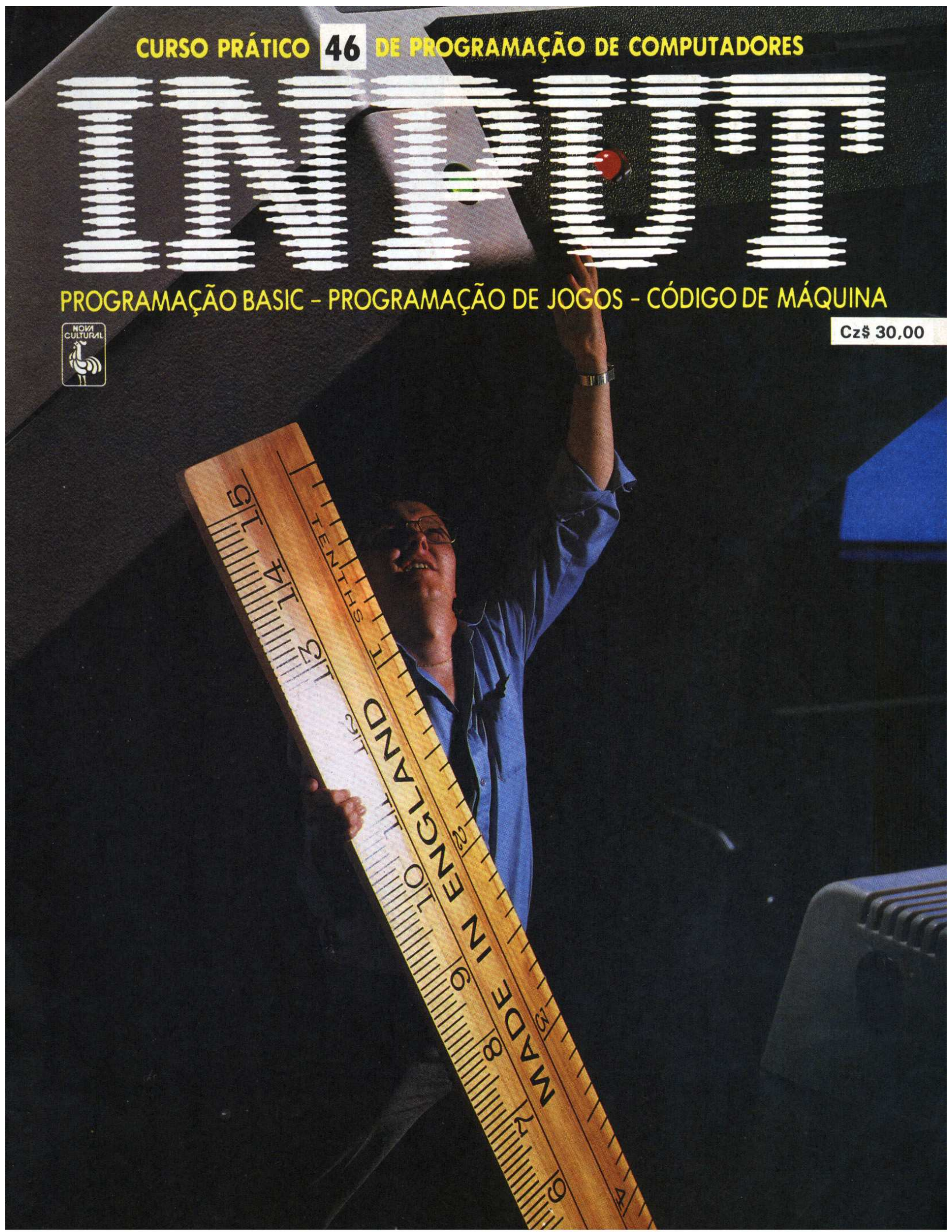
CURSO PRÁTICO 46 DE PROGRAMAÇÃO DE COMPUTADORES

INFO·BIT

PROGRAMAÇÃO BASIC - PROGRAMAÇÃO DE JOGOS - CÓDIGO DE MÁQUINA

NOVA CULTURAL

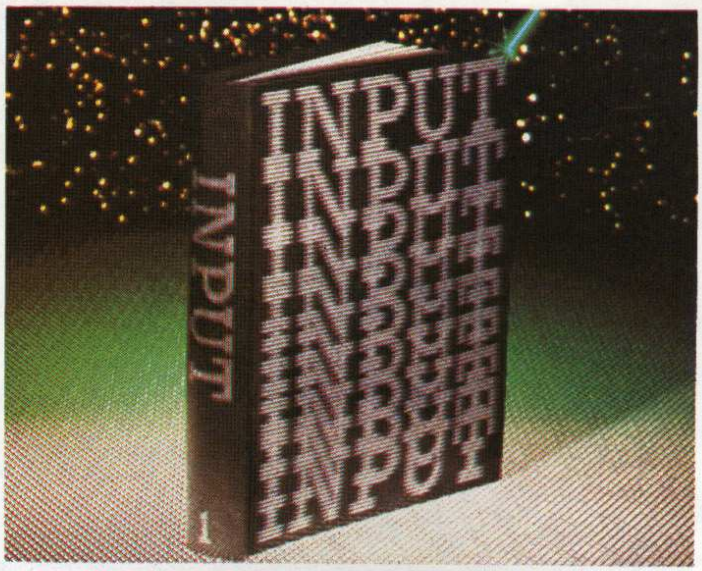
Cz$ 30,00



INPUT

Vol. 4 N.º 46

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
Diretor Editorial: Carmo Chagas

Editores Executivos: Antonio José Filho, Berta Sztark Amar

Editor Chefe: Paulo de Almeida
Editor de Texto: Cláudio A. Y. Cavalcanti
Chefe de Arte: Carlos Luiz Batista
Assistentes de Arte: Dagmar Bastos Sampaio, Grace Alonso Arruda, 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

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

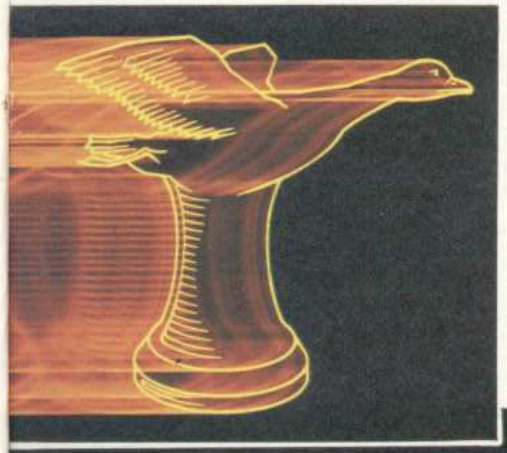
PRODUÇÃO
Gerente de Produção: João Stungis
Coordenador de Impressão: Atílio Roberto Bonon

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 Britto
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.

O JOGO A RAPOSA E OS GANSOS (2)

Examinamos, no artigo anterior, os princípios que fundamentam o jogo A Raposa e os Gansos. Começaremos agora a montar o programa, digitando suas primeiras rotinas — entre elas, a que mapeia os movimentos.



Apresentamos aqui as rotinas de inicialização e de mapeamento dos movimentos. Além delas, você deverá digitar a rotina que oferece ao jogador mais uma partida. Nesse ponto, porém, a execução do programa não resultará em nada, pois ainda faltam rotinas muito importantes. Todas elas serão dadas no próximo artigo da série.

UMA VISÃO GERAL

O programa avalia as posições ocupadas no tabuleiro segundo a configuração das peças. Como cada posição tem um valor, pode-se, ao analisar as jogadas à frente, escolher o melhor movimento pelo resultado numérico maior.

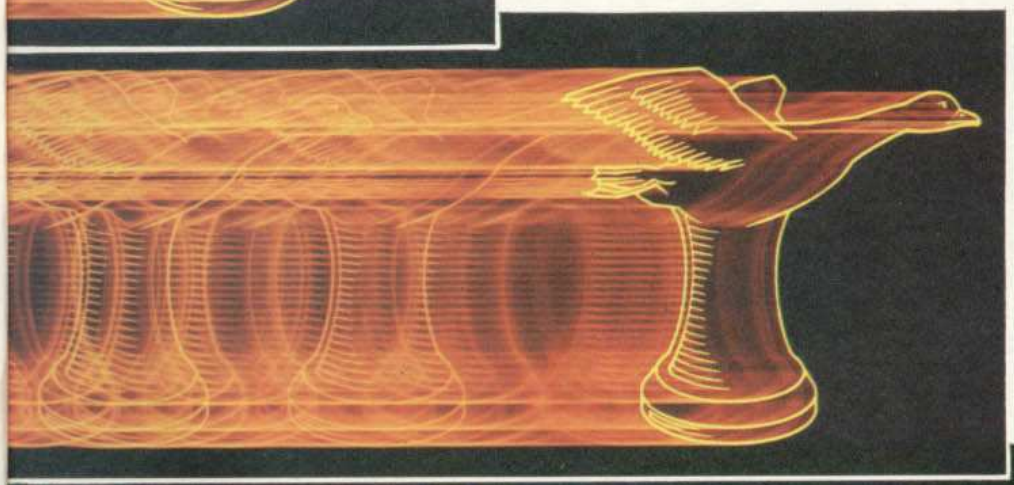
O programa trabalha de três maneiras quando está analisando jogadas. A mais simples (nível 1) consiste no exame de apenas uma jogada. Nos níveis mais elevados, o programa examina as múltiplas possibilidades de jogo, utilizando o algoritmo alfa-beta para economizar tempo. Nos níveis intermediários, analisa todas as possibilidades abertas no momento.

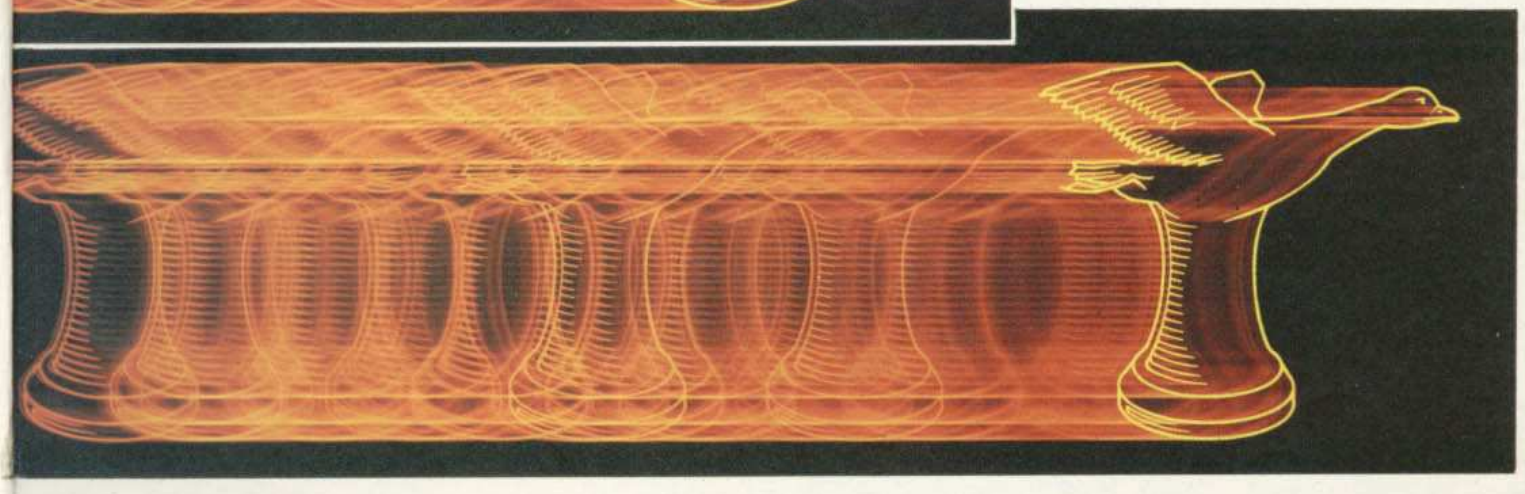
As rotinas situadas da linha 2010 à linha 3000, que são executadas apenas uma vez, foram colocadas no fim do programa. No começo estão as rotinas mais importantes, o que garante uma maior velocidade ao jogo.

INICIALIZAÇÃO

As rotinas apresentadas a seguir, usadas para inicializar o jogo, dimensionam as matrizes, definem as funções e, no caso de alguns micros, estabelecem também o tabuleiro.

```
2010 DIM G(4): DEF FN U(A)=INT
(A-4*INT (A/4)): DEF FN V(A)=IN
T (A-8*INT (A/8))>=4: DEF FN W(
A)=INT (A-2*INT (A/2))
2015 LET HF=0: LET HG=0
2020 DIM B(32): LET B(1)=1: FOR
 I=1 TO 31: LET B(I+1)=B(I)*2:
NEXT I
2026 LET BX=B(32)*2-B(25): LET
E=1E30: LET H=-1E30
2030 LET L2=LN (2)
2040 DIM B$(16,2,16): DIM G$(2,
4): LET G$(1)="  ": LET G$(2)="
  "+CHR$ 146+CHR$ 147: DIM H$(2
,4): LET H$(1)="  ": LET H$(2)=
```





```
CHR$ 146+CHR$ 147+"  "
2050 LET X=1: FOR A=1 TO 2: FOR
 B=1 TO 2: FOR C=1 TO 2: FOR D=
1 TO 2: LET B$(X,1)=G$(D)+G$(C)
+G$(B)+G$(A)
2060 LET B$(X,2)=H$(A)+H$(B)+H$
(C)+H$(D): LET X=X+1: NEXT D: N
EXT C: NEXT B: NEXT A
2070 DIM S$(8,16): GOSUB 6000
2090 DIM F$(2,4): LET F$(1)="
"+CHR$ 144+CHR$ 145: LET F$(2)=
CHR$ 144+CHR$ 145+"  "
2095 DEF FN C(B)=FN U(B-1)*(4-8
*FN V(B-1))+12*FN V(B-1)
```

```
10 GOTO 2010
2010 DIM G(4),B(31),M(3,31),X(3
1),Z(31)
2020 B(0)=1:FOR K=1 TO 31:B(K)=
B(K-1)*2:NEXT
2026 BX=B(31)*2-B(24):E=1E30:H=
-1E30
2030 L2=LOG(2):DEFFNA(F)=INT(LO
G(F)/L2+.001)
```

```
7 KEYOFF:GOTO 2010
10 GOSUB 4000:GOTO 1010
2010 DIM G(4),B(31),M(3,31),X(3
1),Z(31),GS(4,1)
2020 B(0)=1:FOR K=1 TO 31:B(K)=
B(K-1)*2:NEXT
2026 BX=B(31)*2-B(24):E=1E+30:H
=-1E+30
2030 L2=LOG(2):DEFFNA(F)=INT(LO
G(F)/L2+1E-03)
2040 OPEN "GRP:" FOR OUTPUT AS
#1
```

```
10  GOTO 2010

2010  DIM G(4),B(31),M(3,31),X(
31),Z(31):V$ =  CHR$ (7) +  CHR
$ (1) +  CHR$ (1) +  CHR$ (7)
2020 B(0) = 1: FOR K = 1 TO 31:
B(K) = B(K - 1) * 2: NEXT
2026 BX = B(31) * 2 - B(24):E =
 1E30:H =  - 1E30
2030 L2 =  LOG (2): DEF  FN A(F
) =  INT ( LOG (F) / L2 + .001)
```

A linha 2010 dimensiona a matriz utilizada para armazenar as posições dos gansos. A linha 2020 numera cada quadrado do tabuleiro.

A linha 2030 determina o número total de configurações que o programa pode avaliar. O valor 0.001 foi adicionado à definição da função A para evitar a ocorrência de erros de arredondamento no cálculo de logaritmos.

No Spectrum, a matriz B$, dimensionada na linha 2040, mostra as colunas do tabuleiro já ocupadas pelas peças. F$ é usada para mostrar a peça da raposa e a casa em que está, e H$, para os gansos e suas casas. A matriz S$, determina o número dos quadrados.

Nos demais microcomputadores, as figuras são definidas em alta resolução por uma outra rotina.

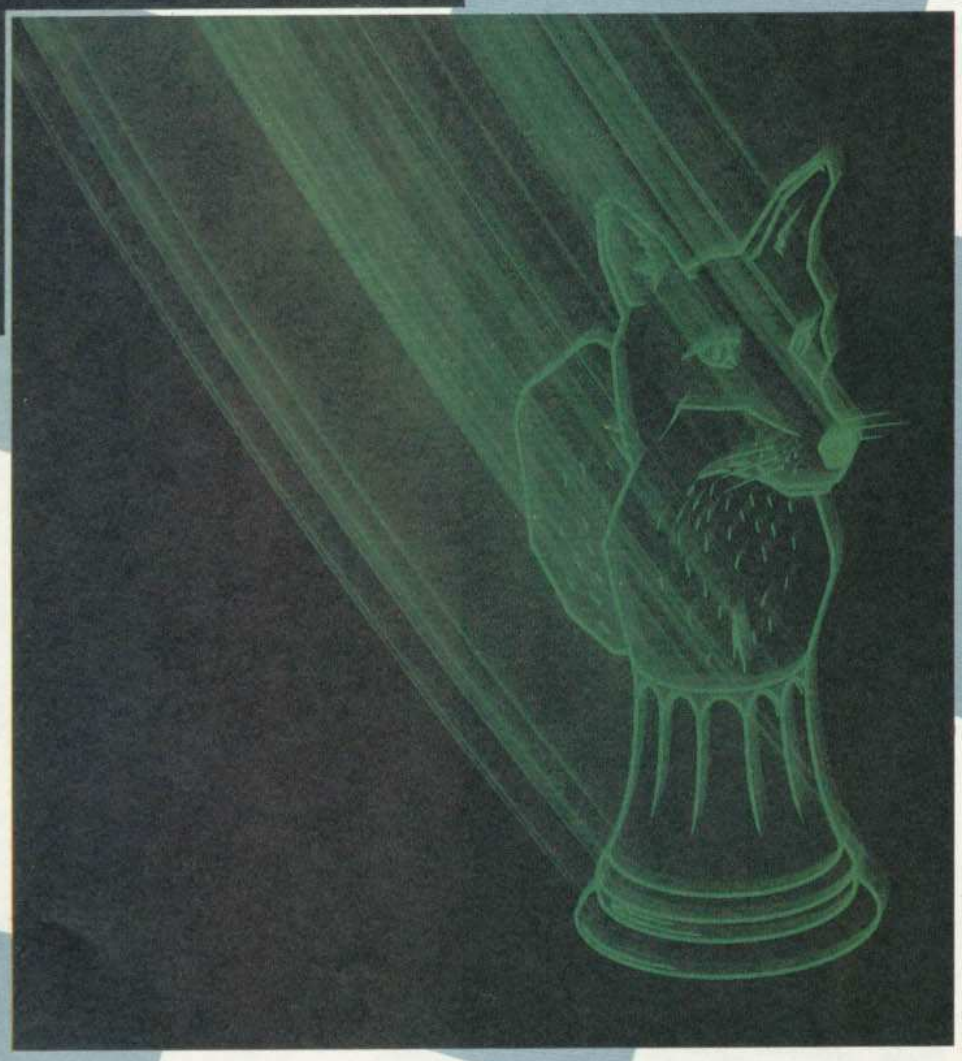
COMEÇANDO

Esta rotina permite a escolha das peças com que se vai jogar e o grau de dificuldade. Comece pelo mais fácil:

```
2700 LET F=2: LET G(1)=29: LET
G(2)=30: LET G(3)=31: LET G(4)=
32: GOSUB 2710: GOTO 1010
2710 CLS : PRINT AT 0,8; INK 1;
"RAPOSA E GANSOS": INPUT "VOCE
QUER ...";TAB 5;"SER A RAPOSA ?
(S/N)   ";I$
2720 LET PF=0: IF I$="S" OR Y$=
"s" THEN  GOTO 2760
2730 LET PF=1: IF I$<>"N" AND I
$<>"n" THEN  GOTO 2710
2740 INPUT "NIVEL DE HABILIDADE
 DA RAPOSA ? ";SF: IF SF<1 OR S
F>10 THEN  GOTO 2740
2750 LET HF=131*(SF=5)+613*(SF=
6)+1997*(SF>6)
2760 INPUT "VOCE QUER ...";TAB
5;"CONTROLAR OS GANSOS ?(S/N) "
;I$
2770 LET PG=0: IF I$="S" OR I$=
"s" THEN  GOTO 2860
2780 LET PG=1: IF I$<>"N" AND I
$<>"n" THEN  GOTO 2760
2790 INPUT "NIVEL DA HABILIDADE
 DOS GANSOS  ";SG: IF SG<1 OR S
G>10 THEN  GOTO 2790
2800 LET HG=131*(SG=5)+613*(SG=
6)+1997*(SG>6): IF HF<HG THEN
LET HF=HG
2860 INPUT "VOCE QUER ALTERAR A
S POSICOES   INICIAIS ?";I$: IF
 I$="N" OR I$="n" THEN  GOTO 30
00
2880 IF I$<>"S" AND I$<>"s" THE
N  GOTO 2860
2890 GOSUB 210: GOSUB 310: INPU
T "QUER MOVER A RAPOSA ?";I$
2900 IF I$="N" OR I$="n" THEN
GOTO 2930
2910 IF I$<>"S" AND I$<>"s" THE
N  GOTO 2890
2920 INPUT "PARA ONDE ?";F: IF
F<1 OR F>32 THEN  GOTO 2920
2930 FOR G=1 TO 4: GOSUB 210: G
OSUB 310
2940 INPUT "QUER MOVER O GANSO
DA POSICAO";(G(G));"?";I$
2950 IF I$="N" OR I$="n" THEN
GOTO 2990
2960 IF I$<>"S" AND I$<>"s" THE
N  GOTO 2940
2970 INPUT "PARA ONDE ?";I: IF
FN X(I) OR I=F THEN  GOTO 2960
2972 IF I<1 OR I>32 THEN  GOTO
2970
2980 LET G(G)=I
2990 NEXT G: IF FN X(F) THEN  P
RINT "HA UM GANSO SOB A RAPOSA"
: FOR I=1 TO 1500: NEXT I: GOTO
 2910
3000 RETURN
```

```
2500 DIM R(1400),S(1400)
2700 F=1:G(1)=28:G(2)=29:G(3)=3
0:G(4)=31:GOSUB 2710:GOTO 1010
2710 CLS:PRINT "VOCE QUER SER A
 RAPOSA (S/N) ?";
2720 K$=INKEY$:IF K$<>"S" AND K
$<>"N" THEN 2720
2730 PRINT K$:PF=1:IF K$="S" TH
EN PF=0:GOTO 2760
2740 PRINT:PRINT "NIVEL DE HABI
LIDADE DA RAPOSA   (0-9) ?";
2745 K$=INKEY$:IF K$<"0" OR K$>
"9" THEN 2745
```

```
2746 SF=VAL(K$)+1:PRINT K$
2750 HF=-131*(SF=5)-613*(SF=6)-
1399*(SF>6)
2760 PRINT:PRINT "VOCE QUER CON
TROLAR OS GANSOS   (S/N) ?";
2770 K$=INKEY$:IF K$<>"S" AND K
$<>"N" THEN 2770
2780 PRINT K$:PG=1:IF K$="S" TH
EN PG=0:GOTO 2860
2790 PRINT:PRINT"NIVEL DE HABIL
IDADE DOS GANSOS  (0-9) ?";
2795 K$=INKEY$:IF K$<"0" OR K$>
"9" THEN 2795
2796 SG=VAL(K$)+1:PRINT K$
2800 HG=-131*(SG=5)-613*(SG=6)-
1399*(SG>6):IF HF<HG THEN HF=HG
2860 PRINT:PRINT"VOCE QUER ALTE
RAR AS POSICOES   INICIAIS (S/N
)?";
2870 K$=INKEY$:IF K$<>"S" AND K
$<>"N" THEN 2870
2880 IF K$="N" THEN 3000
2890 GOSUB 210
2920 DRAW"BM180,80"+MW$:XX=FNXX
(1):YY=FNYY(1):GOSUB 1810:F=4*I
NT(YY/20):F=FNCN(F)
2925 PUT(68,8)-(87,27),SQ,PSET:
PUT(XX,YY+5)-(XX+19,YY+13),FX,P
SET
2930 FOR G=1 TO 4:GOSUB 210
2940 XX=FNXX(G(G)):X1=XX:YY=FNY
Y(G(G)):Y1=YY:GOSUB 1810:PUT(X1
,Y1)-(X1+19,Y1+19),SQ,PSET
2950 I=4*INT(YY/20):I=FNCN(I)
2960 IF (FNX(I) OR I=F) AND I<>
G(G) GOSUB 5000:GOTO 2940
2970 PUT(XX,YY+5)-(XX+19,YY+14)
,GS,PSET:G(G)=I
2990 NEXT:IF FNX(F) GOSUB 5000:
GOTO 2920
2995 C=1:G=G(1)
3000 RETURN
```



```
2500 DIM R(1100),S(1100)
2700 F=1:G(1)=28:G(2)=29:G(3)=3
0:G(4)=31:GOSUB 2710:GOTO 10
2710 CLS:PRINT"Alguém joga com
a raposa? (S/N) ";
2720 K$=INKEY$:IF K$<>"S" AND K
$<>"N" THEN 2720
2730 PRINTK$:PF=1:IF K$="S" THE
N PF=0:GOTO 2760
2740 PRINT:PRINT"Habilidade da
raposa? (0-9) ";
2745 K$=INKEY$:IF K$<"0" OR K$>
"9" THEN 2745
2746 SF=VAL(K$)+1:PRINTK$
2750 HF=-131*(SF=5)-613*(SF=6)-
1099*(SF>6)
2760 PRINT:PRINT"Alguém joga co
m os gansos? (S/N) ";
2770 K$=INKEY$:IF K$<>"S" AND K
$<>"N" THEN 2770
2780 PRINTK$:PG=1:IF K$="S" THE
N PG=0:GOTO 2860
2790 PRINT:PRINT"Habilidade dos
 gansos? (0-9) ";
2795 K$=INKEY$:IF K$<"0" OR K$>
"9" THEN 2795
2796 SG=VAL(K$)+1:PRINTK$
2800 HG=-131*(SG=5)-613*(SG=6)-
1099*(SG>6):IF HF<HG THEN HF=HG
2860 PRINT:PRINT"Você quer alte
rar as posições inicais?  (S/N)
";
2870 K$=INKEY$:IF K$<>"S" AND K
$<>"N" THEN 2870
2880 IF K$="N" THEN 3000
2890 GOSUB 210:GOSUB 4000:GOSUB
2920:GOTO 1010
2920 PRESET (188,80):PRINT#1,"P
ara ?":XX=FNXX(1):YY=FNYY(1):GO
SUB 1810:F=4*INT((YY-2)/20):F=F
NCN(F)
2925 PUT SPRITE FX,(XX,YY),6
2930 FOR G=1 TO 4:GOSUB 210
2940 XX=FNXX(G(G)):YY=FNYY(G(G)
):GOSUB 1810
2950 I=4*INT((YY-2)/20):I=FNCN(
I)
2960 IF (FNX(I) OR I=F) AND I<>
G(G) THEN GOSUB 5000:GOTO 2940
2970 GS(5-G,1)=I:G(G)=I:PUT SPR
ITE GS(5-G,0),(XX,YY),15
2990 NEXT:IF FNX(F) THEN GOSUB
5000:GOTO 2920
2995 C=1:G=G(1)
3000 RETURN
```

```
2500  DIM R(1500),S(1500)
2700 F = 1: FOR I = 28 TO 31:G(
I - 27) = I: NEXT : GOSUB 2710:
 GOTO 1010
2710  HOME : VTAB 22: PRINT "AL
GUEM JOGA COM A RAPOSA? (S/N) "
;
2720  GET K$: IF K$ <  > "S" AN
D K$ <  > "N" THEN 2720
2730  PRINT K$:PF = 1: IF K$ =
"S" THEN PF = 0: GOTO 2760
2740  PRINT "HABILIDADE DA RAPO
SA? (0-9) ";
2745  GET K$: IF K$ < "0" OR K$
 > "9" THEN 2745
2746 SF =  VAL (K$) + 1: PRINT
K$
2750 HF = 131 * (SF = 5) + 613
* (SF = 6) + 1499 * (SF > 6)
2760  HOME : VTAB 22: PRINT "AL
GUEM JOGA COM OS GANSOS? (S/N)
";
2770  GET K$: IF K$ <  > "S" AN
D K$ <  > "N" THEN 2770
2780  PRINT K$:PG = 1: IF K$ =
"S" THEN PG = 0: GOTO 2860
2790  PRINT "HABILIDADE DOS GAN
SOS? (0-9) ";
2795  GET K$: IF K$ < "0" OR K$
 > "9" THEN 2795
2796 SG =  VAL (K$) + 1: PRINT
K$
2800 HG = 131 * (SG = 5) + 613
* (SG = 6) + 1499 * (SG > 6): I
F HF < HG THEN HF = HG
2860  HOME : VTAB 22: PRINT "QU
ER MUDAR AS POSICOES INICIAIS?
(S/N) ";
2870  GET K$: IF K$ <  > "S" AN
D K$ <  > "N" THEN 2870
2880  IF K$ = "N" THEN 3000
2890  GOSUB 210
2920  HOME : VTAB 22: PRINT "MO
VE PARA ONDE?":XX =  FN XX(1):Y
Y =  FN YY(1): GOSUB 1810:F = 4
 *  INT (YY / 20):F =  FN CN(F)

2925  HCOLOR= 0: DRAW FX AT 120
,8: HCOLOR= 3: DRAW FX AT XX +
2,YY + 8
2930  FOR G = 1 TO 4: GOSUB 210

2940 :XX =  FN XX(G(G)):XN = XX
:YY =  FN YY(G(G)):YN = YY: GOS
UB 1810: HCOLOR= 0: DRAW GS AT
XN + 7,YN + 5
2950 I = 4 *  INT (YY / 20):I =
  FN CN(I)
2960  IF ( FN X(I) OR I = F) AN
D I <  > G(G) THEN  GOSUB 5000:
 GOTO 2920
2970  HCOLOR= 3: DRAW GS AT XX
+ 7,YY + 5:G(G) = I
2990  NEXT : IF  FN X(F) THEN
GOSUB 5000: GOTO 2920
2995 C = 1:G = G(1)
3000  RETURN
```

A linha 2700 determina as posições iniciais, com os quatro gansos ocupando os quatro quadrados da coluna inferior do tabuleiro, e a raposa, o segundo quadrado a partir da esquerda do vídeo, na coluna superior.

Em seguida, as linhas 2710 a 2750 pedem que se defina quem jogará pela raposa e que seja indicado um nível de dificuldade de 1 a 10, se for o computador. As linhas 2760 a 2800 são semelhantes, só que tratam dos gansos.

O jogador pode ajustar as posições iniciais não só para dar continuidade a uma partida (será preciso tomar nota das posições), mas, também, para estudar e tentar ganhar o jogo de uma posição particularmente interessante. As linhas 2860 a 3000 perguntam se o jogador quer alterar as posições de início, realiza todas as modificações necessárias e verifica se estas estão de acordo com as regras.

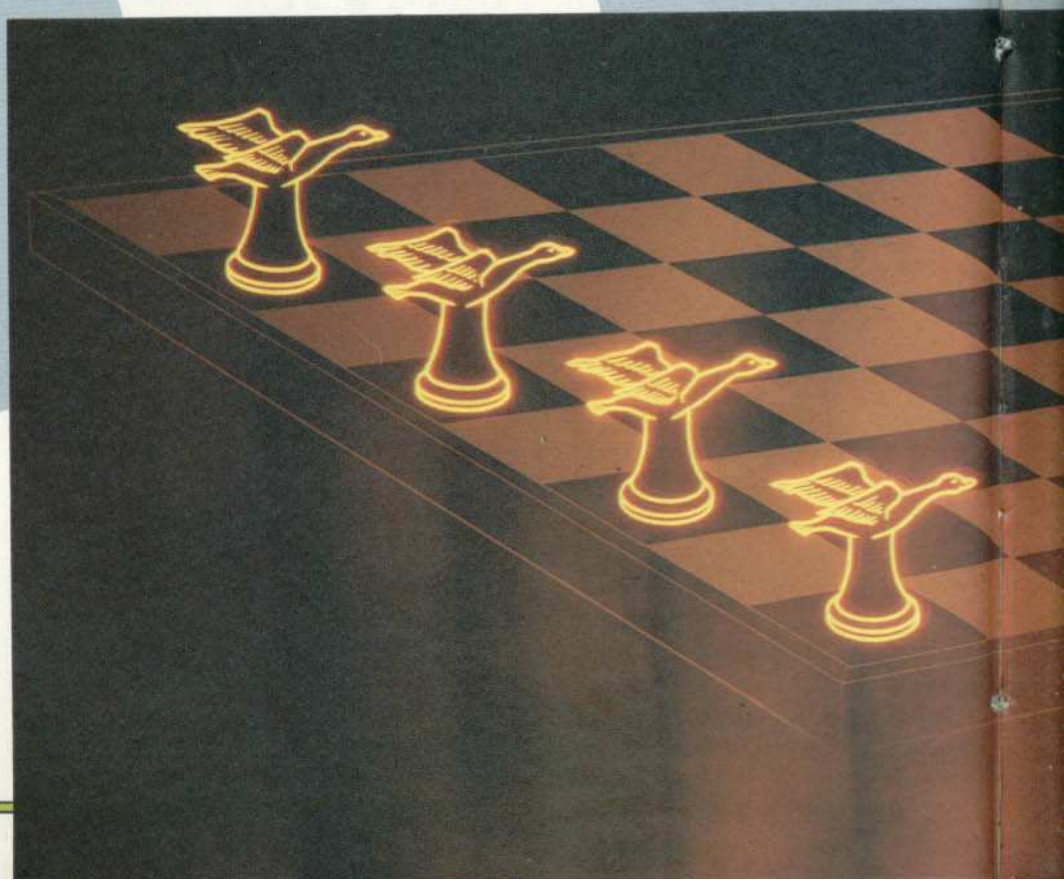
MAPEAMENTO DAS JOGADAS

A rotina de mapeamento de jogadas é uma das mais importantes do jogo:

```
 140 DEF FN X(B)=B=G(1) OR B=G(
 2) OR B=G(3) OR B=G(4)
2100 DIM R$(8,16)
2142 DEF FN Z(B)=(B=G(1))+(B=G(
2))*2+(B=G(3))*3+(B+G(4))*4
2150 DIM M(4,32): DIM X(32): DI
M Z(32)
2160 FOR B=1 TO 32: LET U=B-1-4
*INT (B/4-.2): FOR A=1 TO 4: LE
T M(A,B)=(B-2)-2*U+8*((B<5) OR
(A>2))+(A*7-6)*(U=3)+(A=2)+(A=2
)+(A=4): NEXT A: LET X(B)=((B>4
)+(B<29))*((U<3)+1): LET Z(B)=(
B>4)*((U<3)+1): NEXT B
2180 DIM V(11): DIM A(11): DIM
F(11): DIM P(11): DIM C(11): DI
M R(1): DIM S(1)
```

```
2110 DEFFNF(B)=((B>3)+(B<28))*(
((3ANDB)<3)-1)-1
2120 DEFFNG(B)=(B>3)*(((3 AND B
)<3)-1)-1
2140 DEFFNX(B)=(B=G(1)ORB=G(2)O
RB=G(3)ORB=G(4))
2142 DEFFNZ(B)=-(B=G(1))-(B=G(2
))*2-(B=G(3))*3-(B=G(4))*4
2150 DEFFNXX(B)=-((7ANDB)<4)*(2
8+40*(3ANDB))-((7ANDB)>3)*(128-
40*(3ANDB))
2155 DEFFNYY(B)=8+20*INT(B/4)
2156 DEFFNCN(B)=B-((7ANDB)<4)*(
XX-28)/40-((7ANDB)>3)*(128-XX)/
40
2160 FOR B=0 TO 31:FOR A=0 TO 3
:M(A,B)=B-2*(3ANDB)-2-8*(B<4 OR
 A>1)-(1+A*7)*((3 AND B)=3)+(1
AND A):NEXT:X(B)=FNF(B):Z(B)=FN
G(B):NEXT
```

```
2110 DEFFNF(B)=((B>3)+(B<28))*(
((BMOD4)<3)-1)-1
```



```
2120 DEFFNG(B)=(B>3)*(((BMOD4)<
3)-1)-1
2140 DEFFNX(B)=(B=G(1)ORB=G(2)O
RB=G(3)ORB=G(4))
2142 DEFFNZ(B)=-(B=G(1))-(B=G(2
))*2-(B=G(3))*3-(B=G(4))*4
2144 DEFFNZZ(B)=-(B=GS(1,1))-(B
=GS(2,1))*2-(B=GS(3,1))*3-(B=GS
(4,1))*4
2150 DEFFNXX(B)=-((BMOD8)<4)*(3
8+40*(BMOD4))-((BMOD8)>3)*(138-
40*(BMOD4))
2155 DEFFNYY(B)=20+20*INT(B/4)
2156 DEFFNCN(B)=B-((BMOD8)<4)*(
XX-38)/40-((BMOD8)>3)*(138-XX)/
40
2160 FOR B=0 TO 31:FOR A=0 TO 3
:M(A,B)=B-2*(BMOD4)-2-8*(B<4 OR
 A>1)-(1+A*7)*((BMOD4)=3)+(AMOD
2):NEXT:X(B)=FNF(B):Z(B)=FNG(B)
:NEXT
```

```
2110  DEF  FN MD(X) = X - ( INT
 (X / MD) * MD): DEF  FN M2(X)
= X - ( INT (X / 2) * 2): DEF
FN M4(X) = X - ( INT (X / 4) *
4): DEF  FN M8(X) = X - ( INT (
X / 8) * 8)
2115  DEF  FN F(B) = ((B > 3) +
 (B < 28)) * (( FN M4(B) < 3) +
 1) - 1
2120  DEF  FN G(B) = (B > 3) *
(( FN M4(B) < 3) + 1) - 1
2140  DEF  FN X(B) =  - (B = G(
1) OR B = G(2) OR B = G(3) OR B
 = G(4))
2142  DEF  FN Z(B) = (B = G(1))
 + (B = G(2)) * 2 + (B = G(3))
* 3 + (B = G(4)) * 4
2150  DEF  FN XX(B) = ( FN M8(B
) < 4) * (X1 + 21 + 40 *  FN M4
(B)) + ( FN M8(B) > 3) * (X2 -
41 - 40 *  FN M4(B))
2155  DEF  FN YY(B) = Y1 + 20 *
 INT (B / 4)
2156  DEF  FN CN(B) = B + ( FN
M8(B) < 4) * (XX - 78) / 40 + (
 FN M8(B) > 3) * (178 - XX) / 4
0
2160  FOR B = 0 TO 31: FOR A =
0 TO 3:M(A,B) = B - 2 *  FN M4(
B) - 2 + 8 * (B < 4 OR A > 1) +
 (1 + A * 7) * ( FN M4(B) = 3)
+  FN M2(A): NEXT :X(B) =  FN F
(B):Z(B) =  FN G(B): NEXT
```

As linhas 2110 a 2160 montam o mapa dos movimentos da raposa e dos gansos na matriz M, armazenam o número dos movimentos possíveis da raposa na matriz X e o dos gansos na matriz Z. As matrizes são formadas pelas funções definidas nas linhas 2110 a 2142.

A rotina do Spectrum é mais curta devido à lógica interna dessa máquina.

MAIS UMA VEZ?

Agora, adicione esta rotina:

```
1410 INPUT "QUER JOGAR DE NOVO
(S/N) ?";I$
1420 IF I$="S" OR I$="s" THEN
GOTO 2700
1430 IF I$<>"N" AND I$<>"n" THE
N  GOTO 1410
1440 STOP
```

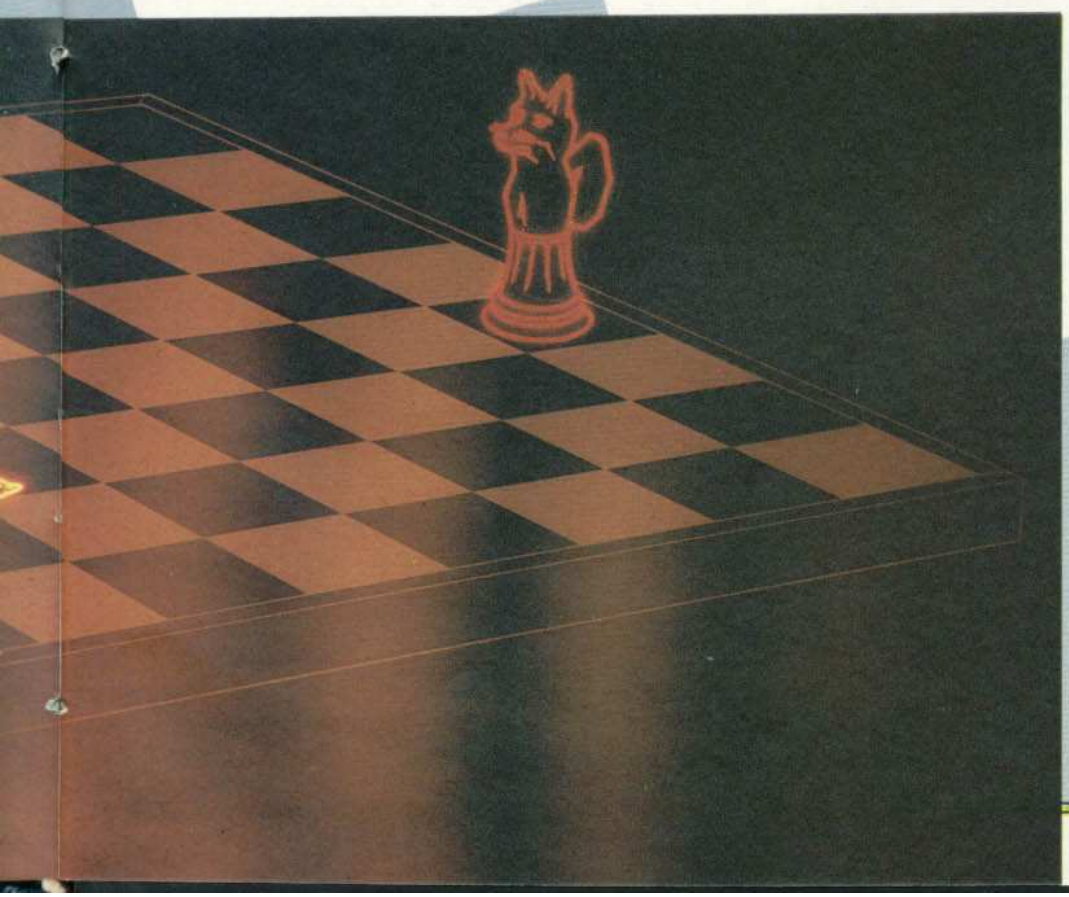
```
1410 PRINT @390,"QUER RECOMECAR
 (S/N) ?"
1420 K$=INKEY$:IF K$="S" GOSUB
4040:CLS:GOTO 2700
1430 IF K$<>"N" THEN 1420
1440 CLS:END
```

```
1410 LOCATE 5,20:PRINT"QUER REC
OMEÇAR? (S/N)".
1420 K$=INKEY$:IF K$="S" THEN R
UN
1430 IF K$<>"N" THEN 1420
1440 CLS:END
```

```
1410  FOR Z = 1 TO 5000: NEXT :
 TEXT : HOME : PRINT "JOGA OUTR
A VEZ? (S/N) ";
1420  GET K$: IF K$ <  > "S" AN
D K$ <  > "N" THEN 1420
1430  IF K$ = "S" THEN  RUN
1440  HOME : END
```

Essas linhas entrarão em cena quando os gansos conseguirem encurralar a raposa ou quando a raposa conseguir atingir o lado oposto do tabuleiro.

Não tente nesta fase executar o programa, pois ainda falta adicionar várias partes vitais. Com as rotinas do próximo artigo você poderá, finalmente, começar a jogar.



O que é Inteligência Artificial?

A Inteligência Artificial (IA), cujos conceitos são utilizados no jogo *A Raposa e os Gansos*, é um campo da Informática que procura desenvolver métodos computacionais para imitar o intelecto humano em várias tarefas, como o diagnóstico de doenças, a prospecção mineral etc. Os jogos estratégicos, como o xadrez, também têm sido alvo de estudos. Aplicam-se a eles sobretudo duas técnicas de IA: a *programação heurística* e a *otimização de busca*.

COMO UTILIZAR UM DISQUETE

O emprego de um acionador de disquetes em seu microcomputador pode envolver algumas complicações. Este artigo mostra como evitar erros comuns e como fazer o melhor uso desse periférico.

Como vimos no artigo da página 876, o disco magnético é a melhor alternativa para quem deseja um armazenamento mais rápido e confiável do que o proporcionado pela fita cassete. Entretanto, ao contrário do que ocorre com o gravador de fita, pode ser complicado fazer funcionar a combinação computador-disquete. Além disso, o usuário precisa aprender uma série de novos comandos para controlar o disco.

Este artigo explica como conectar e usar os acionadores de disquetes disponíveis para os diversos tipos de microcomputador existentes no Brasil. As linhas cobertas no artigo são: TRS-80, TRS-Color, Apple II, TK-2000 e MSX. Uma breve menção é feita aos micros compatíveis com a linha Spectrum (TK-90X), que dispõe de unidades acionadoras apenas no Exterior.

A primeira coisa que você deve fazer ao receber sua unidade de disquetes é checar se ela foi protegida para transporte. Em geral, um pedaço de cartolina do tamanho de um disquete é inserido na unidade, a fim de proteger a cabeça de leitura e gravação. Retire esse protetor somente depois de colocar o acionador no local em que será utilizado. Guarde-o, porém, para que, quando se fizer necessário, possa mudar o computador de lugar.

COMO CONECTAR

Conectar a unidade acionadora de disquetes ao micro pode ser fácil ou complicado, dependendo do modelo.

Os microcomputadores compatíveis com a linha TRS-Color utilizam unidades acionadoras de disquetes de 5 1/4 polegadas (minidisquetes), de face simples e densidade dupla. Encontram-se no mercado gabinetes de um ou de dois acionadores (como os do CP-400).

A conexão é muito simples: o cabo de ligação, do tipo plano, tem na ponta um conector que deve ser inserido na porta de cartuchos do micro.

O software operacional de utilização já está gravado em memória ROM, na própria unidade de disco. Dessa maneira, torna-se instantaneamente disponível tão logo se liga o computador, não sendo necessário carregá-lo de um disquete. O cabo de força da unidade acionadora deve ser ligado separadamente em uma tomada.

Os microcomputadores compatíveis com a linha TRS-80 utilizam unidades acionadoras de disquetes de 5 1/4 polegadas (minidisquetes), de face simples ou dupla e densidade dupla. É possível concectar até quatro unidades por micro. Alguns modelos de computador comportam dois acionadores no gabinete principal e mais dois em caixas separadas. Outras marcas aceitam apenas unidades externas.

Em geral, os acionadores já vêm instalados nos micros que os aceitam no gabinete principal, quando se compra o modelo com disquetes. Se você possui um computador desse tipo e pretende adicionar um ou dois acionadores, recomendamos que não o faça sozinho: peça auxílio ao revendedor, pois a instalação é razoavelmente complexa e requer a abertura do gabinete.

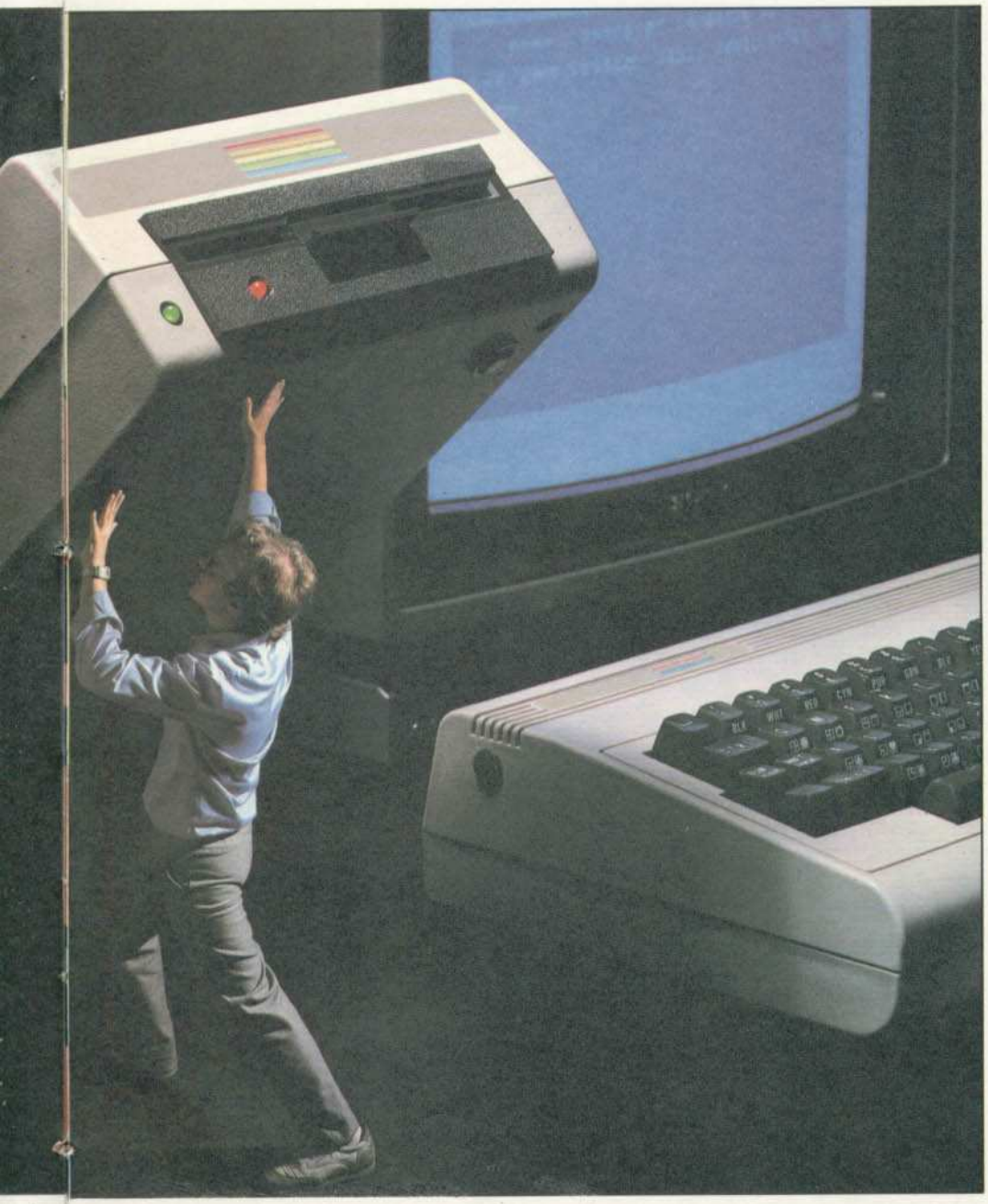
Nos computadores em que os acionadores são ligados externamente, a conexão é muito simples: o cabo de ligação, do tipo plano, tem na ponta um conector que deve ser inserido na porta de expansão de discos do micro. Para ligar mais de uma unidade, deve-se utilizar um cabo próprio, com vários conectores intercalados (*daysy chaining*). Não são necessários cabos de força separados para as unidades acionadoras, pois a alimentação ocorre por intermédio do cabo de conexão.

Os microcomputadores que são compatíveis com a linha Apple II utilizam unidades acionadoras de disquetes de 5 1/4 polegadas (minidisquetes), de face simples e densidade dupla. A placa de controle (interface) deve ser adquirida separadamente, e podem ser conectadas a ela até duas unidades acionadoras. O número de placas de controle que cabem no micro depende da marca, mas, de um modo geral, empregam-se até três placas. Os acionadores costumam alojar-

- COMO CONECTAR UMA UNIDADE DE DISQUETES
- CADEIA DE CONEXÃO
- CUIDADOS FUNDAMENTAIS
- FORMATAÇÃO DE DISCOS
- ARQUIVOS E SEUS NOMES
- ATUALIZAÇÃO DO CATÁLOGO
- O SISTEMA OPERACIONAL
- PRINCIPAIS COMANDOS DE SOFTWARE

se individualmente em caixas externas.

A conexão não é muito simples, porém com alguns cuidados, você mesmo pode fazê-la. Primeiro, desligue o micro da tomada e abra a tampa superior do console (UCP). Localize um conector interno vazio para colocar a placa de controle — utilizam-se, normalmente, os conectores 6 e/ou 5 para isso. O cabo de ligação da unidade, do tipo plano, tem na ponta um conector fêmea que deve ser inserido no macho correspondente, na placa de controle. Não são necessários cabos de força separados para as unidades acionadoras, pois a alimentação ocorre por intermédio do cabo de conexão.



O microcomputador TK-2000 pode usar apenas um acionador de disquetes, com discos de 5 1/4 polegadas (minidisquetes), de face simples e densidade dupla. O acionador é ligado ao console através de uma interface externa. Esta é adquirida à parte e deve ser inserida na porta de expansão do equipamento. Desligue o micro da tomada para proceder à conexão.

O cabo de força da interface e do acionador deve ser ligado separadamente na tomada, pois a UCP não tem potência suficiente para ambos.

Os microcomputadores compatíveis com a linha MSX I utilizam unidades acionadoras de disquetes de 5 1/4 polegadas (minidisquetes), de face simples e densidade dupla, ou microdisquetes de 3,5 polegadas, de face dupla (encontrados apenas no Exterior). Existem gabinetes de um ou de dois acionadores disponíveis no mercado.

A conexão é muito simples: o cabo de ligação tem a interface e um conector na ponta. Este deve ser inserido numa das portas de cartuchos do microcomputador. O software operacional de utilização já está gravado em memória ROM, na própria unidade de disco. Assim, torna-se instantaneamente disponível tão logo se liga o computador, não sendo necessário carregá-lo de um disquete. A unidade acionadora não requer cabo de força separado.

Não existem no mercado nacional unidades de disquete para os micros compatíveis com a linha Sinclair ZX Spectrum. Nos Estados Unidos e na Eu-

ropa, podem ser adquiridos facilmente dois tipos de sistemas de armazenamento magnético auxiliar para esses micros: os acionadores de disquetes de 5 1/4 polegadas, de face simples e densidade simples (em gabinetes com até dois acionadores, e descanso para o teclado), e os acionadores de fitas sem fim (*tape loop*). Estes, por suas características, aproximam-se dos disquetes em modo de operação, e custam bem mais barato. Os modelos mais conhecidos são o Microdrive, da própria Sinclair, e o Wafadrive. Podem ser ligados até oito acionadores em paralelo.

Para ligar o Microdrive, é necessário ter também uma Interface 1 da Sinclari, ou similar, que deve ser inserida na porta de expansão do teclado. Para fazê-lo, desligue o computador. O cabo de conexão, do tipo plano, tem duas pontas: uma para inserção na Interface e outra para o Microdrive. Cada Microdrive, por sua vez, tem um conector fêmea que possibilita a ligação de vários acionadores. Não são necessários cabos de força separados para os acionadores, pois os mesmos são alimentados pelo computador.

FORMATAÇÃO

Antes de usar o disquete para armazenar informação, deve-se prepará-lo por meio de um procedimento de software, chamado de formatação ou inicialização. O programa de formatação simplesmente impõe um padrão de trilhas e setores no disco virgem, definindo (isto é, gravando certos apontadores e listas) o modo como a informação será armazenada no disco.

Os disquetes são inicialmente divididos em trilhas concêntricas, por sua vez "fatiadas" em setores de igual número de bytes. O controlador de disquetes reconhece precisamente as marcas feitas pelo programa formatador. Esse procedimento é chamado de setoreamento por programa (*soft-sectoring*).

O processo de formatação varia conforme a marca do computador e o sistema operacional empregado. Eis aqui os comandos que você pode utilizar para formatar um disquete:

Coloque o disquete com o sistema operacional na unidade acionadora nº 0. Existem duas opções para a formatação de um disquete virgem nos micros desta linha: somente formatação — que produz um disquete sem o sistema operacional — e cópia total — que copia tudo o que o disco-mestre tem, inclusive o sistema operacional. A operação difere conforme o número de acionadores de disquetes do computador.

Para formatar, apenas, digite:

```
FORMAT
```

O programa fará, então, uma série de perguntas. Primeiro, pedirá que você indique a unidade em que está o disquete. Se você tem apenas um acionador, indique *drive 0* e retire o disquete com o sistema operacional do mesmo, colocando o disco virgem em seu lugar. Se você tem dois ou mais acionadores, não é preciso fazer isso: basta inserir o disquete virgem em outra unidade e indicar o seu número ao programa (por exemplo, *drive 1*).

Ao executar a formatação, o programa indicará o número da trilha que está formatando. Ao final, retire o disquete formatado da unidade.

No caso de formatação com cópia, use o comando:

```
BACKUP
```

Entre as informações solicitadas, o programa pede ao usuário que indique se deseja reformatar o disquete, caso ele já esteja previamente formatado. Se a resposta for sim, o programa funcionará como foi explicado anteriormente e, depois, fará a cópia do disquete-mestre. Se você tem apenas um acionador, precisará realizar várias vezes a operação de retirar disquete de cópia e inserir disquete-mestre. Fique atento para não trocar os discos. Convém proteger o disquete-mestre colocando uma etiqueta adesiva sobre o orifício quadrado da margem do envelope. Será preciso, ainda, saber a senha de acesso do disquete-mestre.

Coloque o disquete virgem na unidade acionadora, feche a porta da mesma e digite pelo teclado:

```
DSKINIT
```

Coloque o disquete com o sistema operacional na unidade acionadora 1. Proceda à inicialização, carregando o sistema operacional.

Tire o disquete do sistema e coloque o disquete virgem na unidade acionadora. Feche a porta da mesma e digite pelo teclado o comando **NEW**. Todos os disquetes com o sistema operacional precisam ter um programa em BASIC gravado. Este será carregado automaticamente toda vez que o computador for ligado. Escolha e digite o programa que desejar: ele pode conter desde uma linha de saudação até vários comandos que executem alguma tarefa específica. Em seguida, digite o comando:

```
INIT NOME
```

NOME refere-se ao programa inicial de carregamento, que normalmente recebe a designação de HELLO ou ALO. A formatação será, então, realizada.

Coloque o disquete com o sistema operacional na unidade acionadora A. A operação de formatação varia conforme o número de acionadores de disquete do computador.

Para formatar, apenas, digite:

```
FORMAT A:
```

se o seu computador tem apenas um acionador, ou:

```
FORMAT B:
```

se o seu computador tem dois acionadores.

O programa dará, então, algumas informações. Se você tem apenas um acionador, indique *drive A* e retire o disquete com o sistema operacional do mesmo, colocando o disco virgem em seu lugar. Se você tem dois ou mais acionadores, não é preciso fazer isso: basta inserir o disquete virgem em outra unidade e indicar o seu número ao programa (por exemplo, *drive B*).

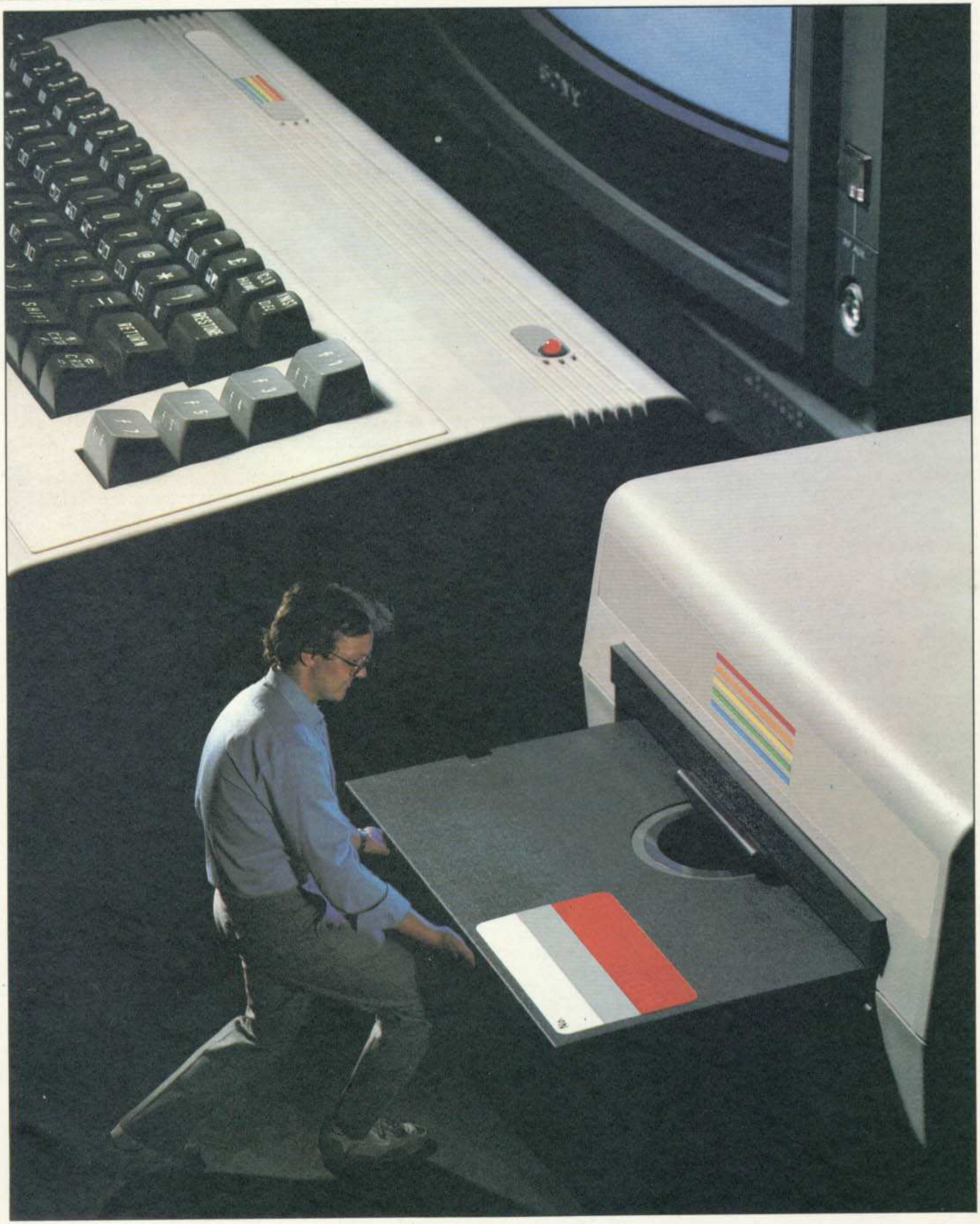
Ao final, retire o disquete formatado da unidade.

Para a formatação de um Microdrive, use o comando:

```
FORMAT "M";1;"NOME"
```

NOMES DE ARQUIVOS

Tudo o que é armazenado em disquete — seja um programa, seja um conjunto de dados — precisa ter um nome. O sistema operacional de disco trata de forma igual dados e programas: ambos são chamados de *arquivos*. O número de caracteres permitidos no nome de um arquivo de disco varia segundo o modelo do computador. A maioria deles aceita nomes formados por oito caracteres, no máximo. Os micros da linha Apple, po-


SHIFT
RETURN
RESTORE

rém, aceitam até trinta caracteres, e o Microdrive para o Spectrum, até dez. O Apple é, também, o único micro que permite a inclusão de espaços em branco e de outros caracteres especiais dentro de um nome de arquivo. Pode-se perfeitamente chamar um arquivo de DADOS DE ENTRADA, por exemplo. Os computadores das linhas TRS-80, TRS-Color e MSX, ao contrário, exigem que o nome do arquivo não tenha espaços em branco e que comece com um caractere alfabético.

Nesses computadores, quando se usa o comando **SAVE** para armazenar alguma coisa em disco, a máquina automaticamente adiciona um *sufixo* ao nome que foi fornecido. Esse sufixo pode ser **BAS**, que significa que um programa em BASIC foi gravado, **BAK**, que indica que o arquivo é uma cópia **(BAcK-up)** de outro arquivo no mesmo disco, ou **DAT**, que identifica um arquivo de dados, distinguindo-o de um programa. Conforme a linguagem, outros sufixos podem ser utilizados, mas devem ter sempre um máximo de três letras. Nos micros das linhas TRS-80 e TRS-Color, são separados do nome principal por uma barra (PROG/BAS, por exemplo); nos da linha MSX por um ponto (PROG. BAS, por exemplo).

Não existem sufixos nos nomes de arquivos para as linhas Apple e TK-2000.

Os nomes de arquivos são essenciais para a operação de um sistema que utiliza discos, pois cada um destes possui um catálogo, ou lista com todos os nomes de arquivos gravados. Tal catálogo (*catalog* ou *directory*) é atualizado sempre que um novo arquivo é criado, gravado ou apagado. Voltaremos a esse assunto mais adiante.

MICRO DICAS

COMO LIGAR E DESLIGAR O ACIONADOR

Ao ligar ou desligar o acionador, verifique se não há algum disco em seu interior. Como os disquetes são muito sensíveis, um setor do catálogo interno pode se apagar quando o acionador for ligado ou desligado.

Esse acidente ocorre com freqüência nos modelos cujo catálogo se situa nas trilhas mais externas, onde usualmente a cabeça fica em repouso. Recuperar a informação contida no restante do disco será, então, impossível, pois o software operacional não terá meios de localizá-la.

COMO USAR O DISQUETE

O conhecimento dos comandos de software é fundamental para o uso correto de unidades de disquetes, assim como de outros periféricos, entre os quais joysticks, canetas ópticas e impressoras. Esses comandos podem fazer parte do sistema operacional, das linguagens de programação (como o BASIC) ou, ainda, de ambos.

Como a sintaxe e o uso dos comandos variam conforme o micro, a linguagem e o sistema operacional, trataremos de cada máquina separadamente.

Os comandos utilizados pelos micros compatíveis com o TRS-Color para gravar e ler programas são o **SAVE** e o **LOAD** — assemelham-se, portanto, aos que se empregam com fita cassete (**CSAVE** e **CLOAD**). A maneira de usá-los também é bastante simples: se você quiser, por exemplo, gravar em disco um programa que está em memória, digite o comando **SAVE "NOME".** Para carregar, digite **LOAD "NOME".** Como o TRS-Color não tem sistema operacional separado do BASIC, trabalhar com uma unidade de disquetes é mais fácil.

Para carregar um programa em BASIC armazenado em disco, não precisa digitar o sufixo: basta indicar o nome dentro de um comando **LOAD.**

Quando se liga o computador pela primeira vez, a opção **VERIFY** é acionada, ou seja, tudo o que for gravado em disco será automaticamente verificado pelo computador. Para desativar essa opção, digite **VERIFY OFF.** Para reativá-la, digite **VERIFY ON.**

A lista dos arquivos gravados em um disquete será obtida com o comando:

```
DIR
```

que é uma abreviação de **DIR**ectory. Esse comando é muito útil, pois mostra na tela quantos bytes ocupa cada arquivo, se se trata de programa ou de dados, quais as suas características etc. **DIR** indica ainda quantos bytes estão sobrando no disquete.

Existe também um comando que possibilita a troca do nome de um arquivo já armazenado em disco. Se você digitar, por exemplo:

```
RENAME "VELHO" TO "NOVO"
```

o comando substituirá o nome do arquivo chamado VELHO por NOVO.

Para apagar arquivos do disco, utilize o comando **KILL.** Por exemplo, para um programa chamado INUTIL, digite:

```
KILL "INUTIL"
```

Os comandos utilizados pelos micros compatíveis com o TRS-80 para gravar e ler programas são o **SAVE** e o **LOAD** — assemelham-se, portanto, aos que se empregam com fita cassete (**CSAVE** e **CLOAD**). A maneira de usá-los também é simples: se você quiser, por exemplo, gravar em disco um programa que está em memória, digite **SAVE "NOME".** Para carregar, digite **LOAD "NOME".** O TRS-80 tem um sistema operacional separado do BASIC, mas os comandos de disco mais importantes também estão disponíveis na linguagem.

Para carregar um programa em BASIC armazenado em disco, não é necessário digitar o sufixo: basta indicar o nome dentro de um comando **LOAD.**

Para obter uma lista dos arquivos gravados em um disquete, a partir do sistema operacional, digite:

```
DIR
```

que é uma abreviação de **DIR**ectory. Esse comando é muito útil, pois mostra na tela quantos bytes ocupa cada arquivo, se se trata de programa ou de dados, quais as suas características etc. **DIR** indica ainda quantos bytes estão sobrando no disquete. O comando equivalente em BASIC é o **FILES**, de efeito semelhante, mas não igual ao do **DIR**.

Existe também um comando que possibilita a troca do nome de um arquivo já armazenado em disco. Digite, por exemplo, em BASIC:

```
RENAME "VELHO" TO "NOVO"
```

e o comando substituirá o nome do arquivo chamado VELHO por NOVO.

Para apagar arquivos do disco, utilize o comando **KILL**. Suponhamos que você queira apagar um programa chamado INUTIL. Digite, em BASIC:

```
KILL "INUTIL/BAS"
```

Os comandos empregados pelos micros compatíveis com as linhas Apple e TK-2000 para gravar e ler programas são o **SAVE** e o **LOAD**. A maneira de usá-los é muito simples: se você quiser, por exemplo, gravar em disco um programa que está em memória, digite **SAVE NOME.** Para carregar, digite **LOAD NOME.** O Apple e o TK-2000 não têm um sistema operacional separado do BASIC, sendo fáceis de usar.

Para obter uma lista dos arquivos gravados em um disquete, a partir do sistema operacional, digite:

```
CATALOG
```

Esse comando é muito útil, pois mostra na tela quantos bytes ocupa cada arquivo, se se trata de programa ou de dados etc.

Existe também um comando que possibilita a troca do nome de um arquivo já armazenado em disco. Digite, por exemplo, em BASIC:

```
RENAME VELHO, NOVO
```

e o comando substituirá o nome do arquivo chamado VELHO por NOVO.

Para apagar arquivos do disco, utilize o comando **DELETE**. Suponhamos que você queira apagar um programa chamado INUTIL. Digite, em BASIC:

```
DELETE INUTIL
```

Outro comando útil é o **VERIFY**, que informa se determinado arquivo foi danificado ou escrito incorretamente. Merecem ainda menção o comando **LOCK** e o **UNLOCK**, que permitem proteger ou retirar a proteção de arquivos contra apagamento acidental.

Os comandos utilizados pelos micros da linha MSX para gravar e ler programas são o **SAVE** e o **LOAD** — os mesmos, portanto, que se empregam com fita cassete. O modo de usá-los é simples: se você quiser, por exemplo, gravar em disco um programa que está em memória, digite **SAVE "A:NOME"**, para registrar um arquivo chamado NOME, na unidade de disquete A. Para carregar um programa, digite **LOAD "A:NOME"**. O MSX pode acomodar até dois disquetes, chamados de A e B.

Quando quiser carregar um programa em BASIC armazenado em disco, não digite o sufixo; basta indicar o nome dentro de um comando **LOAD.**

Para obter uma lista dos arquivos gravados em um disquete, a partir do sistema operacional, digite, em BASIC:

```
FILES
```

Esse comando é muito útil, pois mostra na tela quantos bytes ocupa cada arquivo, se se trata de programa ou de dados, quais as suas características etc.

Existe também um comando que possibilita a troca do nome de um arquivo já armazenado em disco. Digite, por exemplo, em BASIC:

```
NAME "A:VELHO" AS "A:NOVO"
```

e o comando substituirá o nome do arquivo chamado VELHO por NOVO (ambos no disquete A).

Para apagar arquivos do disco, utilize o comando **KILL.** Suponhamos que você queira apagar um programa chamado INUTIL. Digite, em BASIC:

```
KILL "A:INUTIL.BAS"
```

Os comandos **SAVE** e **LOAD** utilizados para a transmissão de dados entre computador e fita cassete foram modificados para permitir a gravação de leitura de programas no Microdrive:

```
LOAD *"m";1;"nome"
SAVE *"m";1;"nome"
```

O asterisco indica que o comando faz parte do BASIC armazenado na ROM da Interface 1. O **m** e o **1** informam que a operação se dará no Microdrive nº 1. Seguindo o mesmo formato, há um comando que verifica gravações depois de feitas (**VERIFY**) e outro que combina dois programas na memória.

O comando **CAT** — abreviatura de **CAT**álogo — é utilizado para exibir na tela a lista de arquivos armazenados em um cartucho de Microdrive.

Para apagar um arquivo do Microdrive, emprega-se o comando **ERASE**, que tem o mesmo formato que o **SAVE** e o **LOAD**, não exigindo, porém, o asterisco.

CUIDADOS COM A UNIDADE DE DISCOS

Ao contrário da UCP, do teclado e do vídeo, que são bastante resistentes e praticamente dispensam quaisquer reparos, as unidades de disco constituem o "calcanhar-de-Aquiles" dos sistemas baseados em microcomputadores. A confiabilidade de operação dessas unidades é muitas vezes maior do que a dos gravadores cassete, por exemplo. Porém, por serem dispositivos mecânicos relativamente complexos e muito delicados, exigem alguns cuidados por parte do usuário, se ele quiser operar por muitas horas, sem problemas.

Devido à sua fragilidade mecânica, as unidades de disquete são extremamente sensíveis a qualquer choque. Mesmo que nada se quebre depois de uma batida mais forte, pode ocorrer um descalibramento da cabeça. Portanto, mantenha a unidade em um ponto seguro da mesa. Se precisar transportá-la, faça-o com todo cuidado e dentro da embalagem acolchoada original.

É importante, também, levar em conta que a cabeça da unidade de disquete trabalha em atrito direto contra o meio magnético. Qualquer poeira, ou detrito, por menor que seja, que tiver acesso a ela, diminuirá a vida da mesma e provocará defeitos.

A temperatura ambiente deve ser mantida em níveis razoáveis durante a operação, sobretudo porque, normalmente, a unidade não possui ventilação.

Quanto aos cuidados gerais, lembre-se de que o cabo de conexão deve estar bem inserido, sem dobras e livres de tensões mecânicas. Limpe periodicamente os contatos dos conectores. Preste muita atenção ao inserir o disquete, para não fazê-lo com a orientação errada, o que pode danificar irremediavelmente a cabeça da unidade.

Evite usar disquetes "flippy", pois soltam fragmentos. Não use produtos de limpeza internamente: para isso, existem disquetes especiais de limpeza, que devem ser rodados a cada cinqüenta a cem horas de uso.

Quais são as vantagens da utilização de cartuchos de programas, em vez de fitas e discos? Posso gravar meus próprios programas em um cartucho?

A maior vantagem dos programas em cartuchos é que eles estão imediatamente disponíveis para uso, não sendo preciso carregá-los. Além disso, são muito mais seguros e resistentes contra a perda de programas.

Os cartuchos (como os existentes para micros das linhas TRS-Color e MSX) são memórias ROM removíveis, que não podem ser alteradas de nenhuma maneira — em outras palavras, constituem um meio permanente de armazenamento. Você não pode, portanto, gravar seus programas neles. Mas há a alternativa de gravação permanente de programas em um tipo de memória chamado EPROM (*Eraseable Programmable Read Only Memory*), o que requer um periférico especial, o programador de EPROM, que pode ser conectado a muitos micros. O custo da montagem de um sistema desse tipo é, porém, muito elevado.

OS SEGREDOS DO TRS-80 (1)

- RECURSOS OCULTOS DO TRS-80
- A MEMÓRIA DO VÍDEO
- O USO DOS COMANDOS PEEK e POKE
- COMO COPIAR A TELA

A memória ROM do TRS-80 guarda recursos que, em geral, os usuários desconhecem. Sabendo quais são eles, você poderá utilizar incríveis truques de programação em BASIC.

Os microcomputadores da linha TRS-80 são considerados tecnicamente obsoletos por muitas pessoas. Entretanto, os usuários desse popular modelo, que já tiveram a oportunidade de explorar mais a fundo seus diversos recursos, podem testemunhar o quanto eles são poderosos — algumas vezes, mais até do que os disponíveis em certos microcomputadores de tipo profissional, maiores e bem mais caros.

Na série de artigos que aqui iniciamos, você verá como explorar alguns dos recursos menos conhecidos do TRS-80 e de seus compatíveis nacionais (como o CP-300 e o CP-500, da Prológica) e internacionais. Daremos exemplos de manipulação de tela, teclado, som e de vários aspectos do BASIC pouco esclarecidos no Manual de Programação.

TRUQUES DE VÍDEO

O vídeo do TRS-80 possui uma propriedade muito interessante: ele é *mapeado em memória*. Isso significa que uma parte da memória RAM é dedicada exclusivamente ao vídeo, e que qualquer coisa nela armazenada terá, automaticamente, um efeito sobre o vídeo, sobretudo se se tratar de um caractere visível (ASCII ou gráfico).

A memória de vídeo começa na locação 15360 e ocupa 1024 bytes, ou seja, um para cada posição na tela. Há, portanto, uma equivalência entre o número indicado em um **PRINT@** e a memória absoluta referente à posição na tela, que é a soma de 15360 a esse número. Por exemplo, o comando

```
PRINT @ 125,"A";
```

colocará o código 65 (ASCII para a letra A maiúscula) na locação 15360 + 125, ou 15485.

PEEK E POKE

Como a memória de vídeo tem uma correspondência byte a byte com a tela, podemos escrever utilizando o comando **POKE**. A linha

```
POKE 15485,65
```

tem exatamente o mesmo efeito que o **PRINT@** mostrado acima.

Se você não quiser recorrer à tabela de códigos ASCII sempre que for escrever na tela com o **POKE**, use esta forma alternativa:

```
POKE 15485,ASC("A")
```

Em alguns casos, o emprego do **POKE** é mais vantajoso que o do **PRINT@**. Para colocar caracteres na tela sem provocar o movimento do cursor, por exemplo, o **POKE** é o comando indicado. Além disso, em certas situações ele pode ser mais rápido do que o **PRINT@**. Finalmente, o comando **POKE** permite a impressão de um caractere no canto inferior direito da tela (posição 1023), sem provocar o deslocamento de toda a tela para cima (*scrolling*).

Quando se trata, porém, de imprimir uma cadeia de caracteres (armazenados em uma variável literal, por exemplo), o uso do **POKE** fica em nítida inferioridade em relação ao do **PRINT.**

Para examinar o conteúdo de uma locação da tela, utiliza-se também a função **PEEK**, o que pode ser muito útil em uma grande variedade de programas, principalmente de jogos. O programa abaixo, por exemplo, lê o que está escrito na linha de cima da tela e, automaticamente, transforma em minúsculas todas as letras.

```
10 CLS
20 INPUT "ENTRE MENSAGEM ";I$
30 CLS:PRINT I$
40 FOR I=15360 TO 15423
50 J=PEEK(I):IF J>64 AND J<91
THEN POKE I,J+32
60 NEXT I:PRINT
```

O mesmo poderia ser feito tomando-se, um a um, os caracteres de **I$**, transformando-os em uma outra cadeia. Porém, o programa ficaria mais complicado e menos rápido. Note que a transformação de tipos maiúsculos para tipos minúsculos é bem fácil: basta, para isso, somar 32 ao código ASCII do caractere maiúsculo.

Um programa de jogo pode nos fornecer um exemplo menos óbvio de aplicação do **PEEK.** Suponhamos que seja necessário testar se uma bala de canhão (um caractere gráfico) colidiu com alguma parte da estrutura complexa de um castelo. Será bem mais conveniente utilizar um **PEEK** a cada posição de deslocamento da bala, uma vez que a função **POINT** não terá um desempenho satisfatório nesse caso.

UMA CÓPIA DA TELA

Muitos programas exigem que se faça uma cópia rápida da tela, em alguma parte da memória que não seja a de vídeo. Esse procedimento é empregado, por exemplo, nos programas que criam "janelas" de tela ou superpõem várias informações sobre o vídeo. Neste último caso, poderemos precisar de diversas cópias da tela, uma de cada "camada" superposta de informações, para que se torne possível a recuperação das telas originais posteriormente.

Uma maneira fácil, mas não rápida, de copiar uma tela consiste em usar os comandos **POKE** e **PEEK** dentro de um laço que percorre cada memória de vídeo. Para isso, é necessário reservar a parte da memória RAM que conterá a cópia. A reserva é feita logo que se aciona o interpretador BASIC, aparecendo na tela a pergunta **MEM?, MEMORY SIZE?** ou outra, conforme a versão do computador utilizado.

O programa abaixo usa a memória RAM acima de 30000:

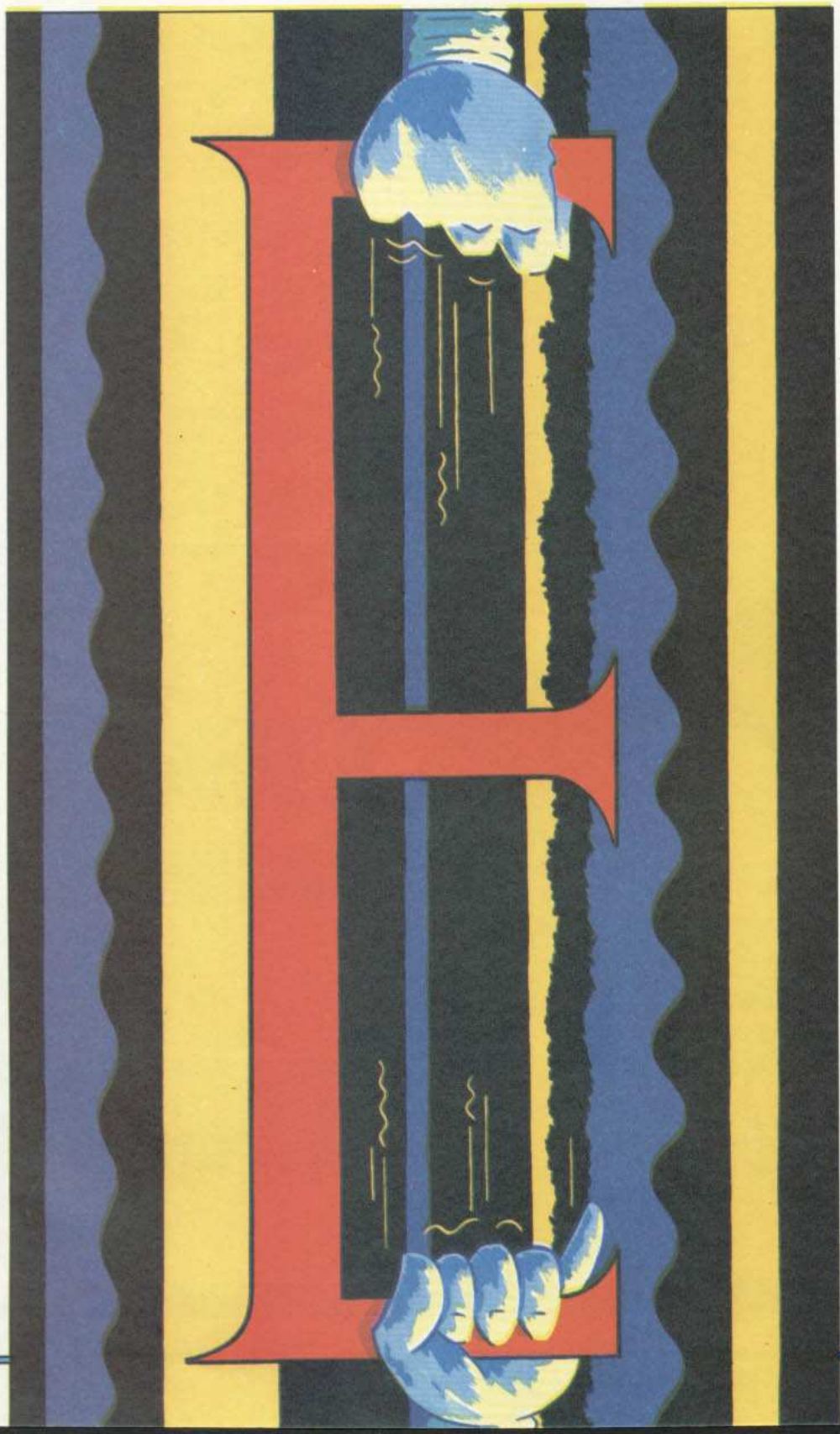
```
100 IM=30001
110 FOR I=15360 TO 16384
120 POKE IM,PEEK(I)
130 IM=IM+1:NEXT
```

Para trazer a cópia de volta para o vídeo, basta trocar os endereços do **PEEK** e do **POKE:**

```
100 IM=30001
110 FOR I=15360 TO 16384
120 POKE I,PEEK(IM)
130 IM=IM+1:NEXT
```

MANCHETES E LETREIROS (1)

- AMPLIAÇÃO DOS CARACTERES DA ROM
- ALTURA DAS LETRAS
- MONTAGEM DE LETRAS COM BLOCOS GRÁFICOS



Se você quiser escrever uma manchete ou um título em letras garrafais, precisará criar tipos especiais. Veja aqui dois diferentes métodos para a obtenção de caracteres ampliados.

O conjunto de caracteres disponíveis através do teclado deixa muito a desejar — o que é compreensível se considerarmos que ele foi criado de maneira a economizar espaço de memória. Assim, quando queremos escrever com letras maiores, destacando a mensagem, precisamos criar nossos próprios caracteres ampliados.

Existem muitas formas de utilizar os recursos gráficos de seu microcomputador para obter novos conjuntos de caracteres. A escolha de uma delas dependerá do tipo de aplicação que você tem em vista. Para ampliações, dispomos de duas alternativas básicas: desenhar as letras linha por linha no modo gráfico de alta resolução ou montar as letras com blocos gráficos.

Ambos os métodos podem ser usados dentro de um programa específico, destinado a escrever determinada frase — mas este é um caminho pouco econômico, já que cada mensagem exige uma nova rotina. A melhor solução, sobretudo quando pretendemos utilizar manchetes com freqüência, consiste em criar um programa que se encarregue de gerar os caracteres necessários. Uma vez feito isso, poderemos simplesmente informar ao computador o texto que deverá ser impresso, deixando a execução do serviço por conta do programa. Nosso trabalho se resumirá, assim, à gravação do gerador de letras e à sua colocação nos programas que precisem de caracteres maiores no vídeo.

Este é o primeiro de dois artigos em que são explicadas as diversas maneiras de escrever manchetes e cartazes. Aqui, trataremos em detalhe dos dois métodos já mencionados. O primeiro utiliza a tabela de padrões de caracteres existente na memória, aumentando seu formato. Como resultado, obtemos tipos semelhantes aos caracteres normais, só que em tamanho maior, disponíveis através

do teclado, da forma usual. Infelizmente, esse método não pode ser empregado pelos usuários do Apple e do TRS-Color, que não têm acesso aos padrões de caracteres.

O segundo método usa blocos gráficos da ROM para construir as letras, funcionando em todos os microcomputadores, menos no Apple, que não dispõe de blocos gráficos da ROM.

No próximo artigo, examinaremos outra maneira de criar letras, que pode ser adaptada a todo tipo de aplicação. Veremos, também, de que modo utilizar os métodos explicados no aperfeiçoamento de programas.

AMPLIAÇÃO DOS CARACTERES

Quando executamos o programa elaborado especialmente para ampliar caracteres, o computador aguarda a entrada da frase que deverá escrever. O Spectrum e o MSX, em seguida, colocam na tela as letras ampliadas, com o dobro da altura original.

Até adquirir alguma experiência e perceber como o programa funciona, escreva palavras curtas. Se usar frases mais longas, elas serão divididas no final da linha.

Com base nos bytes que definem os padrões dos caracteres, o programa cria os blocos gráficos que, juntos, formarão o novo caractere. Para duplicar a altura de uma letra, por exemplo, o computador substitui cada byte do caractere por dois outros, repetidos dentro do mesmo bloco gráfico.

```
1000 INPUT "INTRODUZA UM TEXTO"
, LINE I$: CLS : GOSUB 9000: GO
TO 1000
9000 LET line=0: LET col=-1
9010 LET y=0
9020 FOR i=1 TO LEN I$: LET col
=col+1: IF col=32 THEN  LET col
=0: LET line=line+2
9030 LET t$=I$(i)
9050 FOR x=0 TO 6 STEP 2
9060 POKE USR "a"+x,PEEK (15616
+(8*(CODE t$-32))+y)
9070 POKE USR "a"+1+x,PEEK (156
16+(8*(CODE t$-32))+y)
9080 LET y=y+1
9090 NEXT x
9100 FOR x=1 TO 7 STEP 2
9110 POKE USR "b"+x-1,PEEK (156
16+(8*(CODE t$-32))+y)
9120 POKE USR "b"+x,PEEK (15616
+(8*(CODE t$-32))+y)
9130 LET y=y+1
9140 NEXT x
9150 LET y=0
9160 PRINT AT line,col;CHR$ 144
;AT line+1,col;CHR$ 145
9180 NEXT i
9200 RETURN
```

A primeira linha do programa possibilita a entrada de nosso texto, colocando-o dentro da variável alfanumérica **I$**. Para controlar a posição de impressão do texto na tela, utilizam-se duas variáveis. Seus valores iniciais são 0, para a coordenada vertical, e −1, para a coordenada horizontal. Uma terceira variável, **Y**, recebe o valor inicial zero, antes que o processo de ampliação comece.

A linha 9020 inicia um laço **FOR... NEXT** que dará um número de voltas igual ao comprimento de nossa frase. A cada volta, o valor da coordenada horizontal (variável **col**) aumenta em uma unidade. O valor inicial de **col** era −1 justamente para que a impressão do texto começasse a partir do canto esquerdo da linha, onde **col** = 0.

Se chegar a 32 (uma posição além da extremidade direita da linha), o valor da coordenada horizontal voltará a ser 0, para que o computador coloque as letras restantes a partir do lado esquerdo da tela. A coordenada vertical é, então, aumentada em duas unidades, tornando possível que o texto excedente seja escrito na linha seguinte.

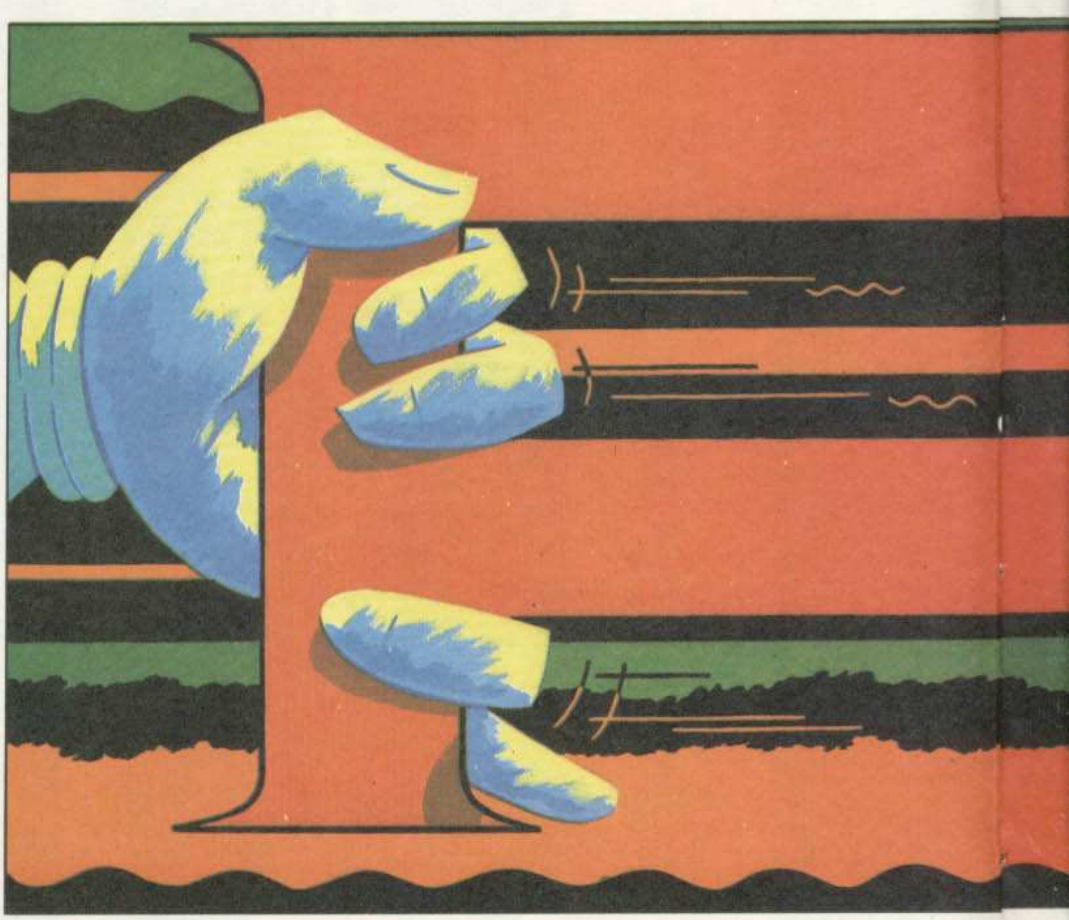
A linha 9030 coloca a letra que deve ser ampliada dentro da variável **t$**, usando o contador do laço, **i** (**t$** conterá, assim, o i-ésimo caractere do texto). Em seguida, o computador dá início a um novo laço, que coloca dois bytes repetidos dentro de um bloco gráfico, quatro vezes.

A CHAVE DO PROCESSO

As contas feitas para calcular o número a ser colocado dentro do bloco constituem a chave de todo o processo de ampliação das letras.

Analisando a linha 9060, você poderá entender os cálculos. A primeira metade é muito simples. Ela coloca um número (cujo cálculo é explicado adiante) com **POKE** em um endereço **x** posições além do primeiro byte do bloco gráfico (UDG) **a**. A variável **x**, que controla o laço **FOR...NEXT**, aumenta com um **STEP** 2, de forma que o segundo byte não sofre alterações quando o laço é executado novamente.

A segunda metade calcula o número que será colocado pelo **POKE**. Esse número corresponde a um dos bytes do padrão do caractere que está sendo amplia-



do, obtido da tabela que fica na ROM. O endereço inicial da tabela é 15616. A fórmula entre parênteses subtrai 32 do código ASCII do caractere e multiplica o resultado por 8, para identificar a posição do primeiro byte do padrão dentro da tabela.

O código ASCII do caractere precisa ser subtraído em 32 unidades, antes da multiplicação por 8, devido ao modo como o Spectrum armazena os padrões dos caracteres (de fato, não há nessa parte da memória oito bytes para cada um dos 32 primeiros caracteres).

Depois de calcular a posição dos padrões do caractere dentro da tabela, o computador soma o valor obtido ao endereço inicial da tabela, 15616, e adiciona o resultado a **y**. Lembre-se de que no início do programa atribuímos a **y** o valor zero. Essa variável será utilizada para indicar o byte que está sendo lido e colocado dentro do UDG. Quando seu valor é zero, o computador lê e coloca no UDG o primeiro byte do caractere.

As linhas 9060 e 9070 colocam em duas linhas consecutivas do UDG o mesmo byte obtido na tabela de padrões, duplicando a altura da letra. Depois, **y** é aumentado em uma unidade e o processo se repete para o próximo byte do padrão, até que o laço **FOR... NEXT** termine — a esta altura, quatro pares de bytes terão sido colocados no UDG **a**.

O Spectrum inicia, então, outro laço **FOR...NEXT**, que procede como o anterior para colocar os quatro últimos bytes do padrão do caractere no UDG **b**, sempre repetindo cada byte duas vezes. O laço termina na linha 9140 e o programa faz **y** voltar a valer 0.

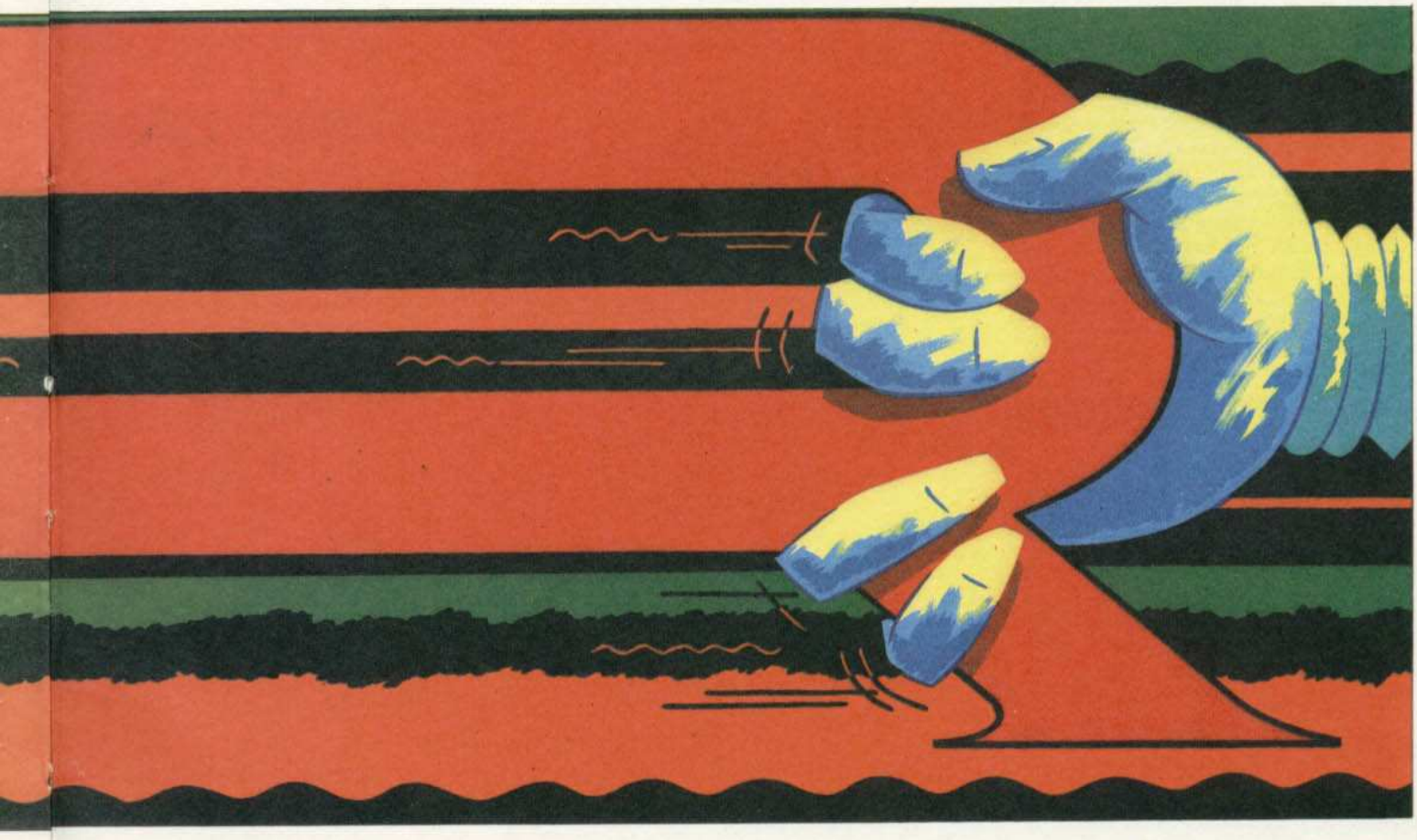
A linha 9160 é a responsável pela impressão dos blocos gráficos, um acima do outro. Ela usa as variáveis **line** e **col** para identificar a posição de impressão do caractere ampliado. No programa, o comando **PRINT "A"** foi substituído por **CHR$ 144** para evitar que o leitor se confunda — o "A" normal e o "A" gráfico pareceriam absolutamente iguais na listagem.

Finalmente, a linha 9180 conclui o laço principal do programa, mandando o computador ampliar a letra seguinte. Ao completar a frase, o computador volta à linha 1000, que possibilita a entrada de um novo texto.

Embora o Apple não nos dê acesso aos padrões binários que definem o formato de seus caracteres, veremos como aplicar a técnica de ampliação utilizando os UDG criados por um programa do artigo da página 526.

Na listagem que se segue, você encontrará apenas as modificações necessárias para ampliar as letras criadas no programa mencionado. Sem as linhas **DATA** iniciais, que se encontram listadas naquele artigo, você não obterá nenhum resultado. Mais ainda, se tentar executar apenas as linhas aqui apresentadas, o computador sairá fora de seu controle e as linhas digitadas acabarão se perdendo.

```
799  HGR
800 A$ = "ESTA MENSAGEM E' UM T
ESTE"
810 C = 7:L = 11: GOSUB 1000
820  END
1000 N = L * 40 + C
1010  FOR J = 1 TO  LEN (A$)
1020 B$ =  MID$ (A$,J,1)
1030 B =  ASC (B$)
1040 L =  INT (N / 40):C = N -
40 * L
1050  FOR I = 0 TO 3
1060  POKE T + (L - 8 * (L > 7)
 - 8 * (L > 15)) * 128 + 40 * (
L > 7) + 40 * (L > 15) + C + 10
24 * 2 * I, PEEK (E + B * 8 + I
).
1065  POKE T + (L - 8 * (L > 7)
```



```
 - 8 * (L > 15)) * 128 + 40 * (
L > 7) + 40 * (L > 15) + C + 10
24 * (2 * I + 1), PEEK (E + B *
 8 + I)
1066  NEXT I:L = L + 1
1070  FOR I = 0 TO 3
1073  POKE T + (L - 8 * (L > 7)
 - 8 * (L > 15)) * 128 + 40 * (
L > 7) + 40 * (L > 15) + C + 10
24 * 2 * I, PEEK (E + B * 8 + I
 + 4)
1076  POKE T + (L - 8 * (L > 7)
 - 8 * (L > 15)) * 128 + 40 * (
L > 7) + 40 * (L > 15) + C + 10
24 * (2 * I + 1), PEEK (E + B *
 8 + I + 4)
1079  NEXT I:N = N + 1: NEXT J
1080  HCOLOR= 6
1090  HPLOT 25,80 TO 250,80 TO
250,105 TO 26,105 TO 26,80
1100  RETURN
```

Consultando o artigo anteriormente citado, veremos que nosso programa utiliza a sub-rotina que começa na linha 1000 para colocar os padrões das letras da frase contida em **A$** na tela de alta resolução. As variáveis **C** e **L** definem a linha e a coluna em que faremos a impressão, sempre supondo que a tela tem quarenta colunas e 24 linhas.

A linha 1000 coloca em **N** a posição da tela em que a primeira letra da frase será escrita. A linha 1010 inicia um laço **FOR...NEXT** que dará tantas voltas quantas forem as letras da frase que desejamos imprimir.

A variável alfanumérica **B$** contém a letra que está sendo escrita. A cada volta, um novo caractere é colocado ali, com o auxílio da função **MID$**, e a linha e a coluna de impressão são recalculadas a partir do valor de **N**.

Em seguida, o programa inicia um laço **FOR...NEXT** que coloca na tela dois bytes repetidos do padrão da letra, quatro vezes. Os cálculos feitos para a repetição dos bytes são a chave do processo de ampliação.

FUNCIONAMENTO DO PROGRAMA

Já tivemos a oportunidade de mostrar aos usuários do Apple e do TK-2000 como é caótica a organização da memória de vídeo desses dois micros. A ordem de preenchimento das linhas de pixels do Apple é tão confusa que quem não tem certa familiaridade com a matemática dificilmente conseguirá entender as fórmulas das linhas 1060 a 1076.

Mas, no nosso caso, isso não tem tanta importância. Para compreender o funcionamento do programa, basta saber que o primeiro laço coloca na tela a metade superior da letra ampliada e o segundo, a metade inferior. Um caractere comum ocuparia oito linhas de vídeo com seus oito bytes. Aqui, para ampliar as letras, cada byte é repetido duas vezes, resultando nas dezesseis linhas do caractere ampliado.

Letras de altura dupla são ideais para páginas-título de jogos ou outros programas.

Assim, a linha 1060 coloca os quatro primeiros bytes do padrão da letra nas posições pares do bloco gráfico correspondente à metade superior do caractere ampliado. A linha 1065, por sua vez, coloca os mesmos quatro primeiros bytes nas posições ímpares daquele bloco. O processo se repete nas linhas 1073 e 1076, só que os bytes são colocados no bloco correspondente à metade inferior do caractere.

```
10 FOR J=0 TO 50 STEP 2
20 FOR X=0 TO 7 STEP 2
30 VPOKE BASE(2)+1024+J*8+X,PEE
K((PEEK(4))+(PEEK(5)*256)+520+Y
)
40 VPOKE BASE(2)+1024+J*8+X+1,P
EEK((PEEK(4))+(PEEK(5)*256)+520
+Y)
50 LET Y=Y+1
60 NEXT X
70 FOR X=8 TO 15 STEP 2
80 VPOKE BASE(2)+1024+J*8+X,PEE
K((PEEK(4))+(PEEK(5)*256)+520+Y
)
90 VPOKE BASE(2)+1024+J*8+X+1,P
EEK((PEEK(4))+(PEEK(5)*256)+520
+Y)
100 LET Y=Y+1
110 NEXT X
120 NEXT J
130 CLS
140 PRINT"INTRODUZA A PALAVRA"
150 INPUT I$
160 CLS:L=0:C=0
170 FOR I=1 TO LEN(I$)
180 LET B=ASC(MID$(I$,I,1))
190 IFB<65 OR B>90 THEN GOTO 21
0
200 LOCATE C,L:PRINTCHR$(128+2*
(B-65)):LOCATE C,L+1:PRINTCHR$(
129+2*(B-65))
210 LET C=C+1
220 IF C=40 THEN LET C=0:L=L+2
230 NEXT I
240 GOTO 140
```

O programa começa com a rotina que transfere os dados de definição dos caracteres da ROM para a memória de vídeo (VRAM), nas linhas 10 a 120.

A seção que vai da linha 130 à 150 limpa a tela e pede que o usuário introduza a palavra ou mensagem a ser ampliada. As linhas 160 a 240 contêm a rotina que imprime as letras no vídeo. As variáveis **L** e **C** registram, respectivamente, a linha e a coluna em que serão impressos os novos caracteres gráficos que irão compor as letras.

O laço que se inicia na linha 170 imprime uma letra a cada valor de **I**, sempre acrescentando 1 à variável **C**, para que as letras fiquem uma ao lado da outra. Quando a frase alcança a extremidade da tela, a linha 200 zera **C** e acrescenta 2 à variável **L**, a fim de que a próxima letra seja escrita na primeira coluna da linha seguinte.

A linha 180 faz a variável **B** assumir o valor do código ASCII do caractere a ser ampliado e a linha 190 verifica se as letras são maiúsculas. A linha 200 imprime os dois blocos gráficos que irão compor a letra, ambos na mesma coluna, mas um abaixo do outro.

Para saber como se obtêm os valores colocados dentro dos parênteses após o comando **CHR$**, é preciso entender o processo de ampliação.

O PROCESSO DE AMPLIAÇÃO

Ao ser ligado, o MSX copia o padrão dos caracteres gravados na ROM na parte de sua memória dedicada ao vídeo (VRAM). É a partir dessa tabela da VRAM que ele imprime as letras no vídeo. Cabe ao nosso programa criar os blocos gráficos que irão compor as letras ampliadas e colocá-los na tabela da VRAM, no lugar que antes era ocupado pelos caracteres padronizados, a partir do caractere de código 128.

Sabemos que o padrão de um caractere é formado por oito bytes. Podemos, portanto, obter um caractere ampliado repetindo duas vezes cada byte. O endereço do padrão dos caracteres na ROM está registrado nos bytes 4 e 5 e o endereço do padrão de caracteres na VRAM é dado por **BASE(2)**. Somando o endereço aí contido a 1024, teremos o endereço da VRAM, que será usado pelo **VPOKE**.

Na linha 30, a viriável **J** simplesmente incrementa a posição nessa memória. O **PEEK** examina o endereço correspondente na RAM, onde estão os padrões gráficos. Para a obtenção do código da

letra A, adiciona-se o número 520. O Y indica o byte a ser copiado. As linhas 70 a 110 do programa fazem a mesma cópia para a parte inferior do bloco gráfico de cada caractere.

O USO DOS CARACTERES GRÁFICOS

Além de ampliar os padrões preexistentes na memória do micro, podemos criar nossas próprias letras. Essa alternativa é especialmente importante para os usuários do TRS-Color, que não permite o acesso aos padrões das letras. Aqui estão alguns programas que utilizam caracteres gráficos do computador para construir letras ampliadas.

```
 10 DATA "▛▀▀▜","▛▀▀▜",
   "▛▀▀▜","▛▀▀▜"
 20 DATA "▌  ▐","▙▄▄▟",
   "▌  ▐","▌  ▐"
 30 DATA "▛▀▀▜","▌  ▐",
   "▌  ▐","▌  ▐"
 40 DATA "▌  ▐","▙▄▄▟",
   "▙▄▄▟","▙▄▄▟"
 50 DATA "▛▀▀▜","▛▀▀▜",
   "▛▀▀▜","▌  ▐"
 60 DATA "▙▄▄▟","▌  ▐",
   "▌  ▐","▙▄▄▟"
 70 DATA "▌  ▐","▛▀▀▜",
   "▌  ▐","▌  ▐"
 80 DATA "▙▄▄▟","▌  ▐",
   "▙▄▄▟","▌  ▐"
 90 DATA "▛▀▀▜","▝▀▀▜",
   "▌  ▐","▌  ▐"
100 DATA "▌  ▐","▐  ▌",
   "▟▄▄▙","▌  ▐"
110 DATA "▌  ▐","▐  ▌",
   "▝▀▀▘","▌  ▐"
120 DATA "▙▄▄▟","▙▄▄▌",
   "▌  ▐","▙▄▄▟"
130 DATA "▙▄▄▟","▙  ▌",
   "▛▀▀▘","▙▄▄▟"
140 DATA "▌  ▐","▙▄▄▟",
   "▌  ▐","▌  ▐"
150 DATA "▌  ▐","▙▄▄▟",
   "▌  ▐","▛▀▀▘"
160 DATA "▌  ▐","▌  ▐",
   "▙▄▄▟","▌  ▐"
170 DATA "▛▀▀▜","▛▀▀▜",
   "▛▀▀▘","▝▀▀▜"
180 DATA "▌  ▐","▌  ▐",
   "▙▄▄▟","▐  ▌"
190 DATA "▌  ▐","▛▀▀▜",
   "▐  ▌","▐  ▌"
200 DATA "▙▄▄▟","▌  ▐",
   "▙▄▄▟","▐  ▌"
210 DATA "▌  ▐","▌  ▐",
   "▐  ▌","▌  ▐"
220 DATA "▌  ▐","▌  ▐",
   "▐  ▌","▚▞  ▐"
230 DATA "▌  ▐","▌  ▐",
   "▐  ▌","▚▞  ▐"
240 DATA "▙▄▄▟","▚▞  ▐",
   "▙▄▄▟","▌  ▐"
250 DATA "▌  ▐","▀▀▀▜"
260 DATA "▙▄▄▟","▐  ▀▜"
270 DATA "▐  ▌","▜▛  ▐"
280 DATA "▐  ▌","▙▄▄▟"
290 POKE 23658,8: DIM a$(27,3)
: DIM b$(27,3): DIM c$(27,3):
DIM d$(27,3)
300 FOR j=0 TO 21 STEP 4
310 FOR i=1 TO 4: READ a$(i+j)
: NEXT i
320 FOR i=1 TO 4: READ b$(i+j)
: NEXT i
330 FOR i=1 TO 4: READ c$(i+j)
: NEXT i
340 FOR i=1 TO 4: READ d$(i+j)
: NEXT i
350 NEXT j
360 FOR i=25 TO 26: READ a$(i)
: NEXT i
370 FOR i=25 TO 26: READ b$(i)
: NEXT i
380 FOR i=25 TO 26: READ c$(i)
: NEXT i
390 FOR i=25 TO 26: READ d$(i)
: NEXT i
400 INPUT "Introduza palavra(m
ax 10letras)", LINE t$: IF LEN
t$>10 THEN  LET t$=t$( TO 10)
405 IF LEN t$=0 THEN  GOTO 400
410 LET s$="": FOR i=1 TO LEN
t$: IF CODE t$(i)<65 OR CODE t
$(i)>90 THEN  LET t$(i)=CHR$
91
420 LET s$=s$+a$(CODE t$(i)-64
): NEXT i
430 PRINT s$
440 LET s$="": FOR i=1 TO LEN
t$
450 LET s$=s$+b$(CODE t$(i)-64
): NEXT i
460 PRINT s$
470 LET s$="": FOR i=1 TO LEN
t$
480 LET s$=s$+c$(CODE t$(i)-64
): NEXT i
490 PRINT s$
500 LET s$="": FOR i=1 TO LEN
t$
510 LET s$=s$+d$(CODE t$(i)-64
): NEXT i
520 PRINT s$
530 PRINT : GOTO 400
```

```
10 SCREEN 0
20 DIM A(78),B(78),C(78),D(78)
30 FOR I=1 TO 78:READ S$:A(I)=V
AL("&H"+S$):NEXT
40 FOR I=1 TO 78:READ S$:B(I)=V
AL("&H"+S$):NEXT
50 FOR I=1 TO 78:READ S$:C(I)=V
AL("&H"+S$):NEXT
60 FOR I=1 TO 78:READ S$:D(I)=V
AL("&H"+S$):NEXT
70 PRINT"introduza até 9 letras
"
80 INPUT T$:IF LEN (T$)>9 OR LE
N (T$)=0 THEN 80
90 FOR X=1 TO 4
100 FOR Y=1 TO LEN(T$)
110 A$=MID$(T$,Y,1)
120 A=ASC(A$)
```

MICRODICAS

COMO USAR A IMPRESSORA PARA CONFECCIONAR CARTAZES E FAIXAS

Os programas elaborados para desenhar letras grandes na tela do microcomputador têm uma utilidade adicional: a impressão de grandes cartazes e faixas.

Como sabemos, as letras normais, obtidas em uma impressora comum para microcomputador, são muito pequenas para serem vistas a distância. Precisamos, assim, lançar mão de métodos semelhantes aos apresentados neste artigo para imprimir letras de tamanho maior.

A escolha da melhor solução dependerá, evidentemente, do tipo de impressora que você possui.

O tipo mais simples é aquele capaz de imprimir apenas texto, não dispondo de blocos gráficos. Nesse caso, programas que trabalham com blocos gráficos da ROM, como os destinados ao Spectrum, não podem ser utilizados diretamente com a impressora, pois ela não os copiará.

Uma boa alternativa será modificar os programas, de modo a compor letras grandes a partir dos caracteres disponíveis na impressora. Experimente, por exemplo, o recurso muito comum de usar repetições do próprio caractere, como mostramos, abaixo, com a letra I maiúscula:

```
IIIIII
  II
  II
  II
  II
IIIIII
```

As linhas **DATA**, que contêm os códigos dos blocos gráficos, podem ser alteradas para descrever a disposição dos caracteres usados na composição. Outra providência necessária é mudar os comandos **PRINT** (que colocam o resultado apenas na tela) para o equivalente ao comando de impressão existente em seu computador: **LPRINT** (nos micros TRS-80, Sinclair e MSX), **PR # 1** (no Apple) e **PRINT # 1** (no TRS-Color).

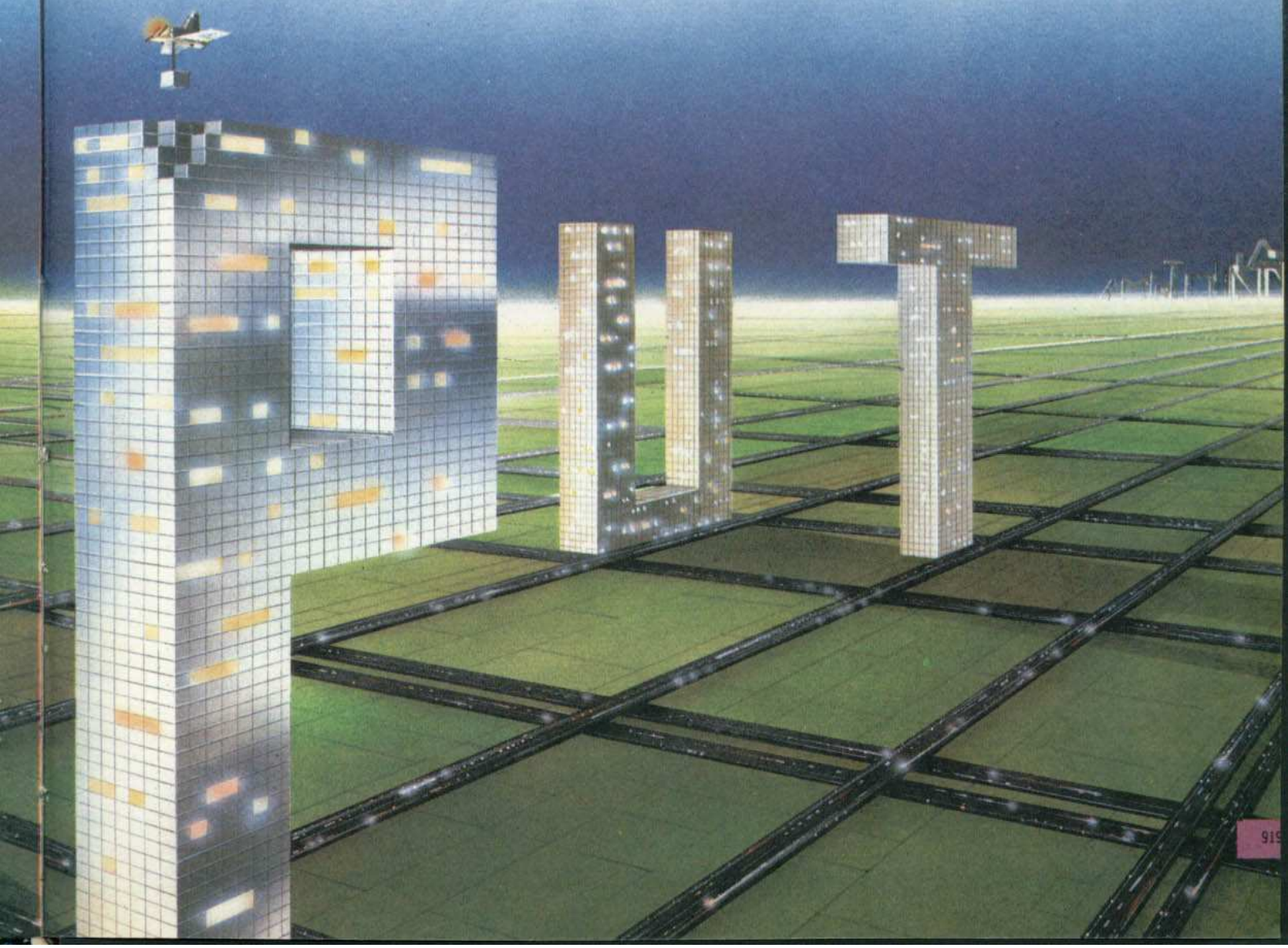
Se sua impressora tiver capacidade gráfica, o trabalho será ainda mais fácil. Usando um comando específico, como o **COPY**, nos micros da linha Sinclair, ou um programa de descarregamento de tela, você poderá transferir para o papel o cartaz desenhado na tela do computador com os programas aqui apresentados.

```
127 IF A<65 OR A>90 THEN CLS:GO
TO 70
129 LET A=A-64
130 B=(A-1)*3
140 ON X GOSUB 190,200,210,220
150 NEXT Y
160 PRINT
170 NEXT X
180 GOTO 70
190 FOR R=B+1 TO B+3:PRINTCHR$(
A(R));:NEXT R: PRINT" ";:RETURN
200 FOR R=B+1 TO B+3:PRINTCHR$(
B(R));:NEXT R:PRINT" ";:RETURN
210 FOR R=B+1 TO B+3:PRINTCHR$(
C(R));:NEXT R:PRINT" ";:RETURN
220 FOR R=B+1 TO B+3:PRINTCHR$(
D(R));:NEXT R:PRINT" ";:RETURN
230 DATA C7,DF,D6,DB,DF,D6
240 DATA C7,DF,D6,DB,DF,D6
250 DATA DB,DF,D3,DB,DF,D3
260 DATA DB,DF,DD,DB,20,DD
270 DATA D5,DB,D3,D5,DB,D3
280 DATA DB,20,DD,DB,20,20
290 DATA DB,DE,DD,DB,20,DD
300 DATA C7,DF,D6,DB,DF,D6
310 DATA C7,DF,D6,DB,DF,D6
320 DATA C7,DF,D6,DF,DB,D3
330 DATA DB,20,DD,DD,20,DD
340 DATA DD,20,DD,DD,20,DD
350 DATA DD,20,DD,DF,DF,D6
360 DATA DD,20,DD,DB,DC,D3
370 DATA DD,20,20,DB,20,DD
380 DATA DB,DC,20,DB,DC,20
390 DATA DB,20,20,DB,DC,DD
400 DATA 20,DB,20,20,DB,20
410 DATA DB,C7,20,DB,20,20
420 DATA DD,DD,DD,DD,DD,DD
430 DATA DD,20,DD,DB,20,DD
440 DATA DD,20,DD,DB,20,DD
450 DATA C1,DC,20,20,DB,20
460 DATA DB,20,DD,DD,20,DD
470 DATA DD,D6,DD,D5,C7,20
480 DATA C1,D4,D3,20,C7,20
490 DATA C7,DF,D6,DB,20,DD
500 DATA DD,20,20,DB,20,DD
510 DATA DB,20,20,DB,20,20
520 DATA DB,D5,DD,DB,20,DD
530 DATA 20,DB,20,D6,DB,20
540 DATA DB,C1,20,DB,20,20
550 DATA DD,20,DD,DD,DD,DD
560 DATA DD,20,DD,DB,DF,20
570 DATA DD,D6,DD,DB,DF,D6
580 DATA 20,20,DD,20,DB,20
590 DATA DB,20,DD,DE,DE,20
600 DATA DD,DD,DD,D4,C1,20
610 DATA 20,DD,20,D4,D3,20
620 DATA DD,20,DD,DB,DC,D3
630 DATA C1,DC,D3,DB,DC,D3
640 DATA DB,DC,D6,DB,20,20
650 DATA DB,DC,DD,DB,20,DD
660 DATA D4,DB,D6,C1,DB,20
670 DATA DB,20,DD,DB,DC,DD
680 DATA DD,20,DD,DD,DE,DD
```

```
690 DATA C1,DC,D3,DB,20,20
700 DATA C1,C7,D6,DB,20,DD
710 DATA C1,DC,D3,20,DB,20
720 DATA DB,DC,DD,20,DD,20
730 DATA C1,C1,D3,DD,20,DD
740 DATA 20,DD,20,C1,DC,D6
```

T

```
10 DIM L(2,3,25)
20 FOR J=0 TO 25:FOR K=0 TO 3:F
OR L=0 TO 2:READ L(L,K,J):NEXT
L,K,J
30 CLS0
40 PRINT @480,"INTRODUZA A PALA
VRA -";B$;
50 A$=INKEY$:IF(A$<"A" OR A$>"Z
")AND A$<>CHR$(8) AND A$<>CHR$(
13) AND A$<>" " THEN 50
60 IF A$=CHR$(13) THEN 110
70 IF A$=CHR$(8) AND B$="" THEN
 50
80 IF A$=CHR$(8) THEN B$=LEFT$(
B$,LEN(B$)-1):GOTO 30
90 IF LEN(B$)>9 THEN 50
100 B$=B$+A$:GOTO 30
110 IF B$="" THEN CLS:END
120 CLS0:PRINT @480,"COR (1-8)
?";
130 A$=INKEY$:IF A$<"1" OR A$>"
8" THEN 130
140 CLS0:CL=VAL(A$)
150 FOR Y=0 TO 3:FOR C=1 TO LEN
(B$):FOR X=0 TO 2
160 IF MID$(B$,C,1)=" " THEN PR
INT CHR$(128);:GOTO 180
170 PRINT CHR$(84+CL*16+L(X,Y,A
SC(MID$(B$,C,1))-65));
180 NEXT X,C:PRINT STRING$(32-P
OS(0),128);:NEXT Y
190 B$="":GOTO 40
1000 DATA 42,40,38,38,28,38,42,
40,38,38,28,38,42,40,30,39,31,3
6,38,28,38,39,31,36
1010 DATA 42,40,38,38,28,28,38,
28,28,39,31,38,42,40,30,38,28,3
8,38,28,38,39,31,36
1020 DATA 42,40,36,39,31,30,38,
28,28,39,31,30,42,40,36,38,28,2
8,42,40,36,38,28,28
1030 DATA 42,40,38,38,28,28,38,
29,30,39,31,38,38,28,38,39,31,3
8,38,28,38,38,28,38
1040 DATA 40,42,36,28,38,28,28,
38,28,31,39,30,28,41,36,28,33,2
8,28,33,28,39,35,28
1050 DATA 38,29,36,39,36,28,42,
30,28,38,32,30,38,28,28,38,28,2
8,38,28,28,39,31,30
1060 DATA 39,29,38,38,38,38,38,
28,38,38,28,38,39,28,38,42,30,3
8,38,37,38,38,32,38
```



Letras maiúsculas no TRS-Color...

...e no Spectrum.

```
1070 DATA 42,40,38,38,28,38,38,
28,38,39,31,38,42,40,38,38,28,3
8,42,40,36,38,28,28
1080 DATA 42,40,38,38,28,38,38,
30,38,39,35,38,42,40,38,38,28,3
8,42,41,36,38,28,38
1090 DATA 42,40,38,39,31,30,28,
28,38,39,31,38,40,42,36,28,38,2
8,28,38,28,28,38,28
1100 DATA 38,28,38,38,28,38,38,
28,38,39,31,38,38,28,38,38,28,3
8,38,28,38,32,34,28
1110 DATA 38,28,38,38,28,38,38,
38,38,39,39,38,38,28,38,32,34,2
8,29,37,28,38,28,38
1120 DATA 38,28,38,39,31,38,28,
38,28,28,38,28,40,40,38,28,29,3
6,29,36,28,39,31,30
```

Os programas apresentados funcionam de maneira bem parecida. Eles começam dimensionando as matrizes que irão conter os caracteres gráficos da ROM necessários à formação das diversas letras. O Spectrum emprega o comando **POKE** para travar as letras maiúsculas. O Spectrum e o MSX usam quatro matrizes, uma para cada linha das letras. Os demais trabalham só com uma matriz.

Ao ser rodado, o programa demora um pouco para funcionar, devido à leitura das linhas **DATA** pelo comando **READ**. No Spectrum, essa tarefa é feita pelas linhas 290 a 390 do programa; no MSX, pelas linhas 30 a 60; nos demais computadores, só pela linha 20.

Em seguida, inicia-se a rotina principal, que possibilita ao usuário escrever uma palavra. Esta deve ter, no máximo, dez letras.

Depois de verificar se a cadeia de caracteres digitada é válida, o programa começa um laço **FOR...NEXT** que imprime uma das letras a cada volta. Esse laço é igual ao do último programa deste artigo, e só termina quando a última letra tiver sido impressa.

O Spectrum usa quatro laços **FOR...NEXT**, um para cada linha das letras, já que elas têm quatro caracteres de altura. O primeiro laço adiciona um grupo de três caracteres a **s$**, para cada uma das letras da frase. Quando não há mais letras, o computador imprime **s$**. O grupo de caracteres adicionado a **s$** varia conforme a linha impressa. Como podemos observar nas linhas 420, 450, 480 e 510, adiciona-se primeiro o cordão **a$**, depois o **b$**, o **c$**, e, finalmente, o **d$**. Cada um desses cordões é, de fato, uma matriz. Os números entre parênteses determinam qual elemento será adicionado a **s$**.

Assim que digitamos o texto e teclamos **ENTER**, o computador salta para a linha 110, que nos permite escolher a cor em que o texto será impresso. As linhas 130 e 140 se encarregam de acertar as cores e o programa prossegue com três laços **FOR...NEXT**.

A linha 160 imprime um bloco negro sempre que o caractere do texto for um espaço. Depois, o computador pula a linha 170, indo direto à 180.

A linha 170, embora pareça extremamente complicada, apenas imprime o próximo caractere gráfico da letra que está sendo desenhada. Os cálculos entre parênteses fornecem o código do caractere adequado. A primeira operação consiste na soma do número 84 ao código da cor multiplicado por 16. Segue-se a adição do resultado ao número apropriado da matriz. **L(X,Y,ASC(MID$(B$,C,1)) −65)** determina qual elemento da matriz **L** será usado nos cálculos.

Os números das linhas **DATA** — colocados depois na matriz **L** — correspondem aos códigos dos diversos caracteres gráficos (básicos, pretos e verdes) menos 100. Eles são distribuídos em grupos de doze (quatro linhas de três caracteres) para cada letra, de modo que o 13º número da lista corresponde à letra B, o 25º, ao primeiro caractere da letra C e assim por diante.

Quando o computador identifica o elemento da matriz que deve imprimir, a linha 170 usa a função **ASC** para calcular seu código e a função **MID$** para saber de qual letra da frase se trata. Os três números entre parênteses do **MID$** definem o cordão que será cortado (**B$**,em nosso caso), seu comprimento (**C**, em nosso caso) e o número de caracteres que se deve obter a partir deste ponto (1, em nosso caso).

O ponto e vírgula após a fórmula da linha 170 evita o salto de linha, fazendo com que as letras ampliadas sejam escritas uma ao lado da outra.

Na linha 180, duas instruções **NEXT** chamam os valores seguintes de **X** (que controla a impressão do caractere adequado da linha da letra) e de **C**.

A segunda parte da linha 180 imprime um número de blocos negros suficiente para que o computador passe à próxima linha e comece a montar a cadeia seguinte de caracteres gráficos — são quatro cadeias ao todo. Ao se completar o último laço, sua frase estará escrita na tela, na cor escolhida e em tamanho ampliado.

A linha 190 aguarda que outra tecla seja pressionada, para que o computador volte à linha 40 e permita a entrada de uma nova palavra.

Quando você quiser interromper o programa, voltando ao BASIC, bastará entrar uma frase "vazia" — ou seja, apertar **ENTER** antes de digitar qualquer letra — ou pressionar **BREAK**.

A introdução da palavra (com, no máximo, nove letras) é feita nas linhas 70 e 80. O laço que vai da linha 90 à 170 "desmonta" essa palavra, letra por letra, e desenha na tela cada uma das quatro camadas que formam um caractere, chamando as rotinas das linhas 190 a 220 através de um **ON × GOSUB** (linha 140). O código que será usado, calculado a partir do código ASCII do caractere, na linha 130 do programa, é armazenado em **B**.

As linhas 230 a 740 contêm os códigos gráficos que compõem as quatro camadas de cada caractere.

Os programas deste artigo mostraram duas maneiras de criar letras ampliadas. No próximo artigo, você verá como obter outro tipo de caractere e, também, como utilizar todos esses métodos em seus próprios programas.

CLASSIFICAÇÃO DOS MICROCOMPUTADORES POR LINHA

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

CLASSIFICAÇÃO DOS MICROCOMPUTADORES POR NOME

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

UM LOGOTIPO PARA CADA MODELO DE COMPUTADOR

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


Sinclair ZX-81


TRS-80


TK-2000


MSX


Spectrum


TRS-Color


Apple II

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

NO PRÓXIMO NÚMERO

PROGRAMAÇÃO BASIC
Velocidade de execução em programas BAISC. Cronometragem dos comandos. Relação entre estrutura, memória e velocidade.

PERIFÉRICOS
Como funciona uma caneta óptica. Sensibilidade. Resolução alta e baixa. Problemas de compatibilidade.

PROGRAMAÇÃO BASIC
Adaptação da escala dos caracteres para qualquer altura e largura. Como desenhar seu próprio tipo de letra.

APLICAÇÕES
Um assistente para o DOS. Como funciona o programa. Limitações.