONTECE"
5260 DATA "VOCE NAO PODE PUXAR
ISTO"
5270 DATA "VOCE CAIU DENTRO DO
VASO E FOI EMBORA COM A DESCARG
A !"
5280 DATA "PARABENS ! VOCE COMP
LETOU A TAREFA !"
5290 DATA "FIM DO PROGRAMA DE A
VENTURAS"
5300 DATA "QUER JOGAR NOVAMENTE
 (S/N) ?"
5310 DATA "HA UM SACO DE BOLAS
DE GUDE AQUI"
5320 DATA "TEM UM TIJOLO NO CHA
O"
5330 DATA "HA UMA CORRENTE PEND
URADA SOBRE O TRONO"
5340 DATA "TEM UM REVOLVER NO C
HAO"
5350 DATA "UM OLHO CRAVEJADO DE
 BRILHANTES ESTA NO CHAO"
5360 DATA "VOCE ESTA DIANTE DE
UMA LAMPADA"
5370 DATA "DE REPENTE SURGE UM
COLETOR DE IMPOSTOS"
5380 DATA "EU NAO SEI COMO"
5390 DATA "VOCE PEGOU"
5400 DATA "NAO ESTA AQUI"
5405 DATA "PRESSIONE QUALQUER T
ECLA PARA CONTINUAR..."
5410 DATA "INSTRUCOES"
5420 DATA "DEVIDO A UM COLAPSO
FINANCEIRO, VOCE TEVE QUE DEIXA
R O PAIS. SEUS PROBLEMAS VAO TE
RMINAR QUANDO VOCE ENCONTRAR O
LEGENDARIO OLHO CRAVEJADO DE BR
ILHANTES DE UM TOTEM INCA."
5430 DATA "DEPOIS DE TRAZE-LO,
VOCE TERA QUE ENCONTRAR A SAIDA
  CUIDADO COM O COLETOR DE IMPO
STOS !"
5440 DATA "*"
```

Nesse programa, foram codificados estes tipos de texto (com suas linhas):

5000 a 5110: mensagens de localização;
5120 a 5260: mensagens de advertência e erro;
5270 a 5370: conseqüências de ações;
5380 a 5405: frases comuns e comandos;
5410 a 5440: instruções do jogo.

Observe que só excluímos as listas de objetos, verbos e locais. As linhas com instruções são as mais longas (cuidado para não ultrapassar 255 caracteres), pois isso facilita seu posterior uso.

Depois de digitar o programa, armazene-o em fita ou disco.

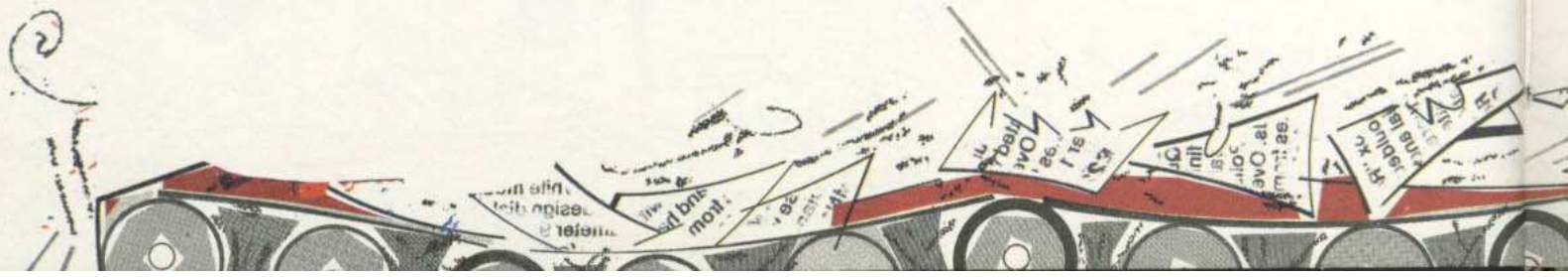
## CODIFICAÇÃO

A técnica de compressão estatística utiliza um código baseado na lista dos caracteres mais freqüentes no texto. Cada código ocupa um nibble (ou seja, quatro bits), e dois nibbles são comprimidos em um byte. Os quinze caracteres mais comuns têm códigos de um nibble (um número de 1 a 15). Os quinze caracteres seguintes, na ordem de freqüência, têm códigos de dois nibbles (um 0, seguido do código de 1 a 15); os próximos quinze têm códigos de três nibbles (dois 0, seguidos de um código de 1 a 15) e assim por diante. Em geral, os quinze caracteres mais freqüentes correspondem a 80% ou mais de todo o texto, e é por isso que se obtém uma compressão em torno dos 50%.

Se usarmos uma lista dos caracteres mais freqüentes em textos em português, conseguiremos uma boa eficiência de compressão, mas não a ideal. Para alcançar o nível máximo de eficiência, devemos usar a ordenação de caracteres encontrada no próprio texto. Se você quiscr determiná-la, siga estas etapas:

- carregue no computador as linhas **DATA** com todas as mensagens;
- acrescente as linhas do programa de contagem de letras fornecido no primeiro artigo da série (página 1332);
- rode o programa e anote o resultado.

Para poupar trabalho ao leitor, preparamos a lista de freqüência de caracteres no texto da aventura de *INPUT* (artigo da página 208 e seguintes).



```
branco 248     H    17
A      176     G    15
E      173     Q    11
O      163     B    10
S       83     F     9
R       79     J     8
D       73     ,     7
C       68     .     6
N       67     !     5
T       64     -     2
U       57     ?     2
I       47     X     2
M       47     Z     2
V       42     (     1
L       38     )     1
P       33     /     1
```

O primeiro conjunto de quinze caracteres (**K$(1)**) é formado por: branco, A,E,O,S,R,D,C,N,T,U,I,M,V e L. Os demais caracteres fazem parte, portanto, dos conjuntos seguintes.

De posse dessa lista, podemos escrever um pequeno programa para testar se a compressão e a descompressão estão funcionando com os textos de exemplo. Carregue novamente na memória somente as linhas **DATA** anteriores, e adicione este programa:

```
10 DIM F$(100),K$(3)
20 K$(1)=" AEOSRDCNTUIMVL":K$(2
)="PHGQBFJ,.!-?XZ(":K$(3)=")/"
180 PRINT "CODIFICANDO..."
190 I=0:K=1
200 READ L$:IF L$="*" THEN GOTO
 230
210 GOSUB 710:I=I+1:F$(I)=X$
215 PRINT I;L$
220 GOTO 200
230 NL=I
240 PRINT:INPUT "NUMERO DA MENS
AGEM ";I
242 IF I=0 THEN GOTO 300
245 IF I<1 OR I>NL THEN PRINT "
*** NAO EXISTE":GOTO 240
250 L$=F$(I):GOSUB 870
290 GOTO 240
300 END
```

Para os computadores das linhas TRS-80 e TRS-Color, adicione a linha:

```
5 CLEAR 2000
```

```
10 DIM f$(100),k$(3,15)
20 LET k$(1)=" AEOSRDCNTUIMVL":
LET k$(2)="PHGQBFJ,.!-?XZ(":LET
 k$(3)=")/"
180 PRINT "CODIFICANDO..."
190 LET i=0:LET k=1
200 READ L$:IF L$="*" THEN GOTO
 230
210 GOSUB 710:LET i=i+1:LET f$(
i)=x$
215 PRINT i;L$
220 GOTO 200
230 LET nL=I
240 PRINT:INPUT "NUMERO DA MENS
AGEM ";i
242 IF i=0 THEN GOTO 300
245 IF i<1 OR i>nL THEN PRINT "
*** NAO EXISTE":GOTO 240
250 LET L$=f$(i):GOSUB 870
290 GOTO 240
300 STOP
```

As sub-rotinas que começam nas linhas 710 (codificação) e 870 (decodificação) podem ser copiadas integralmente do programa listado no primeiro artigo, para os computadores respectivos.

A linha 20 do programa define os três conjuntos de códigos de caracteres. O laço formado pelas linhas 180 a 220 comprime o texto, lendo-o linha por linha em **DATA** e colocando o resultado no conjunto **F$** (uma mensagem por linha). A sub-rotina de codificação é chamada em 210.

Finalmente, as linhas 240 a 290 solicitam ao usuário o número da mensagem que deseja imprimir. Digite 0, se quiser interromper o programa.

## ARMAZENAGEM DO RESULTADO

Vamos agora desenvolver dois programas separadamente. O primeiro funciona como o programa anterior, servindo para comprimir todas as mensagens e instruções. O resultado, gravado em fita ou disco, em um arquivo seqüencial, será depois carregado pelo programa de jogo propriamente dito. Assim, a rotina que executa a compressão não precisa ser incluída no programa do jogo. Neste se introduzem apenas as rotinas responsáveis pelo carregamento, descompressão e impressão.

O segundo programa compõe-se de duas rotinas que podem ser acrescentadas ao jogo de aventura. Uma delas lê o arquivo seqüencial com o texto comprimido; a outra efetua a descompressão quando solicitado. Explicaremos mais adiante como usá-las.

No programa-exemplo, o texto comprimido foi armazenado em um conjunto **F$**. Portanto, bastaria escrever uma sub-rotina de gravação desse conjunto (com texto comprimido) e, depois, uma sub-rotina para lê-lo, de volta à memória, no programa de jogo.

Parece tudo muito simples, mas há um senão: trabalhando com conjuntos alfanuméricos, como o **F$**, no programa que utilizará o texto comprimido, ocuparemos um espaço de memória, que ultrapassará, em tamanho, o exigido pelo texto original, descomprimido.

Devido ao modo como o BASIC trata cadeias de caracteres (*string*), é difícil evitar esse problema. Usaremos, assim, um conjunto numérico **T%** para armazenar os bytes comprimidos. Os dois bytes gerados pela compressão de cada quatro nibbles serão armazenados em um conjunto de **T%**, pois uma variável inteira (em todos os computadores, exceto o Spectrum) ocupa dois bytes, e pode armazenar um número entre −32768 e 32767. O tamanho de **T%** será igual ao número de caracteres comprimidos (no exemplo anterior, 447 bytes).

Para localizar o início de cada mensagem nessa seqüência contínua de códigos, temos que criar um segundo con-

## MICRODICAS

### ACENTUANDO TEXTOS COMPRIMIDOS

Para os perfeccionistas da programação, é frustrante não poder usar caracteres acentuados em um texto que será comprimido pelo algoritmo estatístico discutido neste artigo. Mas, em geral, a falta de acentos não dificulta a leitura do texto.

Entre as poucas exceções destaca-se a palavra *é*, que, sem acento, pode impedir a compreensão do texto. Para evitar esse problema sem aumentar o número de caracteres do conjunto utilizado, recorra ao apóstrofo: coloque-o logo após a letra que deve ter acento agudo, sem deixar espaço — o resultado será satisfatório.

junto **A%**, que conterá os apontadores, ou índices de **T%**. O comprimento de **A%** será igual ao número de mensagens mais 1 (no nosso exemplo, 46).

Se dimensionamos **T%** e **A%** em seu limite, obteremos o máximo de economia de espaço de memória.

A rotina de codificação é a seguinte (o programa de teste vem depois):

```
700 REM - ROTINA DE CODIFICACAO
710 N=0:LN=0
720 FOR J=1 TO LEN(L$)
730 C$=MID$(L$,J,1)
740 P=INSTR(K$(K),C$)
750 N=N+1:C%(N)=P:LN=N
755 IF N<4 THEN GOTO 775
760 T%(NC)=(C%(1) OR (16*C%(2)
))+256*(C%(3) OR (16*C%(4)))-
32768
765 NC=NC+1:N=0
775 IF P=0 THEN K=K+1:GOTO 740
776 K=1
780 NEXT J
785 IF LN=4 THEN RETURN
790 FOR J=LN+1 TO 4:C%(J)=1
792 NEXT J
795 T%(NC)=(C%(1) OR (16*C%(2)
))+256*(C%(3) OR (16*C%(4)))-
32768
796 NC=NC+1:RETURN
```

Para executar a rotina nos micros compatíveis com o Apple, acrescente:

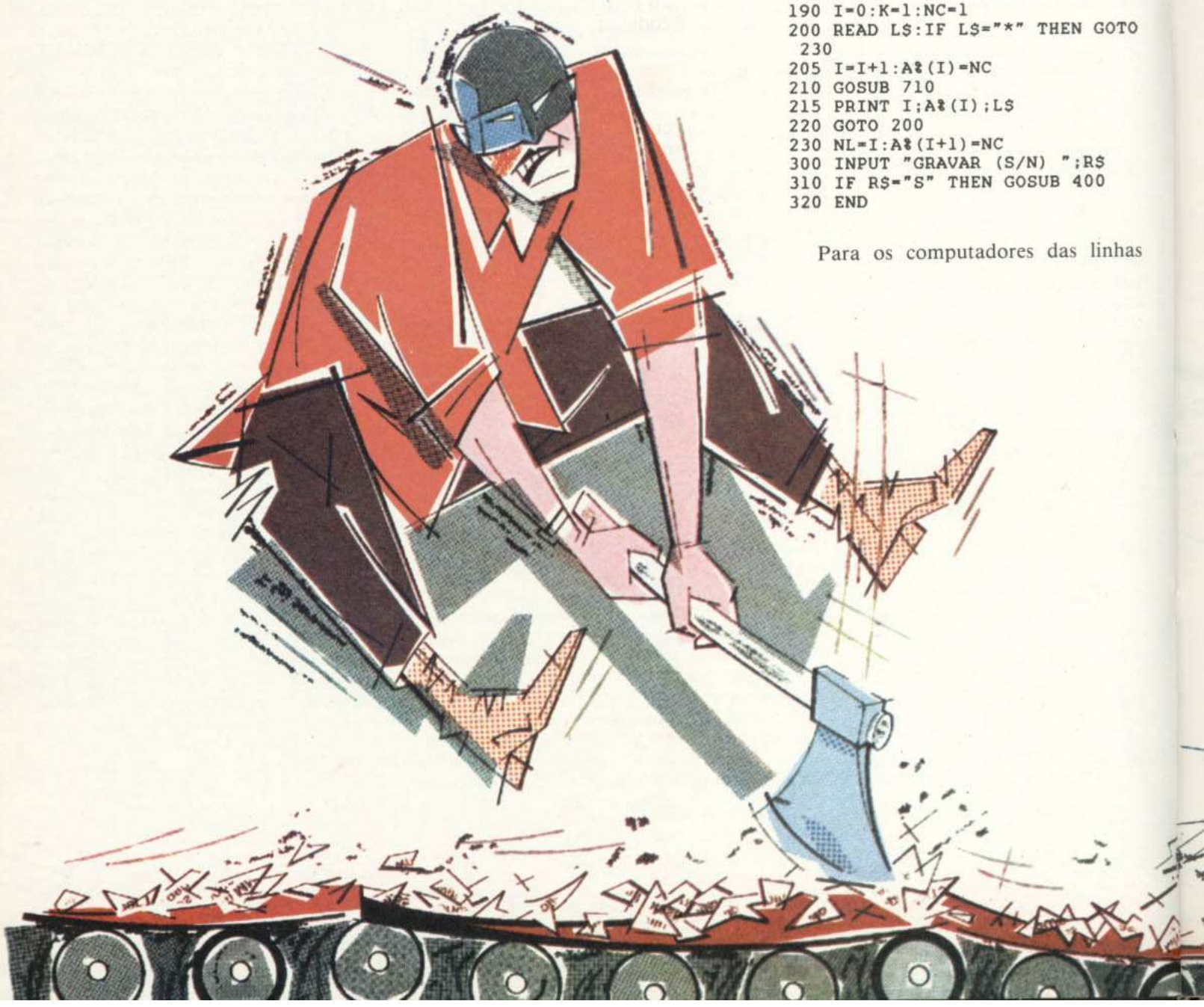
```
740 FOR P=1 TO 15:IF C$=MID$(K$
(K),P,1) THEN GOTO 750
745 NEXT P:P=0
```

A sub-rotina é igual à apresentada no artigo anterior, com uma diferença: os nibbles são armazenados de quatro em quatro, no conjunto **C%**, e comprimidos pelas linhas 760 e 795. O valor 32768 é diminuído do byte assim comprimido, para que seu conteúdo fique entre −32768 e 32767.

Segue-se o programa principal:

```
10 DIM K$(3),T%(450),A%(46)
15 DIM N%(2),C%(4)
20 K$(1)=" AEOSRDCNTUIMVL":K$(2
2)="PHGQBFJ,.!-?XZ(":K$(3)=")/"
180 PRINT "CODIFICANDO..."
190 I=0:K=1:NC=1
200 READ L$:IF L$="*" THEN GOTO
 230
205 I=I+1:A%(I)=NC
210 GOSUB 710
215 PRINT I;A%(I);L$
220 GOTO 200
230 NL=I:A%(I+1)=NC
300 INPUT "GRAVAR (S/N) ";R$
310 IF R$="S" THEN GOSUB 400
320 END
```

Para os computadores das linhas



TRS-80 e TRS-Color, acrescente a linha:

```
5 CLEAR 2000
```

A sub-rotina para o armazenamento em arquivo seqüencial começa na linha 400, e grava os conjuntos **T** (com os códigos comprimidos) e **A** (apontadores):

```
400 REM - ROTINA DE GRAVACAO
410 INPUT "NOME DO ARQUIVO";N$
420 OPEN "O",#-1,N$
430 PRINT#-1,NL,NC
440 FOR I=1 TO NL+1:PRINT#-1,A%
(I):NEXT I
450 FOR I=1 TO NC:PRINT#-1,T%(I
):NEXT I
460 CLOSE#-1:RETURN
```

```
400 REM - ROTINA DE GRAVACAO
410 INPUT "GRAVADOR PRONTO ";N$
430 PRINT#-1,NL,NC
440 FOR I=1 TO NL+1:PRINT#-1,A%
(I):NEXT I
450 FOR I=1 TO NC:PRINT#-1,T%(I
):NEXT I
460 RETURN
```

```
400 REM - ROTINA DE GRAVACAO
410 INPUT "NOME DO ARQUIVO";N$
420 OPEN "CAS:"+N$ FOR OUTPUT
AS #1
430 PRINT#1,NL,NC
440 FOR I=1 TO NL+1:PRINT#1,A%
(I):NEXT I
450 FOR I=1 TO NC:PRINT#1,T%(I
):NEXT I
460 CLOSE#1:RETURN
```

```
400 REM - ROTINA DE GRAVACAO
410 N%(1)=NL:N%(2)=NC
420 INPUT "GRAVADOR PRONTO ";N$
430 STORE N%:STORE A%:STORE T%
460 RETURN
```

Para executar o programa completo de compressão, acrescente ao final as linhas **DATA**. O computador executará toda a tarefa automaticamente. À medida que codifica as linhas, o programa as exibe na tela, perguntando se o usuário deseja gravar o resultado. Se a resposta for **S**, os conjuntos **A** e **T** serão armazenados para uso posterior pela rotina de decodificação.

No Spectrum não há tratamento separado para arquivos seqüenciais em fita, a não ser que se disponha de um Microdrive (não existente no Brasil). Por isso, o programa para essa máquina apresenta uma abordagem diferente dos demais micros.

Inicialmente, carregue o jogo de aventura completo e acrescente as linhas abaixo (renumere-as para compatibilizar com o programa do jogo, se isto se fizer necessário):

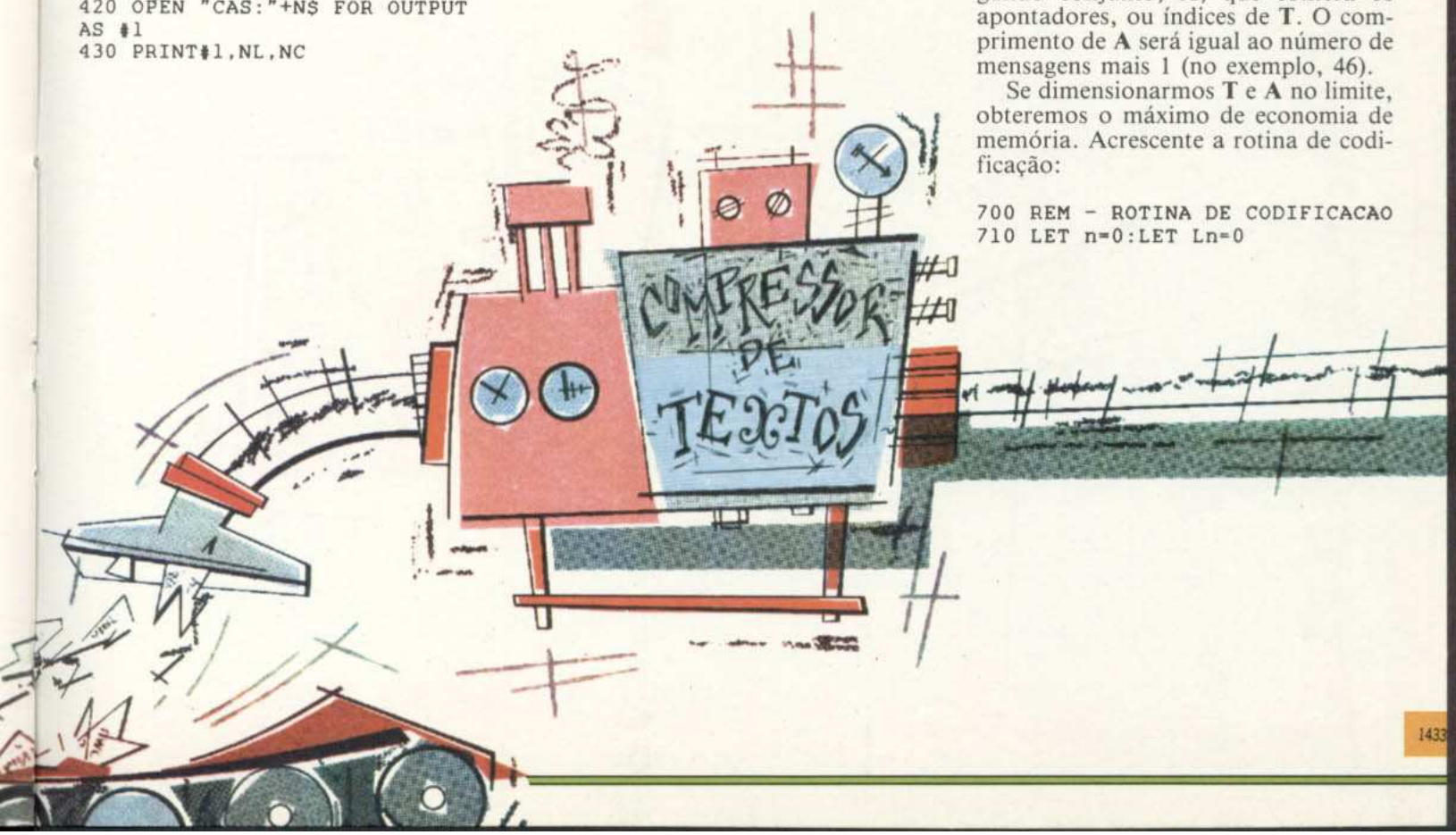
```
10 DIM k$(3,15),t(450),a(46)
15 DIM c(4)
20 LET k$(1)=" AEOSRDCNTUIMVL":
LET k$(2)="PHGQBFJ,.!-?XZ(":LET
 k$(3)=")/"
190 LET i=0:LET k=1:LET nc=1
200 PRINT "ENTRE A MENSAGEM ";:
LINE INPUT L$:IF L$="*" THEN GO
TO 230
205 LET i=i+1:LET a(i)=nc
210 GOSUB 710
215 PRINT i;a(i);L$
220 GOTO 200
230 LET n1=i:LET a(i+1)=nc
240 STOP
```

O programa principal pedirá ao usuário para entrar as mensagens pelo teclado, uma de cada vez. Quando chegar à última, responda com um asterisco, para assinalar o fim.

Usaremos um conjunto numérico **T** para armazenar os bytes comprimidos. Os dois bytes gerados pela compressão de cada quatro nibbles serão armazenados em um elemento de **T**. O tamanho desse conjunto será igual ao número de caracteres comprimidos (no nosso exemplo, 447 bytes). Para localizar o início de cada mensagem nessa seqüência contínua de códigos, temos que criar um segundo conjunto, **A**, que conterá os apontadores, ou índices de **T**. O comprimento de **A** será igual ao número de mensagens mais 1 (no exemplo, 46).

Se dimensionarmos **T** e **A** no limite, obteremos o máximo de economia de memória. Acrescente a rotina de codificação:

```
700 REM - ROTINA DE CODIFICACAO
710 LET n=0:LET Ln=0
```



```
720 FOR j=1 TO LEN L$
730 LET c$=L$(j)
740 FOR p=1 TO 15:IF c$=k$(k,p)
 THEN GOTO 750
745 NEXT p:LET p=0
750 LET n=n+1:LET c(n)=P:LET L
n=n
755 IF n<4 THEN GOTO 775
760 LET t(nc)=(c(1) OR (16*c
(2)))+256*(c(3) OR (16*c(4)))
-32768
765 LET nc=nc+1:LET n=0
775 IF p=0 THEN LET k=k+1:GOTO
740
776 LET k=1
780 NEXT j
785 IF Ln=4 THEN RETURN
790 FOR j=Ln+1 TO 4:c(j)=1:NEX
T j
795 LET t(nc)=(c(1) OR (16*c
(2)))+256*(c(3) OR (16*c(4)))
-32768
796 LET nc=nc+1:RETURN
```

A sub-rotina é igual à apresentada no artigo anterior, com uma diferença: os nibbles são armazenados de quatro em quatro, no conjunto **C%**, e comprimidos pelas linhas 760 ou 795. O valor 32768 é diminuído do byte assim comprimido, para que seu conteúdo fique entre −32768 e 32767.

Depois de acrescentar o programa principal e a rotina de decodificação ao jogo da aventura, execute-o. Após entrar todas as mensagens, grave o programa e os dados juntos, em fita.

## DESCOMPRESSÃO

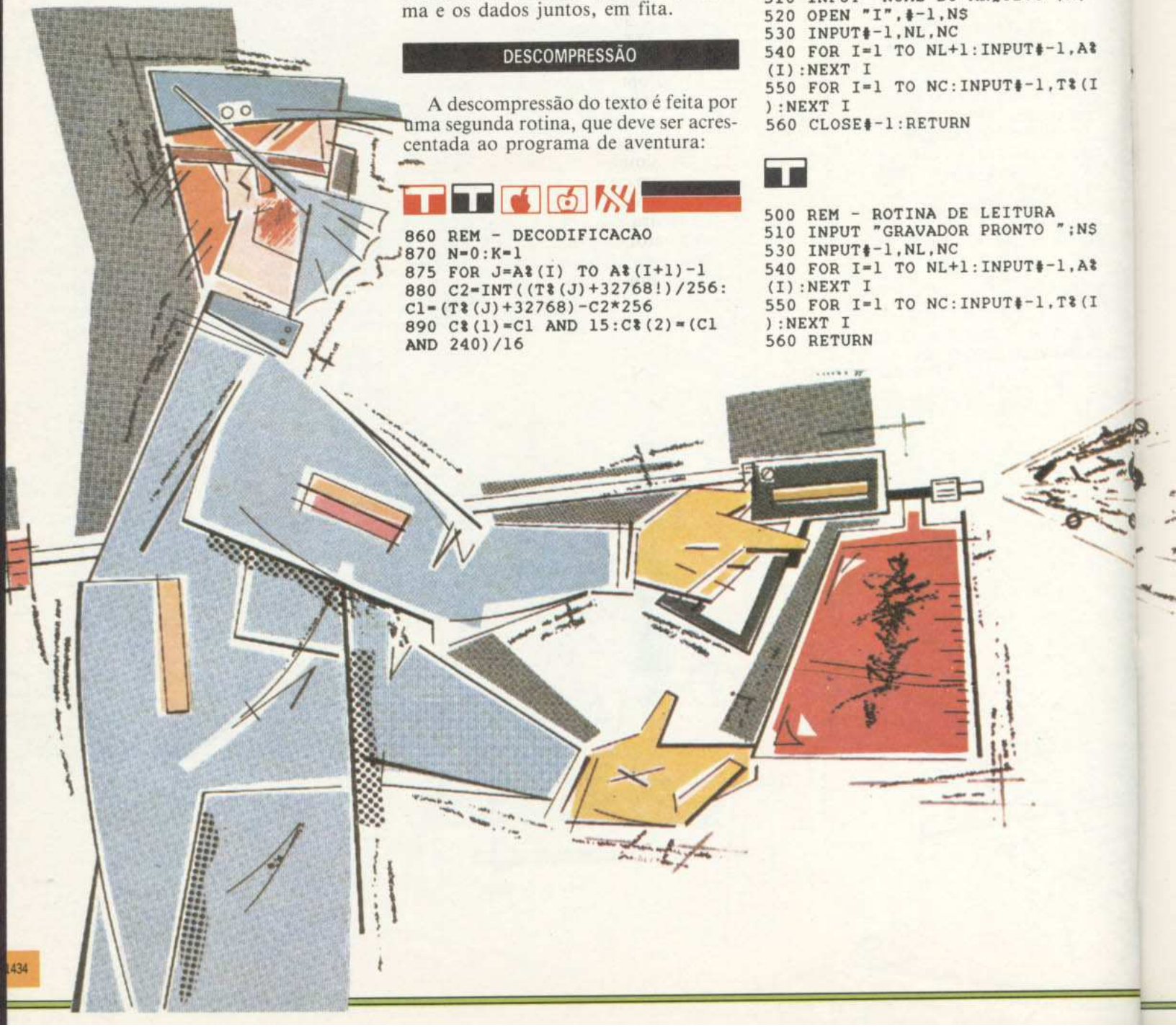
A descompressão do texto é feita por uma segunda rotina, que deve ser acrescentada ao programa de aventura:

```
860 REM - DECODIFICACAO
870 N=0:K=1
875 FOR J=A%(I) TO A%(I+1)-1
880 C2=INT((T%(J)+32768!)/256:
C1=(T%(J)+32768)-C2*256
890 C%(1)=C1 AND 15:C%(2)=(C1
AND 240)/16
895 C%(3)=C2 AND 15:C%(4)=(C2
AND 240)/16
900 FOR L=1 TO 4
910 IF C%(L)=0 THEN K=K+1:GOTO
 930
920 PRINT MID$(K$(K),C%(L),1);:
K=1
930 NEXT L:NEXT J
950 PRINT:RETURN
```

Complete o programa com a rotina de carregamento dos conjuntos **A** e **T**, com os dados no arquivo seqüencial:

```
500 REM - ROTINA DE LEITURA
510 INPUT "NOME DO ARQUIVO";N$
520 OPEN "I",#-1,N$
530 INPUT#-1,NL,NC
540 FOR I=1 TO NL+1:INPUT#-1,A%
(I):NEXT I
550 FOR I=1 TO NC:INPUT#-1,T%(I
):NEXT I
560 CLOSE#-1:RETURN
```

```
500 REM - ROTINA DE LEITURA
510 INPUT "GRAVADOR PRONTO ";N$
530 INPUT#-1,NL,NC
540 FOR I=1 TO NL+1:INPUT#-1,A%
(I):NEXT I
550 FOR I=1 TO NC:INPUT#-1,T%(I
):NEXT I
560 RETURN
```



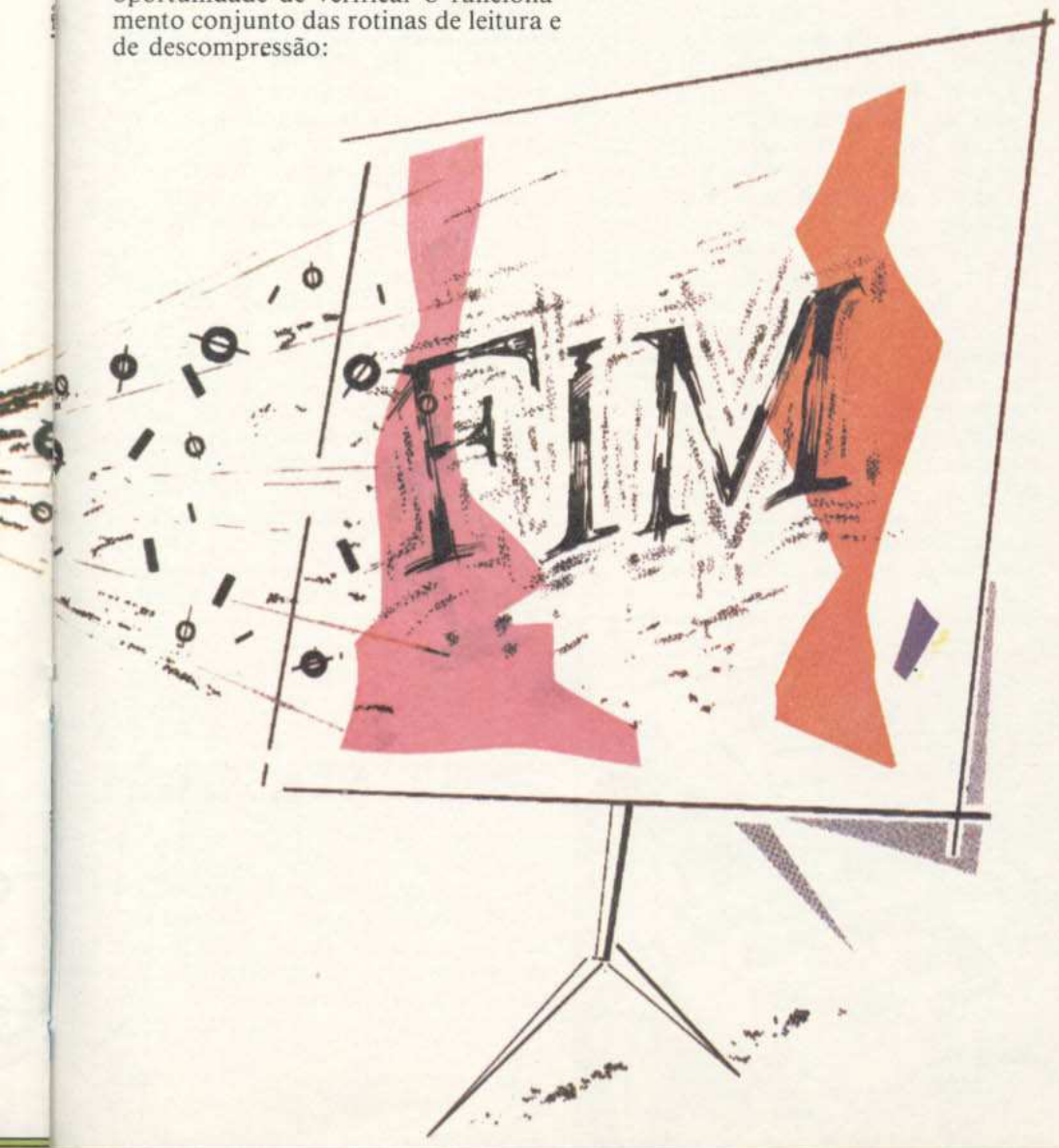
```
500 REM - ROTINA DE LEITURA
510 INPUT "NOME DO ARQUIVO";N$
520 OPEN "CAS:"+N$ FOR INPUT AS
#1
530 INPUT#1,NL,NC
540 FOR I=1 TO NL+1:INPUT#1,A%
(I):NEXT I
550 FOR I=1 TO NC:INPUT#1,T%(I
):NEXT I
560 CLOSE#1:RETURN
```

```
500 REM - ROTINA DE LEITURA
510 INPUT "GRAVADOR PRONTO ";N$
530 RECALL N%:RECALL A%:RECALL
T%:NL=N%(1):NC=N%(2)
560 RETURN
```

Apresentamos a seguir um pequeno programa de teste. Com ele, você terá oportunidade de verificar o funcionamento conjunto das rotinas de leitura e de descompressão:

```
10 DIM K$(3),T%(450),A%(46)
20 K$(1)=" AEOSRDCNTUIMVL":K$(2
)="PHGQBFJ,.!-?XZ(":K$(3)=")/"
30 GOSUB 510
240 PRINT:INPUT "NUMERO DA MENS
AGEM ";I
242 IF I=0 THEN GOTO 300
245 IF I<1 OR I>NL THEN PRINT "
*** NAO EXISTE":GOTO 240
250 GOSUB 870
290 GOTO 240
300 END
```

As linhas 10, 20 e 30 desse programa devem ser colocadas no início do programa do jogo. Elas dimensionam os conjuntos de trabalho do descompressor de textos e chamam a rotina 510. Esta carrega os conjuntos **T** e **A**, a partir do texto comprimido que está armazenado na memória auxiliar.




As linhas 240 a 300 simplesmente imprimem a mensagem solicitada pelo usuário e mostram como a sub-rotina 870 (de decodificação) deve ser usada dentro do programa de aventura.

## COMO ADAPTAR O PROGRAMA

Carregue de novo o programa de aventura e localize as linhas que contêm mensagens incluídas na lista de compressão. Suponhamos que você encontre a seguinte linha (abreviada, como recomendamos no começo deste artigo):

```
345 PRINT "TIJOLO PESADO"
```

Substitua-a por:

```
345 LET I=17:GOSUB 870
```

Essa linha iguala o apontador **I** ao número da mensagem na lista comprimida, e passa o controle para a sub-rotina de descompressão. Esta localiza o início e o fim da mensagem, no conjunto de apontadores **A**, e imprime na tela o texto descomprimido.

Repita esse procedimento com todas as linhas onde ocorrem mensagens, e grave o resultado final. Não se esqueça de adicionar as linhas 10, 20 e 30 do nosso exemplo (renumeradas, se for o caso).

Para o Spectrum, o esquema é um pouco diferente. Carregue o programa de aventura que gravou em fita, juntamente com o programa de compressão e o texto comprimido. Apague as linhas 190 a 796 e modifique cada uma das linhas com mensagens, como foi explicado antes. Em seguida, acrescente:

```
860 REM - DECODIFICACAO
870 LET n=0:LET k=1
875 FOR j=a(i) TO a(i+1)-1
880 LET c2=INT((t(j)+32768)/256
:LET c1=(t(j)+32768)-c2*256
890 LET c(1)=c1 AND 15:LET c(2)
=(c1 AND 240)/16
895 LET c(3)=c2 AND 15:LET c(4)
=(c2 AND 240)/16
900 FOR L=1 TO 4
910 IF c(L)=0 THEN LET k=k+1:G
OTO 930
920 PRINT K$(K,c(L));:LET k=1
930 NEXT L:NEXT j
950 PRINT:RETURN
```

Armazene a nova versão em fita. Para rodar o programa completo, use um **GOTO** para a primeira linha do programa de aventura (não a linha 10 do programa de codificação). Nunca empregue o comando **RUN**, pois você perderá todo o texto das mensagens.

# PROGRAMAÇÃO EM PASCAL

**Muitos consideram o Pascal como a melhor alternativa para o BASIC. Com seus poderosos recursos, podem-se elaborar programas bem estruturados e fáceis de documentar e testar.**

Para quem está familiarizado com apenas uma linguagem de programação, o aprendizado de outra pode apresentar maior ou menor dificuldade — o que depende, sobretudo, do grau de semelhança entre as duas linguagens.

Se a estrutura e sintaxe de ambas forem parecidas, basta fazer uma tradução: o novo "vocabulário" é aprendido, e as regras de formação de instruções são aplicadas da mesma maneira. As vezes, porém, a diferença entre as linguagens é tão grande, que o aprendiz se vê forçado a abrir mão das regras que conhece e adotar um outro raciocínio.

A linguagem LOGO, por exemplo, examinada em artigos anteriores, foi projetada para facilitar ao máximo a introdução de iniciantes ao mundo da programação. Entretanto, para quem foi "criado" em BASIC — como a maioria dos usuários de micros pessoais —, a adaptação pode apresentar problemas. Algumas vezes, chega a ser difícil aceitar as afirmativas de que LOGO foi feito para principiantes!

Isso ocorre, fundamentalmente, porque o LOGO tem uma estrutura interna bastante diferente da do BASIC, embora seu vocabulário seja mais simples. LOGO é uma linguagem mais bem estruturada, e trabalha com listas hierárquicas, como o LISP. Essas características estimulam o programador iniciante a adotar padrões estruturados de raciocínio e de resolução de problemas.

A programação estruturada também é um aspecto importante de uma linguagem imperativa, o Pascal. Como vimos no artigo da página 1288, ele faz parte da família de linguagens algorítmicas e declarativas iniciadas com o ALGOL. Sua filosofia pode parecer totalmente estranha ao programador BASIC. Ainda que experiente, este está acostumado a elaborar programas no teclado, testando e alterando seções à medida que progride, até chegar a um resultado que, muitas vezes, evidencia um objetivo não muito claro no começo do trabalho. O Pascal só lhe parecerá menos complicado se ele tiver desenvolvido hábitos de programação estruturada — em BASIC ou outra linguagem.

Por que então dar-se ao trabalho de aprender Pascal? Além de permitir a elaboração de programas mais bem estruturados e de simplificar a documentação e a execução de testes, essa linguagem apresenta certas vantagens sobre o BASIC. Em programas muito complexos, o Pascal produz um código mais claro e curto. A facilidade em definir procedimentos possibilita a programação modular e a formação de bibliotecas de procedimentos, que podem ser aproveitados integralmente em outros programas. Finalmente, as versões modernas do Pascal são adequadas tanto para programação científica (cálculos matemáticos) quanto para o desenvolvimento de aplicações comerciais.


## FUNDAMENTOS DO PASCAL

O Pascal foi criado em 1970 pelo professor Niklaus Wirth, do Instituto Federal de Tecnologia em Zurique, Suíça. Seu nome é uma homenagem a Blaise Pascal, filósofo e matemático do século XVII, responsável por importantes descobertas científicas e inventor de uma das primeiras máquinas de calcular mecânicas, a *Pascalina*.

O objetivo de Wirth era desenvolver uma linguagem destinada a ensinar os conceitos fundamentais de estruturas de programação. Essa linguagem deveria estimular o estudante a elaborar a estrutura do programa antes de começar a escrevê-lo. Nela Wirth introduziu uma série de recursos a fim de permitir que a solução programada contenha a própria informação a ser processada.

Em outras palavras, para se escrever um programa em Pascal é necessário planejar como resolver o problema. A solução escolhida é refinada sucessivamente, cobrindo detalhes cada vez menores, até se alcançar o nível de um procedimento (unidade algorítmica). Só se trabalha no programa principal depois que todos os procedimentos necessários estiverem programados.

Para desenvolver o Pascal, Wirth utilizou muitas das idéias em que se baseia o ALGOL 606. Originalmente orientado para o ensino de linguagens estruturadas em computadores de grande porte, o PASCAL foi adaptado para uso em microcomputadores. Hoje, é extensamente empregado tanto no ensino quanto em aplicações comerciais em micros.

## PASCAL PARA MICROS

Existem versões do Pascal para diversas linhas de micros pessoais e profissionais. As versões reduzidas (que não têm todos os comandos do Pascal padronizado), chamadas Tiny-Pascal, podem ser carregadas de fita cassete. Porém, o Pascal é mais eficiente quando usado com sistemas baseados em disco, como o UCSD Pascal, para o Apple.

Como são muitas as versões comerciais dessa linguagem, não forneceremos

listagens específicas para cada linha de microcomputador — como fizemos para o BASIC e o LOGO. Mas, mesmo que não tenha um compilador Pascal, não se preocupe: você poderá entender os elementos básicos da linguagem, sem precisar rodar o programa em um computador. Afinal, o desenvolvimento de programas longe da máquina é uma das principais características do Pascal.

Existe outra diferença fundamental entre o BASIC usado em microcomputadores pessoais e o Pascal: ele é uma linguagem *compilada*.

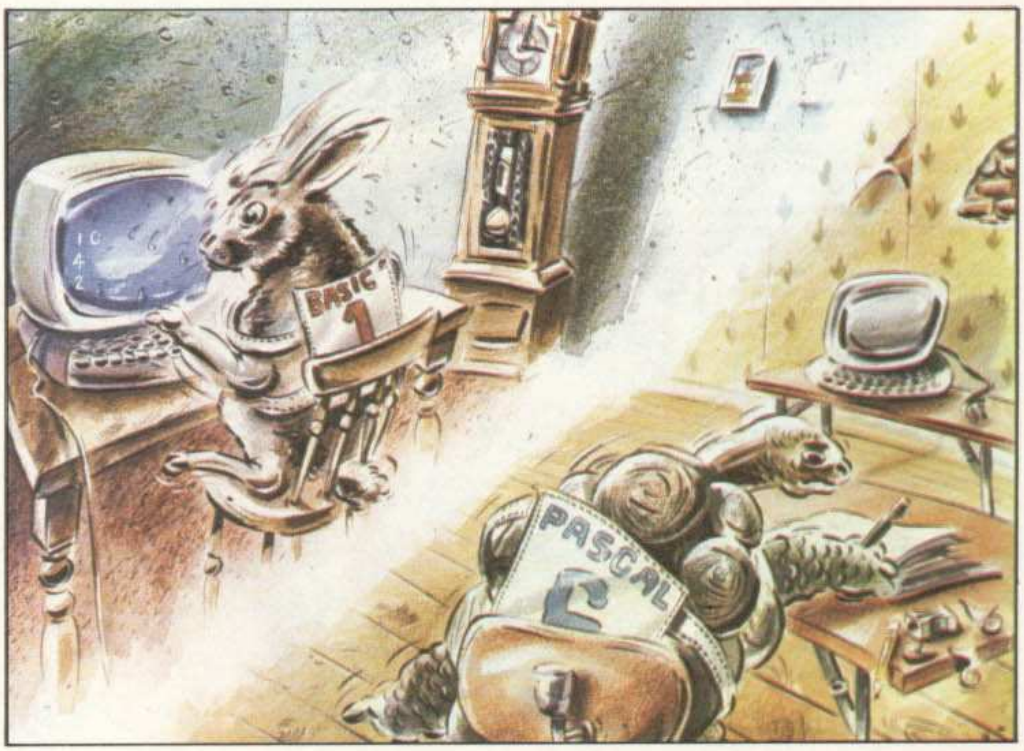
O BASIC adotado na maioria dos micros é uma linguagem *interpretada* — cada vez que o interpretador encontra uma instrução em BASIC, converte-a para códigos de máquina e a executa. Assim, para um laço simples como:

```
10 FOR N=1 TO 100
20 PRINT N
30 NEXT N
```

cada declaração nas linhas 10 a 30 será traduzida cem vezes para código de máquina! Não surpreende, portanto, que o BASIC seja considerado uma linguagem lenta. Outra desvantagem é que o interpretador precisa permanecer na memória o tempo todo, ocupando um espaço valioso, que poderia ser utilizado pelo programa ou dados do usuário.

Uma linguagem compilada funciona de maneira diferente. Uma vez que se tenha entrado o programa na máquina, na forma de um texto (o *programa-fonte*), ele é traduzido de uma vez só para código de máquina (o *programa-objeto*), e é este que o computador executa. Depois de feita a tradução, o programa compilador não é mais necessário, ficando armazenado em fita ou disco. Em conseqüência, mais espaço na memória é liberado para uso do programa aplicativo desenvolvido pelo usuário.

As linguagens compiladas são muito mais rápidas que as interpretadas, pois o demorado processo de tradução para código de máquina é executado apenas uma vez. Embora existam compiladores para o BASIC, essa linguagem não foi projetada para ser compilada; assim, seus programas-objeto são longos e, freqüentemente, tão lentos quanto o programa-fonte. O Pascal, ao contrário, foi projetado especificamente para ser compilado, permitindo a obtenção de programas tão compactos e rápidos quanto os escritos diretamente em Assembler. E há uma vantagem adicional: é mais simples desenvolver um programa em Pascal do que em Assembler.




O Pascal apresenta, portanto, um equilíbrio bem satisfatório entre velocidade de desenvolvimento e de execução. Somente o C, uma linguagem semelhante a ele em estrutura e filosofia, apresenta maior eficiência.

## PROJETO ALGORÍTMICO

Como vimos, a programação em Pascal requer uma fase inicial de projeto, antes de se chegar à máquina. Nessa fase, utilizam-se certas ferramentas de planejamento que ainda não discutimos em *INPUT*, entre as quais o fluxograma. Este deve ser usado com cautela, pois, em si, não é estruturado. Uma ferramenta mais conveniente para o projeto de programas é o *algoritmo*: ùm detalhamento passo a passo do método a ser empregado na solução do problema.

Um exemplo familiar de um algoritmo é a receita culinária: ela contém toda a informação necessária para reproduzir um certo prato. Um programa finalizado tem, igualmente, toda a informação de que o computador precisa para resolver um problema ou realizar uma tarefa de um determinado modo.

Entretanto, o programador (ou o cozinheiro), quando começa a escrever um programa (ou a receita), não tem uma idéia bem definida sobre o procedimento a seguir para chegar ao objetivo desejado. Como exemplo, vamos analisar o processo de elaboração de uma receita de torta. Em uma primeira tentativa, poderíamos dividir a tarefa (ou algoritmo inicial) em quatro passos:

1. Preparar a massa
2. Preparar o recheio
3. Colocar tudo numa fôrma
4. Assar a torta

Ao fazer uma torta, talvez você não precise cumprir todos os passos (é pos-

sível comprar a massa pronta, por exemplo!). Suponhamos, porém, que o computador é um "cozinheiro amador", que deve conhecer cada detalhe.

A próxima etapa consiste em fazer a subdivisão dos passos em operações menores — processo denominado *refinamento gradativo*. Você encontrará esse conceito na metodologia de desenvolvimento de programas muito extensos, que não podem ser projetados de uma só vez. No nosso exemplo, refinando o passo 1, teríamos:

1.1 - Pesar a farinha
1.2 - Pesar a manteiga
1.3 - Misturar a manteiga com a farinha
1.4 - Adicionar água
1.5 - Mexer

Poderíamos refinar ainda mais cada passo, até chegarmos a um nível de simplicidade tal que fosse possível a qualquer um fazer uma torta.

O principal, nesse processo, foi termos considerado previamente todos os passos da solução do problema, em vez de iniciarmos a tarefa em um ponto qualquer, ao acaso. Para a programação em Pascal, o método a ser seguido é muito similar. No estágio inicial, escreve-se cada passo em português mesmo. Refinando-os sucessivamente, chega-se à formalização proporcionada por uma declaração ou instrução em Pascal. Separados do algoritmo inicial, os passos podem ser representados por procedimentos ou funções. Estes são posteriormente reunidos em um programa só. Mas como escrever um programa em Pascal sem conhecer os comandos dessa linguagem? Examinemos alguns exemplos.

## UM PROGRAMA EM PASCAL

Os programas em Pascal costumam parecer complicados demais em relação à simplicidade da tarefa a cumprir. Veja, por exemplo, este programa, que soma dois números e exibe o resultado:

```
program exemplo (input,output);
var n1,n2,resultado:integer;
begin
   read (n1,n2);
   resultado:=n1+n2;
   writeln (resultado)
end.
```

Como se pode notar, o programa é mais longo que um equivalente em BASIC, embora de execução mais rápida. Quanto maior a complexidade da tarefa, porém, maiores a economia e concisão oferecidas por um programa em Pascal.

O programa anterior poderia resultar de um algoritmo inicial como:

1. Estabeleça as condições iniciais
2. Inicie o procedimento
3. Entre os números
4. Acrescente os números
5. Imprima o resultado
6. Termine o procedimento

O programa segue rigorosamente o algoritmo. Observe que é necessário declarar explicitamente os passos 1, 2 e 6 — ou seja, precisamos especificar o que é trabalhado pelo programa, quando iniciá-lo e quando encerrá-lo.

Este é um dos fatores que explicam a maior extensão de um programa escrito em Pascal em relação a um equivalente em BASIC. Vamos agora analisar o programa linha por linha:

```
program exemplo (input,output);
```

significa que o programa terá entradas (**input**) e saídas (**output**) e se chamará **exemplo**. Se ele não precisasse ler dados externos, bastaria escrever:

```
program exemplo (output);
```

O ponto e vírgula no final de uma linha indica o fim de uma declaração.

As características de todas as variáveis usadas por um programa em Pascal devem ser especificadas. A linha:

```
var n1,n2,resultado:integer;
```

informa que usaremos três variáveis inteiras (**integer**): **n1**, **n2** e **resultado**.

Feitas as declarações, assinalamos o início do procedimento propriamente dito, com a palavra begin. Note agora que o texto do programa é recuado em relação à linha anterior. Este é um recurso muito usado em programas estruturados, pois indica para quem lê quais são as declarações dentro de um procedimento ou laço de repetição.

A leitura dos dados pelo teclado é muito semelhante à do BASIC: read (n1,n2). Só podemos usar aqui esses nomes de variáveis porque elas foram declaradas antes, em var.

A declaração aritmética também é semelhante a uma instrução em BASIC:

```
resultado:=n1+n2;
```

O símbolo de atribuição := foi assim definido para evitar a confusão causada pelo = do FORTRAN e do BASIC, que dá a impressão errônea de se tratar de uma fórmula matemática.

Finalmente, escrevemos o resultado:

```
writeln (resultado)
```

que tem um efeito semelhante ao **PRINT RESULTADO** de um programa em BASIC. Não há ponto e vírgula no final da linha, pois a declaração seguinte é um **end**. Alguns compiladores permitem o uso desse sinal desde que se coloque uma linha em branco depois.

O Pascal conta, naturalmente, com vários outros comandos e funções, admitindo também a introdução de notas e comentários no programa, entre chaves: {e}. Com os microcomputadores que não têm essas teclas, pode-se utilizar uma combinação de parênteses com asteriscos (* e *).

## EDIÇÃO DO PROGRAMA

O processo de edição e teste de programas em linguagem compilada, como o Pascal, não é tão direto como com um interpretador. Normalmente, edita-se

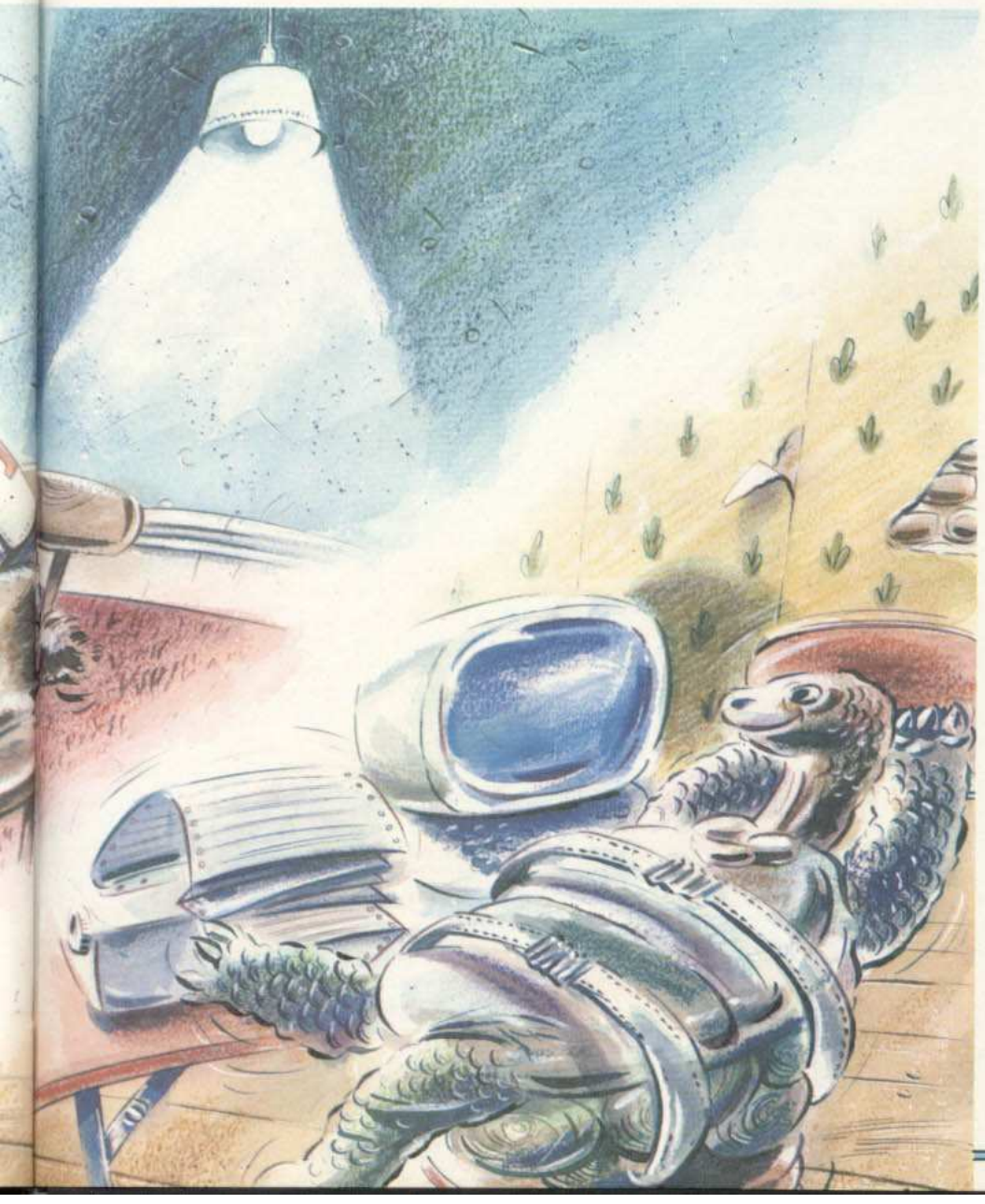
apenas o programa-fonte, com o auxílio de um editor de linhas (quase sempre disponível como um utilitário do sistema operacional) ou de um processador de textos. Depois, compila-se o programa e testa-se o resultado, executando-o. Caso se verifique algum erro, todo o processo deve ser repetido, a partir do programa-fonte.

Alguns compiladores Pascal, como o UCSD e o Turbo-Pascal, dispõem de um editor próprio, o que evita esse vaivém entre editor, sistema operacional e compilador. Nesse caso, tanto o programa-fonte quanto o programa-objeto podem residir na RAM enquanto durar o processo, agilizando-o muito — se o programa não for muito longo, e se houver boa capacidade de memória.

Como está em linguagem de máquina pura, sem comentários, rótulos, ou similares, o programa compilado oferece a vantagem final de inviabilizar a ação de "espiões". Assim, depois de editar um programa em Pascal, você poderá ter a certeza de que ninguém copiará os algoritmos de sua criação.

## PROGRAMAS DISPONIVEIS

Existem compiladores Pascal para a maioria das linhas de microcomputadores. Entretanto, como já mencionamos, eles variam muito entre si, quanto ao funcionamento e desempenho, e nem todos podem ser encontrados com facilidade no Brasil. Antes de adquirir um programa desse tipo, convém fazer uma pesquisa em diferentes "software houses" — de preferência nas especializadas em uma determinada linha de computadores ou nas dedicadas exclusivamente à programação Pascal.



O uso mais sério do Pascal requer componentes mais caros, exigindo, no mínimo, um acionador de disquetes. Por isso, máquinas como o Apple e a IBM-PC contam com as melhores versões.

O compilador Pascal para o Spectrum é exclusivo para microdrives e disquetes (não disponíveis no Brasil). Inclui um editor de linhas, um gerador de números aleatórios e suporte gráfico em alta resolução.

Existem diversas versões do Pascal para o TRS-Color. Suas características variam conforme o sistema operacional utilizado: Flex, RS-DOS (original) ou OS-9 (semelhante ao Unix). Todas funcionam somente em micros dotados de acionadores de disquetes.

Esse micro dispõe de duas versões do Pascal: um compilador que roda sob o sistema operacional TRSDOS e compatíveis, e outro, semelhante ao Turbo Pascal, que roda sob o sistema operacional CP/M. Ambas são bastante completas, mas não incluem suporte gráfico.

As duas versões mais utilizadas de Pascal para as máquinas compatíveis com a linha Apple são o UCSD e o Turbo-Pascal. O UCSD (abreviatura da universidade onde foi desenvolvida: *University of California at San Diego*) foi o primeiro e mais popular *sistema* Pascal (*p-System*). Trata-se, na realidade, de um sistema operacional completo, que substitui o DOS, e não apenas de um compilador.

Já o Turbo-Pascal roda sob o sistema operacional CP/M (*Control Program for Microcomputers*), que pode ser colocado em um Apple quando se troca a UCP por um microprocessador Z-80.

O Pascal para o MSX é do tipo Microsoft e destina-se a máquinas dotadas de acionador de disquetes de 5.25 ou 3.5 polegadas, rodando sob o sistema operacional MSX-DOS ou compatível.

# FORMATAÇÃO DE VALORES

- O PRINT USING
- VALORES MONETÁRIOS
- FORMATAÇÃO COM PARÊNTESES
- UMA ROTINA PODEROSA

**Se você tem problemas com a formatação de saída para valores monetários, poderá encontrar aqui uma boa solução: o uso de duas eficientes rotinas no lugar do comando PRINT USING.**

O comando **PRINT USING** é um dos recursos mais poderosos do BASIC de algumas linhas de computadores, como o TRS-80, o TRS-Color e o MSX. Com ele, podemos formatar de diferentes maneiras a saída de números e cadeias alfanuméricas. Porém, esse comando apresenta inconvenientes, sobretudo quanto à formatação de valores monetários.

No Brasil, usam-se variáveis de precisão dupla (quinze dígitos) para representar valores monetários, pois a precisão simples não nos permite lidar com valores superiores a 10 milhões de cruzados. E o **PRINT USING**, infelizmente, é muito lento quando se trata de formatar variáveis de precisão dupla.

Além disso, não é possível utilizar o **PRINT USING** dos computadores mencionados para formatar valores monetários segundo a convenção brasileira, que impõe a separação dos milhares por pontos e dos centavos por vírgulas.

As funções que explicamos neste artigo oferecem várias alternativas para a formatação de valores, e são muito mais rápidas (três a seis vezes) do que o **PRINT USING** normal.

## FORMATAÇÃO COM PARÊNTESES

TT MSX

Esta rotina permite a formatação de valores monetários segundo as convenções dos balanços contábeis: com os números negativos entre parênteses:

```
10 DEF FNN$(V#,E%)=RIGHT$(STRIN
G$(E%," ")+LEFT$("(",ABS(V#<0))
+LEFT$(" ",ABS(V#>=0))+MID$(STR
$(V#),2),E%)+LEFT$(")",ABS(V#<0
))+LEFT$(" ",ABS(V#=>0))
100 INPUT "Numero de colunas ";
E%
110 INPUT "Valor ";V#
120 PRINT FNN$(V#,E%):GOTO 110
```

A função **FNF$**, estabelecida na linha 10, tem dois argumentos: **V#**, um valor monetário em precisão dupla (pode incluir centavos ou não), e **E%**, um valor inteiro que especifica o número de colunas a usar na formatação (com justificação à direita). As linhas 100 e 120 formam um laço simples de teste.

O BASIC das máquinas mencionadas é capaz de definir em apenas uma linha expressões de grande complexidade — inclusive expressões condicionais que imitam a operação de um **IF...THEN**. Observe, por exemplo, a expressão:

```
LEFT$("(",ABS(V# <0))
```

Pode parecer estranho utilizar um operador de comparação (sinal de menor) dentro de uma expressão matemática. Mas podemos compreender sua função se levarmos em conta que o resultado numérico de uma expressão lógica é igual a −1 (ou 1, em alguns computadores), se ela for verdadeira, e a 0, se for falsa. Assim, se **V#** for menor que 0, a expressão anterior resultará em um *string* contendo um parêntese à esquerda — LEFT$("(",1). A linha 10 concatenará esse parêntese (se o número for negativo) ou um espaço em branco (se o número for positivo) à cadeia de saída. A expressão **LEFT$ (" ", ABS (V# > 0))** se encarrega da operação, que depois é repetida para o parêntese direito.

## DINHEIRO À BRASILEIRA

Examinaremos agora uma função capaz de formatar valores usando pontos e vírgulas, segundo a convenção bancária brasileira. Símbolos especiais representarão números positivos (créditos) e números negativos (débitos).

Como essa função é mais complexa, definiremos uma sub-rotina:

```
110 INPUT "Numero de colunas ";
E%
120 INPUT "Caractere de preench
imento ";B$
130 INPUT "Simbolo p/valor posi
tivo ";P$
140 INPUT "Simbolo p/valor nega
tivo ";N$
150 INPUT "Valor ";V#
160 GOSUB 1000:PRINT S$:GOTO 15
0
1000 I$=MID$(STR$(FIX(V#)),2):F
$=MID$(STR$(INT(100*(V#-FIX(V#)
))),2):S$=""
1010 IF LEN(I$)<=3 AND VAL(F$)>
0 THEN S$=I$+S$+","+F$:GOTO 103
0
1015 IF LEN(I$)<=3 AND VAL(F$)=
0 THEN S$=I$+S$:GOTO 1030
1020 S$="."+RIGHT$(I$,3)+S$:I$=
LEFT$(I$,LEN(I$)-3):GOTO 1010
1030 IF V#<0 THEN X$=N$ ELSE X$
=P$
1040 S$=RIGHT$(STRING$(E%,B$)+S
$+X$,E%):RETURN
```

Os argumentos de entrada são: **V#**, valor monetário em precisão dupla (incluindo centavos ou não, após o ponto); **E%**, número de colunas para formatação; **B$**, caractere de preenchimento à esquerda (asteriscos, zeros ou um espaço em branco); **P$** e **N$**, símbolos à direita do valor, para indicar créditos e débitos, respectivamente. O string de saída retorna em **S$**. As linhas 110 a 160 testam a sub-rotina.

Na linha 1000, **V#** é quebrado em dois valores inteiros, correspondentes aos cruzados e aos centavos. Esses valores são armazenados em duas cadeias, **I$** e **F$**, respectivamente.

As linhas 1010 e 1015 verificam se **I$** tem três ou menos dígitos (um valor igual ou menor que 999 cruzados), e se existe um valor em centavos **(VAL(F$)>0)**. Conforme o resultado dos dois testes, a rotina termina, saltando para a linha 1030, que concatena o símbolo adequado para créditos e débitos ao final de **S$**. A linha 1040 enquadra o string **S$** de saída dentro de **E%** espaços. O laço formado pela linha 1020 reduz progressivamente o tamanho de **I$**, separando cada grupo de três dígitos por um ponto, até que **LEN(I$)** seja menor ou igual a 3.

Entre as aplicações dessa rotina, inclui-se o preenchimento de cheques — os caracteres de proteção de campo são colocados à esquerda do valor. Se quiser adicionar um cifrão (ou o símbolo Cz$) entre os caracteres de proteção e o valor numérico, na linha 1000 da rotina, que inicializa S$, substitua a expressão **S$ = ""** por **S$ = "Cz$"**.

## 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

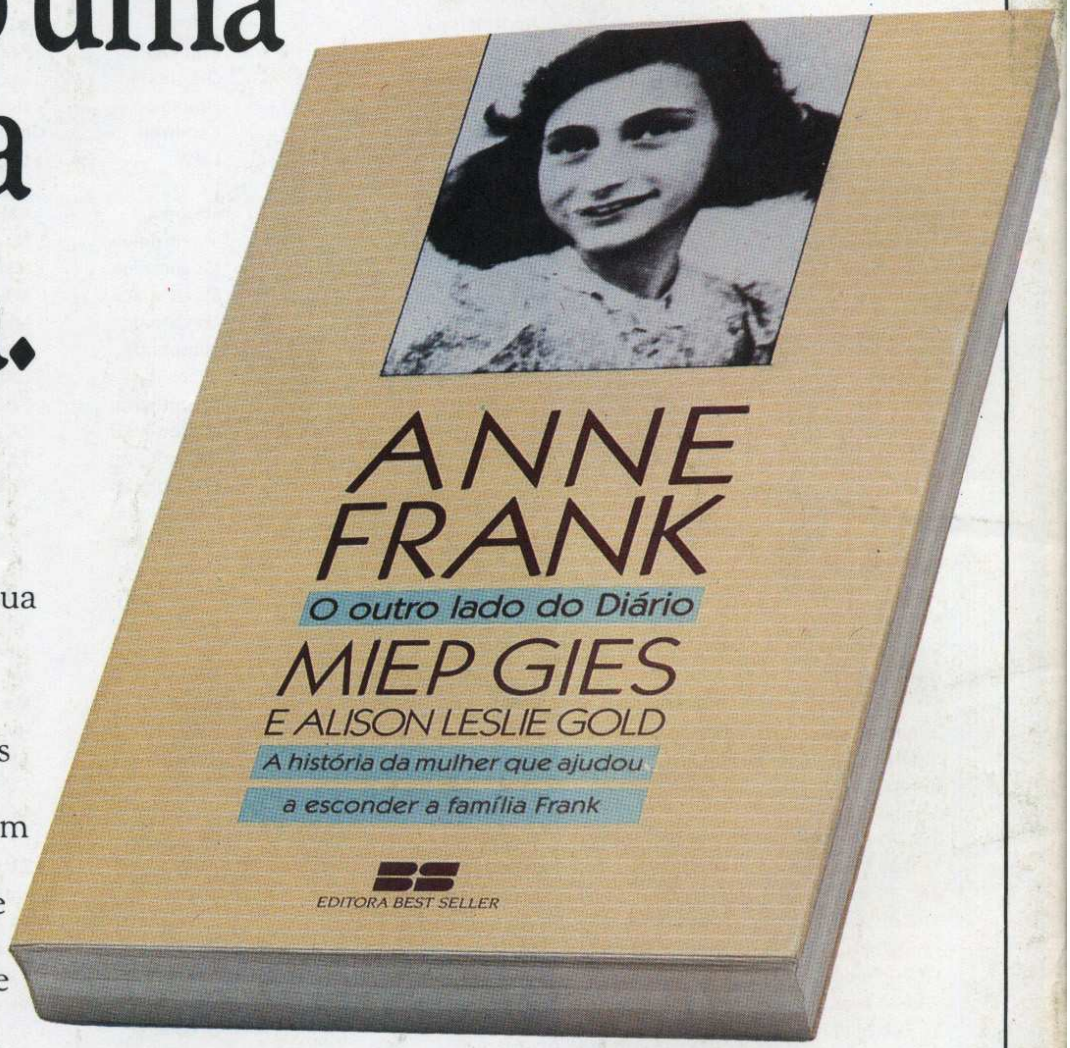

Spectrum


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.

# O Diário de Anne Frank emocionou milhões. E era só uma parte da história.




Anne Frank é lembrada como símbolo do extermínio de milhões de judeus. Seu drama e o de sua família são vistos agora de um ângulo diferente, o da mulher holandesa que ajudou os Frank durante os dois anos que passaram escondidos dos nazistas num sótão de Amsterdã, e que se tornou o único elo entre eles e o mundo exterior. É com tocante simplicidade que esta história triste — o "outro lado" do *Diário* de Anne Frank — é contada neste livro, poderoso testemunho da coragem de que é capaz o ser humano, mesmo nos momentos sombrios da História.

*Magnificamente escrito por uma pessoa devotada ao ser humano. Seu estilo simples cativa o leitor. Um livro recomendável aos que se preocupam com os rumos do mundo."*

— Isaac Bashevis Singer, Prêmio Nobel de Literatura de 1978

EDITORA BEST SELLER

JÁ NAS LIVRARIAS