CURSO PRÁTICO 4 DE PROGRAMAÇÃO DE COMPUTADORES

BASIC

PROGRAMAÇÃO BASIC - PROGRAMAÇÃO DE JOGOS - CÓDIGO DE MÁQUINA

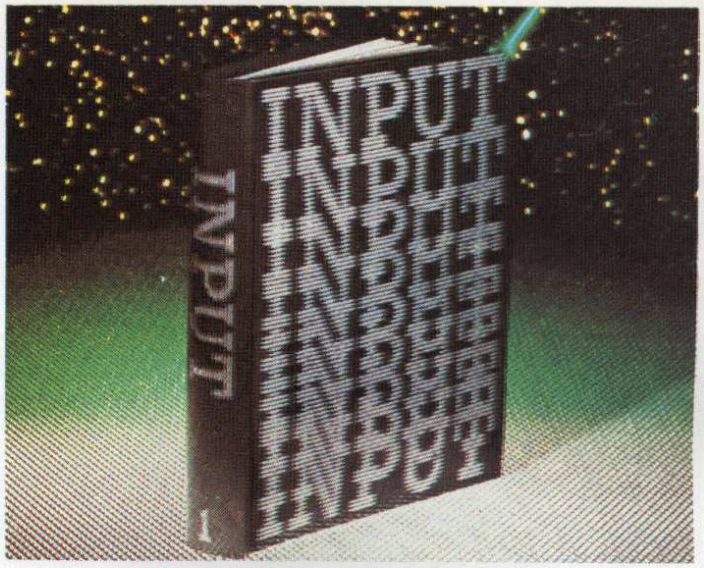
NOVA CULTURAL

Cz$ 20,00

INPUT

Vol. 1 Nº 4

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 — Distribuidor Nacional de Publicações — Números Atrasados — Estrada Velha de Osasco, 132 (Jardim Tereza) — CEP 06000 — Osasco — São Paulo. **3. Por telex** — Utilize o nº (011) 33670 ABSA. Em **Portugal**, os pedidos devem ser feitos à Distribuidora Jardim de Publicações Ltd. — Qta. Pau Varais, Azinhaga de Fetais — 2685, Camarate — Lisboa; Tel. 257-2542 — Apartado 57 — Telex 43 069 JARLIS P.
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. **Atenção**: Após seis meses do encerramento da coleção, os pedidos serão atendidos, dependendo da disponibilidade de estoque. **Obs.:** Quando pedir livros, mencione sempre o título e/ou o autor da obra, além do número da edição.

COLABORE CONOSCO

Encaminhe seus comentários, críticas, sugestões ou reclamações ao Serviço de Atendimento ao Leitor — Caixa Postal 9442, São Paulo — SP.

Editor
VICTOR CIVITA

REDAÇÃO

Diretora Editorial: Iara Rodrigues

Editor chefe: Paulo de Almeida
Editor de texto: Cláudio A.V. Cavalcanti
Editor de Arte: Eduardo Barreto
Chefe de Arte: Carlos Luiz Batista
Assistentes de Arte: Ailton Oliveira Lopes, Dilvacy M. Santos, José Maria de Oliveira, Grace A. 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
Secretário Gráfico: Antonio José Filho

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: Aluísio J. Dornellas de Barros, Maria Fernanda Sabbatini
Adaptação, programação e redação: Aluísio J. Dornellas de Barros, Marcelo R. Pires Therezo, Raul Neder Porrelli
Coordenação geral: Rejane Felizatti Sabbatini
Assistente de Arte: Dagmar Bastos Sampaio

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

PRODUÇÃO
Gerente de Produção: João Stungis
Coordenador de Impressão: Atilio Roberto Bonon
Preparador de Texto/Coordenador: Eliel Silveira Cunha

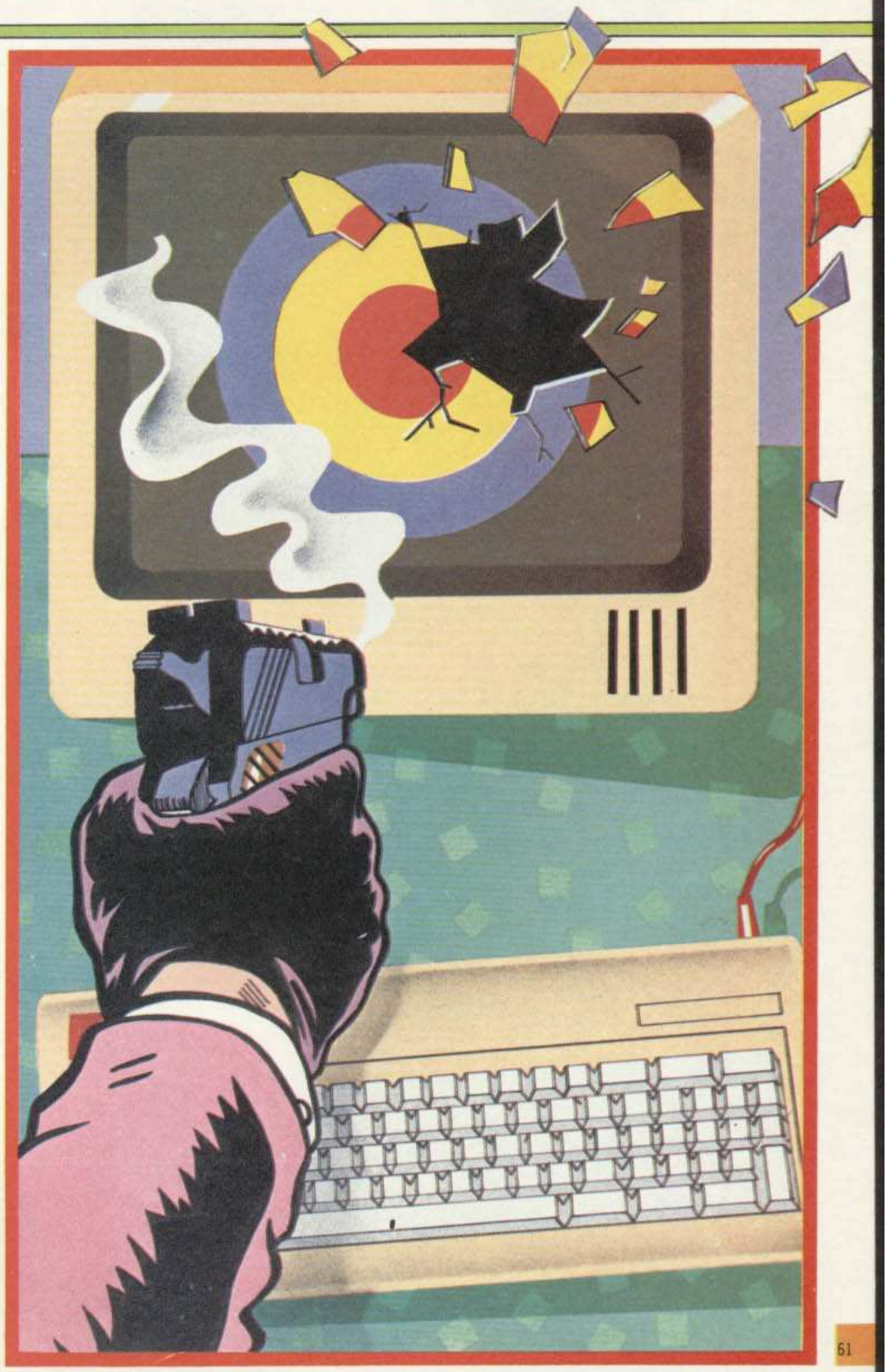
Preparadores de Texto: Ana Maria Dilguerian, Antonio Francelino de Oliveira, Karina Ap. V. Grechi, Levon Yacubian, Maria Teresa Galluzzi, Paulo Felipe Mendrone
Revisor/Coordenador: José Maria de Assis
Revisoras: Conceição Aparecida Gabriel, Isabel Leite de Camargo, Ligia Aparecida Ricetto, Maria do Carmo Leme Monteiro, Maria Luiza Simões, Maria Teresa Martins Lopes.

Esta obra foi composta na AM Produções Gráficas Ltda. e impressa na Divisão Gráfica da Editora Abril S.A.

MARQUE O TEMPO E OS PONTOS

- O JOGO DO CAMPO MINADO
- APRENDA A MONTAR UM PLACAR
- COMO INCORPORAR AO JOGO UM MEDIDOR DE TEMPO

Todo jogo de ação realiza algum tipo de contagem de pontos ou de tempo, de modo a tornar a competição mais emocionante. Nesta lição você aprenderá como fazer isso com a ajuda de programas bem simples.



Quase todos os jogos de computadores precisam de alguma forma de marcação de pontos ou de cronometragem — ou ambos. Sem eles você não poderá saber se está se saindo bem no jogo, se está melhorando, ou quem é o melhor jogador. Além disso, são esses elementos de medida de desempenho que dão graça aos jogos de computador.

Seria possível, evidentemente, obrigar alguém a sentar-se ao seu lado e ficar contando os disparos que você faz. A melhor solução, no entanto, é programar o computador para fazer isso por você. Umas poucas linhas a mais no programa darão cabo dessa tarefa, não importa a complexidade do esquema de marcação de pontos.

Da mesma forma, não existe necessidade de recorrer a um cronômetro para marcar o tempo. Como se viu nos programas para o jogo de labirinto (pág. 46 a 52), quase todos os tipos de computador têm um relógio embutido, que pode ser utilizado para melhorar seus jogos (a exceção é o Apple II, que precisa de uma placa especial).

CAMPO MINADO

Para que você possa ver como montar esquemas práticos de contagem e cronometragem, apresentamos a seguir um jogo adequado a todas as máquinas, com exceção das compatíveis com o ZX81, e ao qual acrescentaremos progressivamente todas as rotinas necessárias. Os esquemas são simples e podem ser colocados também em outros jogos.

O jogo é chamado Campo Minado, e nele você é um comandante de tanque cuja missão é resgatar os pára-quedistas que estão pousando imprudentemente em um campo minado. Por onde quer

que passe, o tanque corre o risco de detonar uma mina escondida ao acaso pelo computador. Assim como em campo de batalha real, as minas são invisíveis; portanto, mova-se com cuidado!

O tanque (apenas um caractere #, infelizmente, pois você ainda não aprendeu tudo sobre combinar gráficos e movimentos em um programa BASIC) executa movimentos por meio das seguintes teclas: Z para a esquerda, X para a direita, P para cima e L para baixo. O âmago do programa está na rotina de movimentação de um caractere na tela estudada numa lição anterior.

Quando você digitar e rodar esta seção do jogo verá que ele ainda não está completo, pois nada acontece depois que o pára-quedista é resgatado. Apenas o tanque continua a vagar, sem rumo, através da tela. O programa deve ser interrompido pressionando-se as teclas <**BREAK**>, <**ESCAPE**> ou <**STOP**>, ou você terá que esperar até que o tanque detone uma mina escondida ao acaso. Nesse momento, o jogo terminará. Mas não se preocupe, essa situação melhorará tão logo você acrescente as rotinas de contagem e cronometragem que se seguem.

T T

```
50 LET PO=210
60 CLS
70 PRINT @256,STRING$(32,"-")
90 LET X=RND(256)-1
110 IF X<>PO THEN PRINT @X,"O";
ELSE GOTO 90
120 PRINT @PO,"#";
130 LET LP=PO
140 IF PEEK(340)=247 THEN LET P
O=PO-1:GOTO 190
150 IF PEEK(338)=247 THEN LET P
O=PO+1:GOTO 210
160 IF PEEK(338)=251 THEN LET P
O=PO-32:GOTO 220
170 IF PEEK(342)=253 THEN LET P
O=PO+32:GOTO 220
180 GOTO 140
190 IF (LP AND 31)=0 THEN LET P
O=LP
200 GOTO 220
210 IF (PO AND 31)=0 THEN LET P
O=LP
220 IF PO>255 OR PO<0 THEN LET
PO=LP:GOTO 140
230 PRINT @LP," ";
240 PRINT @PO,"#";
250 LET M=RND(256)-1
270 IF M=PO THEN PRINT @PO," ":
PRINT @130,"BOOM!!! VOCE ACERTO
U UMA MINA!":STOP
310 GOTO 130
```

O movimento do tanque pela tela é controlado pelas linhas 140 a 170, que verificam quais foram as teclas pressionadas (veja a pág. 33, se não se lembra de como isso é feito). A linha 220 testa IF PO > 255... para manter o tanque no alto da parte central da tela. Ora, a última locação de tela no alto da parte central do vídeo é a 255; assim o **IF...** impede que o tanque avance abaixo da linha pontilhada, desenhada pela linha 70.

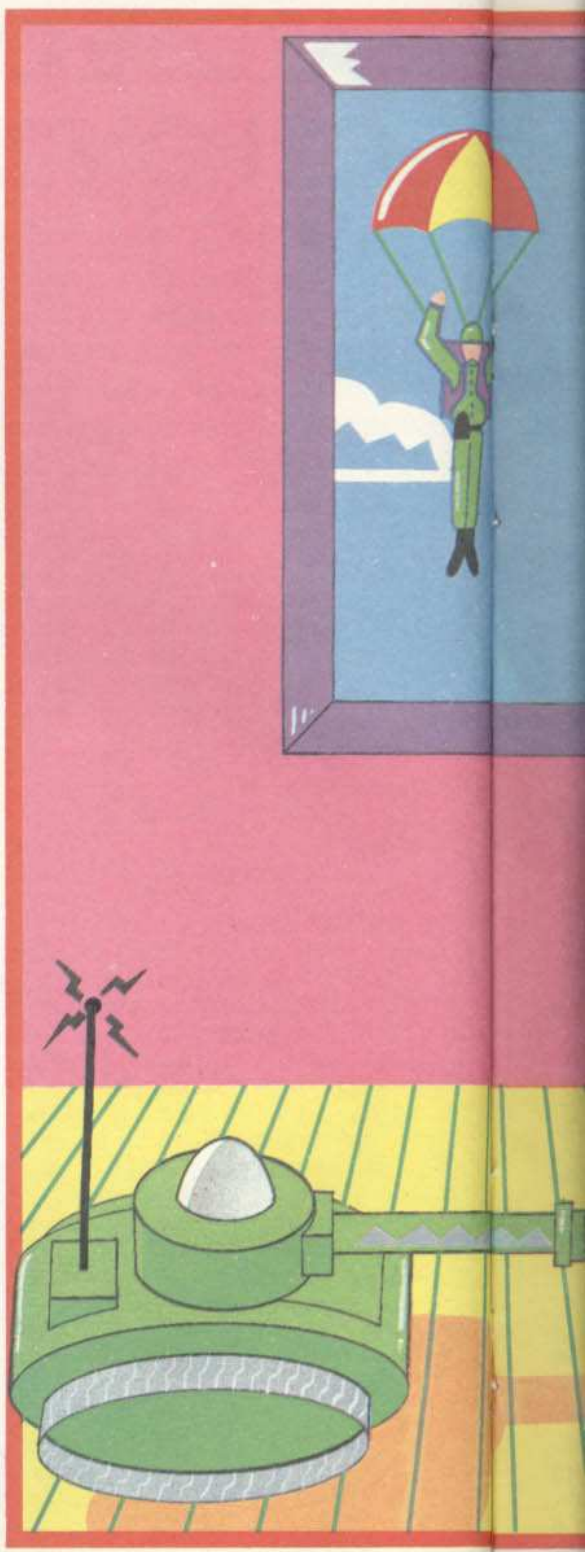
Esta última traça a linha através da tela, utilizando uma nova função — **STRING$**. O **STRING$(32,"—")**, como aparece na linha 70, diz ao TRS-Color para desenhar um cordão (*string*, em inglês) de 32 travessões (se você quisesse dez pontos de interrogação deveria utilizar **STRING$(10,"?")**, e assim por diante). O lugar em que o pára-quedista cai é escolhido por um número aleatório na linha 90, e a linha 110 exibe o pára-quedas na tela se a posição escolhida já não estiver ocupada pelo tanque. Se estiver, a linha 90 escolherá outra posição.

Finalmente, a posição da mina escondida é escolhida pela linha 250. A linha 270 checa então se o tanque e a mina estão na mesma locação de tela. Se estiverem, a mensagem de explosão será exibida, e o programa parará.

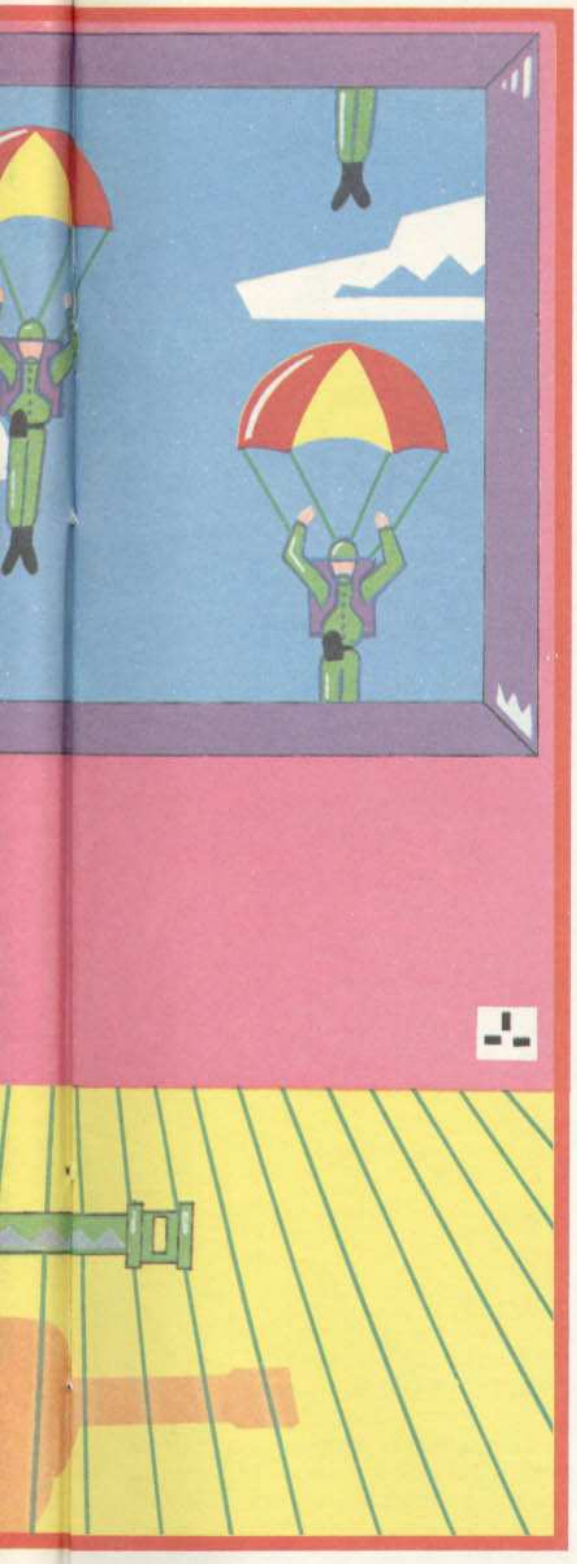
S

```
 50 LET tx=16: LET ty=5
 60 CLS
 70 PRINT AT 11,0;"----------
 ----------"
 90 LET px=INT (RND*30)+1
100 LET py=INT (RND*10)
110 IF px=tx AND py=ty THEN
GOTO 90
120 PRINT AT py,px;"O";AT ty,
 tx;"#"
130 LET txx=tx: LET tyy=ty
140 LET a$=INKEY$
145 IF a$="w" THEN  LET ty=ty-
1
150 IF a$="z" THEN  LET ty=ty+
1
160 IF a$="a" THEN  LET tx=tx-
1
170 IF a$="s" THEN  LET tx=tx+
1
190 IF ty<0 OR ty>10 THEN  LET
 ty=tyy
200 IF tx<0 OR tx>30 THEN  LET
 tx=txx
230 PRINT AT tyy,txx;" "
240 PRINT AT ty,tx;"#"
250 LET mx=INT (RND*30)+1
260 LET my=INT (RND*10)
270 IF mx=tx AND my=ty THEN
PRINT AT  my,mx;" ": PRINT AT
8,0;"BOOM!!! - Voce acertou um
a mina!":   GOTO 330
310 GOTO 130
```

No programa do Spectrum, a tela é dividida ao meio pela linha 70, que imprime uma série de 32 travessões ao longo do vídeo. As linhas 145 a 170 controlam os movimentos do tanque. Uma explicação completa desse processo está na página 31.



As linhas 190 e 200 impedem que o tanque saia da área situada acima da linha (pontilhada) traçada pela linha 70, e impede que ele desapareça da tela. A posição de queda dos pára-quedistas é



escolhida aleatoriamente pelas linhas 90 e 100. A linha 110 verifica se o tanque e um dos pára-quedistas ocupam a mesma locação de tela. Se isso ocorrer, uma nova posição de queda será escolhida, indo-se para a linha 90. As linhas 250 e 260 escolhem um lugar para a mina. A linha 270 compara a posição da mina com a do tanque. Se eles estiverem no mesmo lugar, o tanque será exibido na tela e aparecerá uma mensagem de explosão. Neste caso, o programa parará.

```
50 LET PO=220
60 CLS
70 LOCATE 0,12:PRINT STRING$(39
,"_")
90 LET X=INT(RND(1)*480)
110 IF X<>PO THEN VPOKE BASE(0)
+X,ASC("O") ELSE GOTO 90
120 VPOKE BASE(0)+PO,ASC("▮")
130 LET LP=PO
135 LET K$=INKEY$:IF K$="" THEN
 GOTO 135
140 IF K$=CHR$(29) THEN LET PO=
PO-1:GOTO 190
150 IF K$=CHR$(28) THEN LET PO=
PO+1:GOTO 210
160 IF K$=CHR$(30) THEN LET PO=
PO-40:GOTO 220
170 IF K$=CHR$(31) THEN LET PO=
PO+40:GOTO 220
180 GOTO 140
190 IF (LP/40-INT(LP/40))=0 THE
N LET PO=LP
200 GOTO 220
210 IF (PO/40-INT(PO/40))=0 THE
N LET PO=LP
220 IF PO>479 OR PO<0 THEN PO=L
P:GOTO 130
230 VPOKE BASE(0)+LP,ASC(" ")
240 VPOKE BASE(0)+PO,ASC("▮")
250 LET M=INT(RND(1)*480)
270 IF M=PO THEN VPOKE BASE(0)+
PO,ASC(" "):LOCATE 5,4:PRINT "B
OOM!!-Você acertou numa mina":E
ND
310 GOTO 130
```

O movimento do tanque pela tela é, como no caso anterior, controlado pelas linhas 140 a 170, que verificam quais foram as teclas pressionadas. A linha 220 testa **IF PO > 479...** a fim de restringir o campo de ação do tanque à parte alta do meio do vídeo. Agora, porém, a última locação de tela nessa posição é a 479; assim, **O IF...** impede que ela avance abaixo da linha pontilhada desenhada pela linha 70.

Também como no caso anterior a função **STRING$** é empregada pela linha 70 para traçar a linha ao longo da tela. O **STRING$(39, "—")**, como aparece na linha 70, diz simplesmente ao MSX para desenhar um cordão de 39 travessões (veja na pág. anterior, abaixo do programa para o TRS80 e o TRS-Color, como fazer para obter dez pontos de interrogação).

O lugar em que o pára-quedista cai é escolhido por um número na linha 90; a linha 110 mostra o pára-quedas na tela se a posição escolhida já não estiver ocupada pelo tanque. Se estiver, a linha 90 escolherá outra posição. Não se preocupe em entender, por enquanto, o comando **VPOKE BASE**, na linha 110. Ele serve para colocar um caractere comum (ou gráfico) em uma certa posição da tela, e é usado aqui por ser mais rápido que o **LOCATE**.

A linha 250, escolhe a posição da mina escondida. A linha 270 verifica se o tanque e a mina estão na mesma locação da tela. Sendo positivo, a mensagem de explosão é exibida, e o programa termina.

```
50  LET TX = 20: LET TY = 10
60  HOME
70  FOR I = 1 TO 40
74  HTAB I: VTAB 13: PRINT "-";
78  NEXT I
90  LET PX =  INT ( RND (1) * 4
0) + 1
100  LET PY =  INT ( RND (1) *
12) + 1
110  IF PX <  > TX AND PY <  >
TY THEN  HTAB PX: VTAB PY: PRIN
T "O";: GOTO 120
111  GOTO 90
120  HTAB TX: VTAB TY: PRINT "▮
";
130  LET LX = TX: LET LY = TY
140  LET K =  PEEK ( - 16384):
POKE  - 16368,0
150  IF K = 208 THEN  LET TY =
TY - 1
160  IF K = 204 THEN  LET TY =
TY + 1
170  IF K = 218 THEN  LET TX =
TX - 1
175  IF K = 216 THEN  LET TX =
TX + 1
190  IF TX < 1 OR TX > 40 THEN
 LET TX = LX
200  IF TY < 1 OR TY > 12 THEN
 LET TY = LY
230  HTAB LX: VTAB LY: PRINT "
";
240  HTAB TX: VTAB TY: PRINT "▮
";
250  LET MX =  INT ( RND (1) *
40) + 1
260  LET MY =  INT ( RND (1) *
12) + 1
270  IF MX = TX AND MY = TY THE
N  HTAB MX: VTAB MY: PRINT " ";
: HTAB 7: VTAB 7: PRINT "BOOM!!
-Voce acertou uma mina";: GOTO
330
310  GOTO 130
```

O movimento do tanque pela tela é controlado pelas linhas 140 a 170, que checam as teclas pressionadas (lembre-se da explicação da pág. 33). Os comandos **PEEK** e **POKE** da linha 140 são necessários porque o Apple não tem uma função em BASIC, como o **INKEY$**, que serve para fazer a varredura do teclado. A fim de restringir as manobras do tanque, as linhas 190 e 200 testam se as suas coordenadas (TX e TY) não excedem 40 e 12 posições, respectivamen-

te (margem direita da tela e posição de linha acima da linha pontilhada).

As linhas 70 a 78 desenham uma linha pontilhada ao longo da tela.

Um número aleatório nas linhas 90 e 100 escolhe o lugar em que o pára-quedista cai; a linha 110 exibe o pára-quedas na tela se a posição escolhida já não estiver ocupada pelo tanque. Se estiver, então a linha 90 escolherá uma outra posição. Neste caso, a posição da mina escondida é escolhida pelas linhas 250 e 260. A linha 270 então verifica se o tanque e a mina estão na mesma locação da tela. Se eles estiverem, será exibida uma mensagem de explosão, e o programa se deterá.

MARQUE PONTOS

Em jogos do tipo videogame, a contagem normalmente é aumentada quando as posições de tela de dois objetos são iguais. Os objetos podem ser um míssil e, o seu alvo, o "come-come" e um monstrinho, um tanque e um pára-quedista, um cavalo e o poste de chegada, ou qualquer outra coisa.

Assim, acrescente estas linhas ao seu programa, para ver como um mecanismo de contagem funciona na prática.

```
40 LET S=0
280 IF X=PO THEN LET S=S+1:GOTO
 90
330 PRINT @295,S;"PARAQUEDISTAS
";
```

```
 40 LET s=0
280 IF px=tx AND py=ty THEN
 LET s=s+1 : GOTO 90
330 PRINT AT 14,8;s;" PARAQUED
ISTAS"
```

```
40 LET S=0
280 IF X=PO THEN LET S=S+1:GOTO
 90
330 LOCATE 8,16:PRINT S;" paraq
uedistas"
```

```
40  LET S = 0
280  IF PX = TX AND PY = TY THE
N  LET S = S + 1: GOTO 90
330  HTAB 12: VTAB 15: PRINT S;
" PARAQUEDISTAS";
```

Você precisará substituir o **STOP** ou **END** na linha 270 por **GOTO 330**. A linha 270 deverá ficar assim, agora:

```
270 IF M=PO THEN PRINT @PO," ":
PRINT @
130,"BOOM!!! VOCE ACERTOU UMA M
INA!":G OTO 330
```

```
270 IF mx=tx AND my=ty THEN
PRINT AT my,mx;"!": PRINT AT 8
,3;"BOOM!!! - VOCE ACERTOU UMA
 MINA!": GOTO 310
```

```
270 IF M=PO THEN VPOKE BASE(0)+
PO,ASC(" "):LOCATE 5,4:PRINT "B
OOM!!-Você acertou numa mina":G
OTO 330
```

```
270  IF MX = TX AND MY = TY THE
N  HTAB MX: VTAB MY: PRINT " ";
: HTAB 7: VTAB 7: PRINT "BOOM!!
-VOCE ACERTOU NUMA MINA";: GOTO
 330
```

A linha 280 é a mais importante. Ela checa se o tanque e o pára-quedista estão na mesma locação da tela. Se estiverem, então o placar será incrementado de 1 ponto.

A linha 40 volta o placar para zero, antes de o jogo começar, e a linha 330 exibe a contagem. Ao modificar-se o final da linha 270, o computador exibe a contagem depois que a mina tiver sido detonada.

O jogo funciona assim: o jogador recebe continuamente pára-quedistas para resgatar. Quando um é resgatado, outro pára-quedista cai do céu.

O jogo termina quando o tanque em movimento passa sobre a mina, provocando uma explosão.

RECORDES

Acrescentar um dispositivo de registro do recorde de pontos no jogo não é difícil. Tudo o que você precisa é de uma variável para armazenar o recorde, e algum modo de atualizá-la assim que este for quebrado. Fora isso, é necessária ainda uma rotina de exibição do recorde. Estas são as linhas que você deve acrescentar para aumentar a capacidade de registrar e exibir recordes no jogo "Campo Minado":

```
30 LET HS=0
350 IF S>HS THEN LET HS=S
370 PRINT @424,"RECORDE=";HS;
```

```
 30 LET hs=0
350 IF s>hs THEN  LET hs=s
370 PRINT AT 18,8;"RECORDE=";hs
```

```
30 LET HS=0
350 IF S>HS THEN LET HS=S
370 LOCATE 12,20:PRINT "Recorde
: ";HS
```

```
30  LET HS = 0
350  IF S > HS THEN  LET HS = S
370  HTAB 12: VTAB 20: PRINT "R
ecorde=";HS;
```

Primeiro o programa deve introduzir a variável contendo o recorde com um valor o mais baixo possível. Assim, a linha 30 iguala HS a zero. Quando o jogo termina, a linha 350 compara o último escore (S) com o recorde (HS). Se o escore for maior do que o recorde, então HS será atualizada, recebendo o valor em S. Finalmente, a linha 370 exibe o recorde na tela.

Pode-se pensar que essas linhas serão suficientes para incorporar sempre novos recordes ao jogo. Infelizmente isso

não é verdade. Sempre que o programa é rodado, o computador zera automaticamente o valor de HS registrado no jogo anterior, assim como os valores de todas as outras variáveis. Para manter o valor de HS entre uma partida e outra do jogo, você precisa acrescentar linhas de programa para perguntar ao jogador se ele quer jogar outra vez, como as que foram usadas em jogos anteriores.

```
390 FOR F=1 TO 1000:NEXT F
400 PRINT @130
410 PRINT @135,"OUTRA VEZ(S/N)?
"
420 LET K$=INKEY:IF K$="" THEN
GOTO 420
430 IF K$="S" THEN GOTO 40
440 IF K$="N" THEN END
450 GOTO 420
```

```
390 PAUSE 100
400 PRINT AT 8,3;TAB 31
410 PRINT AT 8,7;"Quer jogar d
e novo (S/N)?"
420 LET a$=INKEY$
430 IF a$="s" THEN  GOTO 40
440 IF a$="n" THEN  STOP
450 GOTO 420
```

```
390 FOR F=1 TO 1000:NEXT F
400 LOCATE 0,4:PRINT STRING$(39
," ")
410 PRINT TAB(10);"Outra vez? (
S/N)"
420 LET K$=INKEY$:IF K$="" THEN
 GOTO 420
430 IF K$="S" OR K$="s" THEN GO
TO 40
440 IF K$="N" OR K$="n" THEN EN
D
450 GOTO 420
```

```
390  FOR F = 1 TO 1000: NEXT F
400  FOR I = 1 TO 40: HTAB I: V
TAB 7: PRINT " ";: NEXT I
410  HTAB 11: VTAB 8: PRINT "Ou
tra vez? (S/N)";
420  GET A$
430  IF A$ = "S" THEN  GOTO 40
440  IF A$ = "N" THEN  END
450  GOTO 420
```

Isto é o que a linha extra faz:

Cria-se um pequeno retardo de tempo, introduzido pelo laço **FOR...NEXT** na linha 390, nos programas para todos os computadores, exceto para o Spectrum, que usa o comando **PAUSE**. A linha 400 limpa uma parte da tela, de modo a prepará-la para que a mensagem "MAIS UMA VEZ? (S/N)" seja impressa pela linha 410.

A rotina que pergunta ao jogador se quer jogar de novo e colhe a sua resposta está nas linhas 410 a 450 (a linha 430 reiniciará o programa na linha 40 se S for pressionado, e a linha 440 parará o programa se N for pressionado). A linha 450 assegura que qualquer outra tecla será ignorada. Como agora não existe necessidade de digitar **RUN** cada vez que você desejar jogar de novo, o valor de HS é preservado.

COMO FAZER A CRONOMETRAGEM

Do jeito que está, o jogo depende muito da sorte — o jogador simplesmente continua andando com o tanque, até encontrar uma mina escondida.

Podemos, entretanto, introduzir novos elementos no jogo, transformando-o, se quisermos, em uma corrida contra o relógio. Um desses elementos poderia ser, por exemplo, um dispositivo interno para marcar o tempo gasto em resgatar dez pára-quedistas.

```
80 TIMER=0
290 IF S<10 AND X=PO THEN GOTO
90
300 IF S=10 THEN GOTO 320
320 LET T=TIMER
330 PRINT @295,S;"PARAQUEDISTAS
";
340 IF S=10 THEN PRINT @327,"EM
";T/50;" SEGUNDOS"
```

```
 10 LET t=0
 80 POKE 23672,0: POKE 23673,0
290 IF s<10 AND px=tx AND py=t
y THEN  GOTO 90
300 IF s=10 THEN  GOTO 320
320 LET t=PEEK 23672+256*PEEK
23673
340 IF s=10 THEN  PRINT AT 16,
8;"EM ";t/50;" SEGUNDOS"
```

```
80 TIME=0
290 IF S<10 AND X=PO THEN GOTO
90
300 IF S=10 THEN GOTO 320
320 LET T=TIME
340 IF S=10 THEN LOCATE 10,18:P
RINT "em ";T/50;" segundos"
```

O relógio existente em cada um dos tipos de micros acima corre durante todo o tempo em que o computador estiver ligado.

Para iniciar a cronometragem do jogo, você precisa apenas zerar uma va-

riável de tempo. A linha que coloca em zero o cronômetro é a linha 80 — você notará que para zerar o cronômetro do TRS-Color e do MSX basta fazer **TIMER** = *O*. No Spectrum é mais complicado: a linha do programa zera duas locações específicas na memória de trabalho do micro, através de **POKE 23 672,0** e **POKE 23 673,0**.

O relógio é "parado" pela linha 320. Na verdade, ele não se detém; assim, você simplesmente instrui a máquina a "lembrar-se" do valor de tempo marcado em um determinado momento — quando dois objetos coincidem na tela, por exemplo.

Nesses programas, a variável que contém a leitura do relógio é chamada de **T** — no MSX e no TRS-Color, isso é feito com a linha **LET T = TIMER**, e no Spectrum, com **LET T = PEEK 23672 + 256*PEEK 23673**.

O comando **PEEK**, como aprenderemos em uma lição posterior, "olha" os valores numéricos contidos nas mesmas locações de memória zeradas no início do programa.

A locação 23672, como qualquer locação de memória em um micro de 8 bits, pode conter números inteiros entre 0 e 255. A locação 23673 é incrementada de 1, toda vez que esse número for excedido na locação 23672.

Assim, para calcular o valor total do tempo em que o cronômetro do Spectrum esteve correndo, você deve multiplicar o valor da locação 23673 por 256 e acrescentar o valor armazenado na locação 23672. Esta é a razão da expressão numérica na linha 90, na versão para o Spectrum.

O relógio deve ser parado quando o jogador tiver resgatado dez pára-quedistas; assim, a linha 300 checa se este valor foi atingido e salta para a linha 320, que "pára" o cronômetro.

Se o total de resgatados for menor que dez, a linha 290 manda outro pára-quedista para ser salvo pelo jogador.

A linha 340 só exibirá o tempo total de jogo se os dez pára-quedistas tiverem sido recuperados. A leitura do cronômetro é dividida por 50 no programa. Assim, o tempo exibido será expresso, aproximadamente, em segundos.

```
80  LET T = 0
290  IF S < 0 AND PX = TX AND P
Y = TY THEN  GOTO 90
300  IF S = 10 THEN  GOTO 330
340  IF S = 10 THEN  HTAB 12: V
TAB 16: PRINT "em ";T / 6;" seg
undos";
```

A contagem de tempo na versão do programa para os microcomputadores da linha Apple é feita de forma inteiramente diferente das outras máquinas, pois eles não têm relógio interno.

Neste caso, definimos uma variável denominada **TIME**, que conterá o valor cronometrado, e uma variável **T**, que é incrementada de 1 a cada ciclo de varredura do teclado. Este é um truque muito fácil de ser implementado, e que serve para qualquer computador que não tenha relógio interno.

A variável relógio é zerada na linha 80 e recebe o valor do tempo gasto na linha 320. Para calcular o valor do tempo em segundos, divide-se o conteúdo de **T** por 6.

RECORDE DE TEMPO

Da mesma forma que adicionamos o registro de recordes de pontos ao programa anterior, poderíamos agora acrescentar um registro do recorde de tempo (no caso, o menor tempo gasto para salvar dez pára-quedistas). O princípio geral de registro de recordes de tempo pode ser aplicado em muitos outros jogos.

Estas são as linhas a serem acrescentadas ao programa:

```
20 LET LT=999999
360 IF T<LT AND S=10 THEN LET L
T=T
380 PRINT @452,"MENOR TEMPO=";L
T/50;" SEGUNDOS"
```

```
 20 LET lt=999999
360 IF t<lt AND s=10 THEN  LET
 lt=t
380 PRINT AT 21,4;"MENOR TEMPO
=";lt/50;" SEGUNDOS"
```

```
20 LET LT=999999!
360 IF T<LT AND S=10 THEN LET L
T=T
380 LOCATE 6,22:PRINT "Melhor t
empo: ";LT/50;" segundos"
```

```
20  LET LT = 99999
360  IF S = 10 AND T < LT THEN
LT = T
380  HTAB 12: VTAB 21: PRINT "e
m ";LT / 6;" segundos";
```

Inicialmente, devemos inicializar a variável usada para armazenar o menor tempo (**LT**). Esse valor, ao contrário do usado para introduzir a variável de recorde de pontos, deve ser bem alto, no começo. A linha 20, incorporada ao programa, iguala a variável **LT** a 999999.

A linha 360 compara o último tem-

po de jogo registrado com o menor tempo, armazenado em **LT**. Se aquele for menor ainda do que o último recorde de tempo, passará a ser o novo recorde, transferindo-se o seu valor para **LT**.

Finalmente, a linha 380 exibe o recorde de tempo, em segundos. O valor da variável **LT** é dividido por 50 ou por 100, para que isto se torne possível.

TEMPO DE REAÇÃO

Até agora, você viu como medir o tempo com auxílio do relógio interno do computador, dentro de um programa que checa, por exemplo, as posições de dois objetos na tela.

Outra aplicação interessante para esse modo de aferição de tempo é utilizar o teclado para medir a velocidade de reação do jogador a algum tipo de ação contrária.

Você já sabe como utilizar a função **INKEY$** (em caso de dúvida, veja na pág. 28). Ela pode servir não só para controlar a movimentação de objetos na tela, mas também para iniciar e parar contagens de tempo.

Eis aqui um jogo da variedade "rápido-no-gatilho", que ilustra muito bem esse tipo de aplicação. O programa mostra na tela: "SAQUE!", e o jogador deve reagir o mais rapidamente que puder, pressionando qualquer tecla no teclado.

```
20 CLS
30 LET N=RND(900)
40 FOR F=0 TO N
50 NEXT F
60 PRINT @269,"SAQUE!!"
70 TIMER=0
80 IF INKEY$=""THEN GOTO 80
90 LET T=TIMER
100 PRINT @269,"BANG!!!"
110 FOR F=1 TO 300
120 NEXT F
130 LET M=RND(25)
140 IF T<M THEN PRINT @264,"VOC
E SOBREVIVEU"
150 IF T>M THEN PRINT @266,"VOC
E ESTA MORTO"
160 IF T=M THEN PRINT @264,"VOC
ES ESTAO AMBOS MORTOS"
```

```
20 CLS
30 LET N=INT (RND*400)+1
40 PAUSE n
60 PRINT AT 11,14;"SAQUE!!"
70 POKE 23672,0: POKE 23673,0
80 IF INKEY$="" THEN  GOTO 80
90 LET T=PEEK 23672+256*PEEK
23673
100 PRINT AT 11,14;"BANG!!"
110 PAUSE 50
130 LET m=INT (RND*35)+1
140 IF t<m THEN  PRINT AT 11,9
;"VOCE SOBREVIVEU"
150 IF t>m THEN  PRINT AT 11,9
;"VOCE ESTA MORTO"
160 IF t=m THEN  PRINT AT 11,9
;"VOCES ESTAO AMBOS MORTOS"
```

```
20 CLS
30 LET N=RND(1)*900
40 FOR F=1 TO N
50 NEXT F
60 LOCATE 17,11:PRINT "SAQUE!"
70 TIME=0
80 IF INKEY$="" THEN 80
90 LET T=TIME/6
100 LOCATE 17,11:PRINT "BANG!!"
110 FOR F=1 TO 300
120 NEXT F
130 LET M=RND(1)*25
140 IF T<M THEN LOCATE 13,11:PR
INT "Você sobreviveu"
150 IF T>M THEN LOCATE 13,11:PR
INT "Você está morto"
160 IF T=M THEN LOCATE 13,11:PR
INT "Ambos morreram"
```

```
20   HOME
30   LET N =  RND (1) * 2000
40   FOR F = 1 TO N
50   NEXT F
60   HTAB 17: VTAB 13: PRINT "SA
QUE!!"
70   LET T = 0
80 X =  PEEK ( - 16384): POKE
- 16368,0
81   IF X > 127 THEN  GOTO 100
85   LET T = T + 1
90   GOTO 80
100  HTAB 17: VTAB 13: PRINT "B
ANG!!"
110  FOR F = 1 TO 1000
120  NEXT F
130  LET M =  RND (1) * 20
140  IF T < M THEN  HTAB 12: VT
AB 13: PRINT "VOCE SOBREVIVEU"
150  IF T > M THEN  HTAB 12: VT
AB 13: PRINT "VOCE ESTA MORTO"
160  IF T = M THEN  HTAB 12: VT
AB 13: PRINT "AMBOS MORRERAM "
```

O programa é muito simples. Após a linha 20 ter limpado a tela, uma pausa aleatória é introduzida pelas linhas 30 a 50. A linha 60 exibe "SAQUE!", e o cronômetro é acionado imediatamente pela linha 70. A linha 80 faz com que a máquina espere até que uma tecla seja pressionada. A linha é exatamente igual à que foi utilizada nos programas de jogos com controle pelo teclado (veja pág. 28).

Assim que qualquer tecla tenha sido pressionada, o programa continua para a linha 90, que pára efetivamente o cronômetro, armazenando o seu valor naquele momento na variável **T**. A linha 100 exibe "BANG!!".

Existe uma pausa introduzida pelas linhas 110 e 120 (a linha 110 somente no Spectrum) antes que um novo retardo inicial de tempo seja sorteado pelo programa. Isto é feito pela linha 130 do programa.

Em seguida, o computador sorteia uma variável aleatória **M**, que conterá o tempo levado pelo computador para "sacar" sua arma. As linhas 140 a 160 comparam os valores **T** (do jogador) e **M** (da máquina) e declaram quem foi o vencedor do duelo (ou se ambos os duelistas morreram!).

Existe algum limite máximo para o período de medida de tempo?

Sim, existe — embora normalmente seja tão longo que, na prática, não faz diferença. O relógio interno da maioria dos computadores domésticos corre a velocidades semelhantes. O fator limitante, portanto, é o número máximo de impulsos de relógio que o computador pode armazenar. As linhas TRS-Color e MSX utilizam dois bytes; por isso podem contar até 65 535 impulsos de relógio, um a cada 1/50 de segundo. Isso dá uma capacidade máxima de temporização de cerca de 22 minutos. Já os micros da linha Spectrum usam 3 bytes, o que lhes dá cerca de *quatro* dias de medida contínua de tempo!

Como funciona o cronômetro interno de um microcomputador?

Todo microcomputador tem um dispositivo interno para sincronização das atividades da Unidade Central de Processamento, que é chamado de relógio ou *clock*. Esse relógio interno tem por função gerar pulsos elétricos repetidos a uma freqüência constante para cada micro: essa freqüência é chamada de *velocidade de relógio*, e seu valor varia conforme a marca do computador.

Por exemplo, o Apple II tem relógio de 1 MHz (megahertz, ou seja, um milhão de "batidas" de relógio por segundo); o TRS-80 tem 2,7 MHz, e assim por diante. Mas esse temporizador interno não pode ser aproveitado para medir o tempo como um verdadeiro cronômetro.

ORGANIZE AS SUAS COLEÇÕES (1)

Seu caderninho de endereços está uma bagunça? Sua coleção de discos tem mais problemas do que discos? Eis aqui um programa que o ajudará a colocar ordem nesse caos.

As pessoas que nunca utilizaram um computador pessoal sempre perguntam que tipo de aplicações domésticas ele poderia ter. As respostas dadas, porém, raramente são convincentes.

Neste artigo apresentamos um programa que pode ser útil para muitas pessoas. Ele é um programa de *banco de dados*, ou seja, de organização e recuperação de qualquer tipo de informação que possa ser fichada. É um sistema de arquivamento tão flexível que pode encontrar dezenas de aplicações no seu dia-a-dia. Ele é muito útil, por exemplo, para armazenar os nomes e os endereços de amigos ou membros de um clube, ou tomar notas dos aniversários da família e dos amigos, ou armazenar detalhes sobre suas coleções de moedas, borboletas, discos ou receitas, ou até mesmo organizar melhor a sua crescente coleção de jogos de computador.

O único limite para o que se pode fazer com esse programa está no tamanho da memória RAM do seu micro. Na maioria das vezes, o mínimo necessário para aplicações práticas é uma máquina com 32K.

Seja como for, é necessário lembrar sempre o seguinte: como as memórias dos computadores domésticos são pequenas, em comparação com as das máquinas profissionais, quanto menos volumosa a informação a ser mantida pelo computador, melhor.

O MENU PRINCIPAL

Assim que você rodar o programa pela primeira vez, através do comando **RUN**, o menu principal será imediatamente impresso na tela. Ele consiste numa lista de coisas que se pode fazer com o arquivo, como "entrar um registro", por exemplo, ou "pesquisar o arquivo".

ABRA SEU ARQUIVO

Os computadores necessitam de detalhes precisos do que se quer, antes de qualquer operação. Para abrir um novo arquivo é necessário primeiro dizer ao computador o número de registros que se quer e a extensão máxima que cada registro pode ter. "ABRIR UM ARQUIVO" é a opção 1 do menu principal; assim, para selecioná-la, você deve pressionar a tecla 1: as palavras "VOCÊ TEM CERTEZA?" aparecerão na tela. Essa reação da máquina é uma precaução contra o pressionamento acidental da tecla 1, pois chamar a rotina de "ABRIR UM ARQUIVO" quando o sistema de arquivamento já estiver armazenando dados pode comprometer todo o trabalho.

Se você tiver certeza de que quer abrir um novo arquivo, pressione S. Mas, se o arquivo já estiver armazenando dados e você não deseja apagá-los, pressione qualquer outra tecla, e o programa retornará ao menu principal.

O TAMANHO DOS CAMPOS

Uma vez que você tenha pressionado o S para continuar, o computador perguntará quantos campos você quer. "Campos" são os itens de informação que você quer armazenados em cada registro. Por exemplo, se você quer registrar dados sobre os seus amigos, poderiam ser definidos os seguintes campos: *nome*, *endereço*, *cidade* e *número do telefone* — quatro ao todo.

Este programa só permite um número máximo de oito campos para um registro individual; do contrário, ele não poderia exibi-los, todos, na tela ao mesmo tempo. Uma vez fornecido o número de campos a pergunta seguinte do computador será: "Qual o nome do campo 1?". (No exemplo acima, a resposta poderia ser NOME.)

A seguir, o programa pede a extensão do primeiro campo, isto é, o número máximo de caracteres que ele pode conter.

A extensão máxima para qualquer campo permitida no programa é de dezenove caracteres. Isso significa que, se a informação que se quer arquivar — um endereço, por exemplo — não couber nessa extensão, é preciso dividir o campo em duas ou mais partes. No caso de um endereço isso pode ser feito por meio da definição de campos separados para a rua, número da casa, cidade, código postal, etc.

De posse da informação sobre o primeiro campo, o computador fará a mesma pergunta sobre o segundo campo, o terceiro, e assim por diante. Obviamente, quanto menores forem os nomes dos campos e os números dos caracteres em cada campo, mais registros poderão ser retidos no arquivo.

Isto feito, o computador calculará para quantos registros ele tem espaço na memória. Esse número será exibido na tela. No programa para o TK-90X, você deve especificar quantos registros quer armazenar, evitando que o computador gaste tempo excessivo gravando ou lendo longos arquivos da fita.

■ – MANTENHA EM ORDEM AS
■ SUAS COLEÇÕES
■ – UM PROGRAMA PRÁTICO DE
■ ARQUIVAMENTO DISPONÍVEL
■ – UTILIZE EFICIENTEMENTE A
■ MEMÓRIA DISPONÍVEL
– ESTABELEÇA UM NOVO ARQUIVO
■ – REVEJA SEU ARQUIVO
■ – COMO ARMAZENAR E
■ RECUPERAR INFORMAÇÕES

COMO ENTRAR UM REGISTRO

Terminado o processo de criação de um novo arquivo, o programa trará de volta o menu principal, onde você deve selecionar a opção 2 (tecla 2), para iniciar a entrada da informação em seus registros.

No alto da tela, o computador manterá uma contagem dos registros que você já entrou, juntamente com o espaço total disponível na memória. Essa linha de informação dirá, por exemplo: "Você utilizou 10 dos 100 registros".

Logo abaixo disso, na tela, serão exibidos os nomes dos campos. O cursor aparecerá na última linha do vídeo, de tal modo que tudo que você escrever pelo teclado será registrado no campo exibido. Lembre-se de manter a informação digitada tão curta quanto possível e dentro da extensão máxima de caracteres que você estabeleceu para cada campo.

Quando você pressionar a tecla <**ENTER**> ou <**RETURN**>, a informação digitada será impressa próxima ao nome do campo. A linha inferior da tela será limpa e ficará pronta para o próximo campo. Esse procedimento começa com o primeiro campo no alto da tela e prossegue por toda a tela cada vez que você digita uma informação e pressiona <**ENTER**> ou <**RETURN**>. Quando você tiver entrado o último campo do registro, o computador irá para o próximo registro a ser entrado. Se você pressionar a tecla <**ENTER**> ou <**RETURN**> novamente, antes de começar a digitar o primeiro campo, o computador trará o menu principal de volta.

EXAMINE OS REGISTROS

Para examinar os registros que você entrou, selecione a opção número 3 do menu principal, "VER REGISTROS", pressionando a tecla 3. O primeiro registro será então exibido na tela — não necessariamente o primeiro que você introduziu, mas o primeiro de acordo com o próprio método de seleção do programa.

Os métodos de arranjo dos computadores variam muito pouco. Mas, de modo geral, eles selecionam os registros em ordem alfabética através do primeiro campo, o qual, em muitos casos, será "NOME". Para fazer isso, o computador examina a primeira letra do primeiro campo de todos os registros e os ordena alfabeticamente. Se mais de um registro tiver a mesma inicial, ele os ordena em função da segunda letra, e assim por diante.

Quando o primeiro campo contiver algarismos, o computador selecionará sempre um deles antes de qualquer letra; mas prosseguirá com o mesmo método de ordenação, dígito por dígito, quando estiver decidindo entre números, além de olhar para o número como um todo. Em outras palavras, se você colocar os números de 1 a 100 no primeiro campo, por exemplo, o computador ordenará em primeiro lugar os números 1, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19 e 100 antes de movimentar-se para o 2, 20, 21, etc. A maneira de se contornar esse problema é evitar o emprego de números no primeiro campo ou entrar os números com zeros à esquerda, como, por exemplo: 001, 002 ...010, 011 ... até 100.

Quando, por outro lado, se utiliza uma mistura de letras maiúsculas e mi-

núsculas, o computador escolhe primeiro as maiúsculas. Assim, "ABC Limitada" poderá ser antes de "Aarão e Companhia". Dependendo do que o arquivo de dados contiver, pode ser que o mais conveniente seja você digitar todas as informações apenas em letras maiúsculas, para resolver o problema. Se o seu computador aceitar caracteres acentuados, você poderá ter problemas na ordenação, pois na codificação brasileira para os caracteres (adotada, por exemplo, para os micros da linha MSX) as letras acentuadas aparecem depois das minúsculas, na ordem alfabética.

Quando você pedir para ver os registros, aparecerão as seguintes opções:

```
PROSSEGUE    RETORNA    MENU
```

escritas na parte inferior da tela. Se você pressionar a tecla P, o computador exibirá o registro seguinte; se você pressionar o R várias vezes, ele se movimentará rapidamente através de todo o arquivo, registro por registro.

Uma pressão na tecla R traz de volta à tela o registro anterior ao exibido. Assim, usando apenas o P e o R, você pode percorrer o arquivo, registro por registro, para a frente e para trás. Já a tecla M provoca o retorno ao menu principal, qualquer que seja o ponto onde você estiver no arquivo.

Embaixo da primeira linha de opções aparecerá uma segunda com as seguintes alternativas:

```
CORRIGE    APAGA    IMPRIME
```

Essas funções não constam do atual programa e serão explicadas no próximo artigo desta série. Portanto, nada acontecerá se você tentar usá-las na presente versão do programa.

ARMAZENE ARQUIVOS NA MEMÓRIA

À medida que você for entrando dados nos registros do seu arquivo, eles serão armazenados na memória principal (memória RAM) do computador, dentro do limite máximo no número de registros, calculado pelo programa. É necessário, portanto, armazenar esse arquivo em fita magnética, se você quiser desligar o computador; caso contrário, perderá toda a informação digitada. Posteriormente, quando você quiser modificar ou acrescentar dados, consultar ou listar o arquivo, poderá carregar de volta o arquivo para a memória RAM do computador, lendo-o em fita.

Na maioria das versões do programa listadas abaixo, o menu principal oferece as duas mencionadas opções ao usuário: gravar ou carregar o arquivo de dados. A única exceção é a versão para os micros da linha Sinclair, que gravam o programa juntamente com os dados contidos em memória.

Quando você quiser salvar o seu arquivo nos micros das linhas TRS-Color ou MSX, pressione a tecla 5 para selecionar a opção de armazenamento no menu principal, e o computador lhe pedirá que dê um nome qualquer ao arquivo. Uma vez que você tenha teclado o nome do arquivo e pressionado <**RETURN**> ou <**ENTER**>, o computador pedirá que você posicione o gravador para a gravação, apertando as teclas **RECORD** e **PLAY**, e pressionando = <**ENTER**> em seguida. Enquanto grava, o computador mantém você informado. O tempo de gravação depende do número de registros a ser armazenado.

Nos micros da linha Sinclair, a função de gravação é ativada de forma diferente da anterior. Neste caso, selecione primeiro a opção 7 (SAÍDA) no menu principal. Em seguida, usando o comando **SAVE** normal, digitado diretamente pelo teclado, acione o gravador e grave o programa, dando-lhe um nome adequado. Os dados serão gravados conjuntamente, conforme já foi explicado.

Posteriormente, quando quiser consultar o arquivo, você deve (nos micros das linhas citadas acima) carregar o computador em dois estágios. Inicialmente, é necessário carregar o programa, utilizando o comando normal **LOAD** (ou **CLOAD**) de sua máquina. Em seguida, deve-se mandar executar o programa, acionando o comando **RUN**, e selecionar a opção 6 no menu principal, "CARREGAR ARQUIVO". O computador pedirá então o nome do ar-

quivo desejado. Quando você tiver teclado o nome do arquivo e pressionado <**RETURN**> ou <**ENTER**>, a máquina ordenará: "POSICIONE O GRAVADOR E PRESSIONE <**ENTER**>". Feito isto, ele dirá novamente: "COLOQUE O GRAVADOR NA POSIÇÃO **PLAY** E PRESSIONE <**ENTER**>".

Assim que a fita cassete começar a rodar, o computador a examinará até encontrar o arquivo que você deseja, o qual será carregado (o que poderá demorar algum tempo). Concluída a operação, ele dirá que o arquivo foi carregado corretamente. Se o arquivo que você quiser não estiver na fita, o computador testará todos os arquivos que encontrar nela, até o final. De qualquer maneira, no final, o programa fará retornar o menu principal. Só então você poderá selecionar a opção 6 novamente e procurar outro arquivo para carregar.

Nos micros da linha Sinclair, como é o caso do TK-90X, os dados e o programa principal são carregados juntos. Uma vez que tiver carregado o primeiro arquivo utilizando o comando **LOAD** padrão, poderá carregar outros arquivos por meio da opção 6 no menu principal.

A versão do programa para computadores da linha Apple também tem funções de menu iguais às dos micros já considerados. A única diferença é que o programa utiliza arquivos em disquete, para armazenamento e recuperação de dados, pois não funciona satisfatoriamente com fita cassete.

Os programas listados contêm grandes intervalos em seus números de linhas — da linha 2000 à linha 6000.

Todavia, isso foi feito intencionalmente. Digite os programas exatamente como estão, pois as linhas que faltam serão utilizadas para incorporar as partes responsáveis pelas funções de modificação, apagamento, impressão e cruzamento de informações dos registros.

Os detalhes dessas opções estão no próximo artigo desta série.

```
20 PCLEAR1:CLEAR 11000:RS$="P R
MCAI":B$=CHR$(128)
30 CLS:PRINT@32,STRING$(8,B$);"
MENU";STRING$(3," ");"PRINCIPAL
";STRING$(8,B$)
40 PRINT @164,"1-ABRIR UM ARQUI
VO"
50 PRINT @196,"2-ENTRAR COM UM
REGISTRO"
60 PRINT @228,"3-VER REGISTROS"
70 PRINT @260,"4-OPCAO DE PROCU
RA"
80 PRINT @292,"5-SALVAR ARQUIVO
"
90 PRINT @324,"6-CARREGAR ARQUI
VO"
100 PRINT @356,"7-SAIDA
110 PRINT @481,"SELECIONE OPCAO
:";
120 IN$=INKEY$:IF IN$<"1"OR IN$
>"7" THEN 120
130 IF IN$ <>"1" AND IN$<>"6" A
ND R=0 AND IN$<>"7"THEN 120
140 SOUND 30,1:CLS:IN=VAL(IN$)
150 ON IN GOSUB 1000,2000,6000,
5000,7000,8000,9000
160 GOTO 30
1000 PRINT @41,"ABRIR UM ARQUIV
O":PRINT @231,"VOCE TEM CERTEZA
(S/N)?"
1010 IN$=INKEY$:IF IN$<>"S" AND
 IN$<>"N" THEN 1010
1020 IF IN$<>"S"THEN RETURN
1030 IF R>0 THEN RUN 9200
1040 CLS:PRINT @41,"ABRIR UM AR
QUIVO"
1050 PRINT @385,"NUMERO DE CAMP
OS(1-8)";:INPUT A: A=ABS(INT(A)
)
1060 IF A>8 OR A<1 THEN 1050
1070 DIM A(A),N$(A)
1080 PRINT @384,"":PRINT @96,""
:FOR N=1 TO A
1090 PRINT:PRINT"NOME DO CAMPO"
;N;"?";:LINEINPUT N$(N):N$(N)=L
EFT$(N$(N),10)
1100 PRINT"COMPRIMENTO DO CAMPO
";N;:INPUT A(N):A(N)=ABS(INT(A(
N)))
1110 IF A(N)>19 OR A(N)<1 THEN
1100
1120 TS=TS+A(N)
1130 NEXT:R=INT(11000/(5+5*A))-
1:PRINT" NUMERO MAXIMO DE REGIS
TROS=";R
1140 DIM A$(R,A):FOR I=1 TO 200
0:NEXT:RETURN
2000 G=0
2010 IF NR=R THEN 2180
2020 NR=NR+1
2030 CLS:PRINT @0,"VOCE USOU ";
NR-1;" DOS ";R;" REGISTROS DISP
ONIVEIS"
2040 FOR N=1 TO A:PRINT @32*N+3
2,N$(N);" :":PRINT @448,"":PRIN
T @416,""
2050 PRINT @416,"(ATE ";A(N);"C
ARACTERES)  ";:LINE INPUT A$(NR
,N)
2060 IF A$(NR,N)="" AND  N=1 TH
EN N=A:G=1:GOTO 2080
2070 A$(NR,N)=LEFT$(A$(NR,N),A(
N)):PRINT @32*N+45,A$(NR,N)
2080 NEXT
2090 IF G=1 THEN 2160
2100 C=NR:FOR F=1 TO 150:NEXT:I
F NR=1 THEN 2150
2110 IF A$(C,1)>=A$(C-1,1) THEN
 2150
2120 FOR N=1 TO A:X$=A$(C,N):A$
(C,N)=A$(C-1,N):A$(C-1,N)=X$:NE
XT:C=C-1
2130 IF C=1 THEN 2150
2140 GOTO 2110
2150 GOTO 2010
2160 NR=NR-1
2170 RETURN
2180 CLS3:PRINT @235," ARQUIVO
CHEIO ";:FOR G=1 TO 5: SCREEN 0
,1: FOR F=1 TO 500:NEXT
2190 SCREEN 0,0:FOR F=1 TO 500:
 NEXT F,G: RETURN
3000 RETURN:REM LINHA TEMPORARI
A
4000 RETURN:REM LINHA TEMPORARI
A
5000 RETURN:REM LINHA TEMPORARI
A
6000 D=1
6010 IF NR<1 THEN 6170
6020 GOSUB 8500
6030 PRINT @451,"pROSSEGUE    r
ETORNA   mENU  cORRIGE    aPAGA
    iMPRIME";
6040 IN$=INKEY$:IF IN$="" THEN
6040
6050 IN=INSTR(1,RS$,IN$)
6060 ON IN GOTO 6080,6080,6090,
6100,6110,6120,6130
6070 GOTO 6030
6080 D=D+1:GOTO 6140
6090 D=D-1:GOTO 6140
6100 RETURN
6110 GOSUB 3000:GOTO 6020
6120 GOSUB 4000:GOTO 6010
6130 GOSUB 10000:GOTO 6030
6140 IF D>NR THEN D=1
6150 IF D<1 THEN D=NR
6160 GOTO 6010
6170 CLS3:PRINT@233," ARQUIVO V
AZIO ";
6180 FOR G=1 TO 5:SCREEN 0,1:FO
```

MICRO DICAS

COMO TORNAR UM PROGRAMA LONGO MAIS FÁCIL DE DIGITAR

Digitar um programa longo escrito por outra pessoa pode ser uma tarefa bastante trabalhosa e desanimadora, mesmo para os mais persistentes. Mas você pode torná-la mais fácil, e gratificante, digitando apenas uma curta seção de cada vez e testando-a em seguida.

Alguns programas estão estruturados em seções ou módulos mais ou menos independentes, tais como subrotinas, procedimentos e laços, mas se você não conseguir identificá-los digite apenas seções curtas de 20 ou 30 linhas.

Quando nada se sabe de programação BASIC, a chance de errar uma linha e não perceber é relativamente grande, pois a mudança de apenas um pequeno sinal ou letra já é suficiente para impedir que o programa funcione corretamente.

Eis aí um bom incentivo para você seguir também o curso de programação BASIC de INPUT, de modo sistemático e desde o começo.

```
R F=1 TO 300:NEXT:SCREEN 0,0:FO
R F=1 TO 300:NEXT F,G:RETURN
7000 AUDIOON:MOTORON:CLS:PRINT
@65,"POSICIONE O GRAVADOR E PRE
SSIONE <ENTER>";
7010 IN$=INKEY$:IF IN$<>CHR$(13
) THEN 7010
7020 MOTOROFF:PRINT @129,"COLOQ
UE O GRAVADOR EM POSICAO 'REC'
E PRESSIONE <ENTER>";
7030 IN$=INKEY$:IF IN$<>CHR$(13
) THEN 7030
7040 PRINT:INPUT" NOME DO ARQUI
VO ";FI$
7050 CLS6:PRINT @232,"GRAVANDO
";FI$;
7060 MOTORON:FOR I=1 TO 1000:NE
XT
7070 OPEN"O",#-1,FI$
7080 PRINT#-1,FI$,R,A,NR
7090 FOR N=1 TO A:PRINT#-1,N$(N
),A(N):NEXT
7100 C=1
7110 IF A$(C,1)="" THEN 7140
7120 FOR N=1 TO A:PRINT#-1,A$(C
,N):NEXT
7130 C=C+1:GOTO 7110
7140 CLOSE#-1:RETURN
8000 CLS:PRINT@70,"VOCE TEM CER
TEZA (S/N)?"
8010 IN$=INKEY$:IF IN$<>"S" AND
 IN$<>"N" THEN 8010
8020 IF IN$="N" THEN RETURN
8030 AUDIOON:MOTORON:CLS:PRINT
@65,"POSICIONE O GRAVADOR E PRE
SSIONE <ENTER>"
8040 IN$=INKEY$:IF IN$<>CHR$(13
) THEN 8040
8050 MOTOROFF:PRINT @129,"COLOQ
UE O GRAVADOR  NA POSICAO  'PLA
Y' E PRESSIONE <ENTER>"
8060 IN$=INKEY$:IF IN$<>CHR$(13
) THEN 8060
8070 IF R>0 THEN RUN 9210
8080 INPUT " NOME DO ARQUIVO";F
I$
8090 CLS7:PRINT @231,"PROCURAND
O ";
8100 OPEN "I",#-1,FI$
8110 INPUT #-1,FI$
8120 PRINT @231," ACHEI  ";FI$;
" ";
8130 INPUT#-1,R,A,NR
8140 DIM A(A),N$(A),A$(R,A)
8150 FOR N=1 TO A:INPUT#-1,N$(N
),A(N):NEXT
8160 C=1
8170 IFEOF(-1)THEN 8200
8180 FOR N=1 TO A:INPUT #-1,A$(
C,N)
8190 NEXT: C=C+1:GOTO 8170
8200 CLOSE #-1:RETURN
8500 CLS:PRINT@0,"NUMERO DO REG
ISTRO";D:FOR N=1 TO A:PRINT @32
*N+32,N$(N);" :";TAB(13);A$(D,N
):NEXT:RETURN
9000 CLS4 : PRINT @70,"VOCE TEM
 CERTEZA (S/N)?";
9010 IN$=INKEY$:IF IN$<>"S" AND
 IN$<>"N" THEN 9010
9020 IF IN$="N" THEN RETURN
9030 CLS:END
9200 GOSUB 1040:GOTO 9220
9210 GOSUB 8080
9220 B$=CHR$(128):RS$="P RMCAI"
:GOTO 30
10000 PRINT @451,"   CHEQUE IMP
RESSORA    cONT ";
10010 PRINT @480,"
                ";
10020 IN$=INKEY$:IF IN$="" THEN
 10020
10030 IF IN$<>"C" THEN RETURN
10040 FOR Y=0 TO A+4:FOR X=0 TO
 31:P=PEEK(1024+X+Y*32):IFP>95
AND P<127 THEN P=P-64
10050 IF P>0 AND P<27 THEN P=P+
96
10060 IF P=0 THEN P=32
10070 PRINT #-2,CHR$(P);:NEXT:P
RINT#-2,CHR$(13);:NEXT
10080 FOR N=1 TO 3:PRINT#-2,CHR
$(13):NEXT
10090 RETURN
```

```
  5 LET R=0: LET U=0: LET V=1
 10 BORDER V: PAPER V: INK 7:
 POKE 23609,20: POKE 23658,8
100 CLS : PRINT  INVERSE V;AT
V,10; " M E N U "
110 PRINT AT 5,6;"1-Abrir um a
rquivo" ''TAB 6;"2-Entrar com
um registro"''TAB 6;"3-Ver reg
istros"''TAB 6;"4-Opcao de pro
cura"''TAB 6;"6-Carregar arqui
vo"''TAB 6;"7-Saida";#V;TAB 6;
"-SELECIONE OPCAO-"
500 LET  I$=INKEY$ : IF  I$=""
 THEN  GOTO 500
510 IF  I$<"1" OR I$>"7"  THEN
 GOTO 500
520 IF R=U  AND  I$<>"1"  AND
I$ <>"6" AND I$<>"7" THEN
 GOTO 500
530 SOUND .1,10: CLS : GOSUB (
 CODE I$ -48)*1000: GOTO 100
1000 PRINT AT 7,8;"Voce tem cer
teza?": PAUSE U: IF INKEY$="" T
HEN  GOTO 1000
1010 IF INKEY$<>"S" THEN  RETUR
N
1020 PRINT  INVERSE V;AT 10,5;"
 CRIAR UM NOVO ARQUIVO "
1030 INPUT AT 0,0;"Numero de ca
mpos(1-8)? ";A: IF a<1 OR a>8 T
HEN  GOTO 1030
1040 DIM A(A): DIM B(A+V): DIM
N$(A,10): LET T=U: FOR N=V TO A
1050 INPUT AT 0,0;"Nome do camp
o ";(N);" ? "; LINE N$(N)
1060 INPUT AT V,0;"Comprimento
do campo ";(N);" ? ";A(N): IF A
(N)>50 THEN  GOTO 1060
1070 LET B(N)=T: LET T=T+A(N):
NEXT N: LET B(N)=T
1080 PRINT AT 16,2;"Espaco para
 ";INT (((PEEK 23730+256*PEEK 2
3731)-29500)/T);" registros"
1090 INPUT "Quantos registros?
";R: DIM A$(R,T): RETURN
2000 LET C=V
2010 IF A$(C,V)=" " THEN  GOTO
2100
2020 IF C=R THEN  GOTO 2500
2030 LET C=C+V: GOTO 2010
2100 PRINT AT 0,0;"Voce usou ";
C-V;" dos ";R;" registros     d
isponiveis"
2110 FOR N=V TO A: PRINT  INVER
SE V;AT V+N*2,U;N$(N); INVERSE
0;AT V+N*2,12; FLASH V;"?": INP
UT "(ate ";(A(N));" caracteres)
", LINE A$(C,B(N)+V TO B(N+V)):
 PRINT AT V+2*N,12;A$(C,B(N)+V
TO B(N+V)): NEXT N
2120 FOR F=V TO 150: NEXT F: IF
 C=V THEN  RETURN
2130 LET N=C
2140 IF A$(C)>=A$(C-V) THEN  RE
TURN
2150 LET X$=A$(C): LET A$(C)=A$
(C-V): LET A$(C-V)=X$: LET C=C-
V: IF C=V THEN  RETURN
2160 GOTO 2140
2500 CLS : PRINT  FLASH 1;AT 10
,2;" A R Q U I V O   C H E I O
": FOR F=V TO 400: NEXT F: RETU
RN
3000 LET D=V: IF A$(V,V)=" " TH
EN  RETURN
3010 IF D=U THEN  LET D=V
3015 IF D-V=R THEN  LET D=D-V
3020 IF A$(D,V)=" " THEN  LET D
=D-V
3030 GOSUB 9500
3040 IF OP=V THEN  LET D=D+V: G
OTO 3010
3050 IF OP=2 THEN  LET D=D-V: G
OTO 3010
3060 IF OP=3 THEN  RETURN
3070 IF OP=4 THEN  GOSUB 8000
3080 IF OP=5 THEN  LET MD=V: GO
SUB 9000: IF D=U THEN  RETURN
3090 GOTO 3030
4000 RETURN : REM LINHA TEMPORA
RIA
5000 INPUT "Digite o nome do ar
quivo  "; LINE Q$: IF LEN Q$<V
OR LEN Q$>10 THEN  GOTO 5000
5010 SAVE Q$ LINE 10: RETURN
6000 PRINT AT 8,U;"Digite o nom
e do  arquivo  a sercarregado,
ou somente <ENTER> para carrega
r o primeiro arquivo"
6010 INPUT  LINE X$: IF LEN X$>
10 THEN  GOTO 6010
6020 PRINT AT 13,U;"PRESSIONE P
LAY NO GRAVADOR": LOAD X$
7000 PRINT AT 10,8;"Voce tem ce
rteza?": IF INKEY$="" THEN  GOT
O 7000
7010 IF INKEY$<>"S" THEN  RETUR
N
7020 RAND USR U
8000 RETURN : REM LINHA TEMPORA
RIA
9000 RETURN : REM LINHA TEMPORA
RIA
9500 PRINT AT U,U;"Registro num
ero ";D;"  ": FOR N=V TO A: PRI
NT  INVERSE V;AT V+2*N,U;N$(N);
 INVERSE U;TAB 12;A$(D,B(N)+V T
O B(N+V)): NEXT N
9510 PRINT  INVERSE V;AT 20,U;"
P(rossegue)  R(etorna)  M(enu)
  E(menda)    A(paga)  I(mprim
e)"
```

```
9520 IF INKEY$="" THEN  GOTO 95
20
9530 LET V$=INKEY$: IF V$="I" T
HEN  COPY : LPRINT : LPRINT : L
PRINT : GOTO 9520
9540 LET OP=U: IF V$="P" THEN
LET OP=V: LET MO=V
9550 IF V$="R" THEN  LET OP=2:
LET MO=-V
9560 IF V$="M" THEN  LET OP=3
9570 IF V$="E" THEN  LET OP=4
9580 IF V$="A" THEN  LET OP=5
9590 IF OP=U THEN  GOTO 9520
9600 SOUND .1,10: RETURN
```

```
10 KEYOFF:MOTOROFF
20 CLS:CLEAR:R$="P RMCAI"
30 CLStLOCATE 5,1:PRINT" M E N
U   P R I N C I P A L "
40 LOCATE 9,5:PRINT"1:-ABRIR UM
 ARQUIVO"
50 LOCATE 9,7:PRINT"2:-ENTRADA
DE REGISTROS"
60 LOCATE 9,9:PRINT"3:-LISTAR R
EGISTROS"
70 LOCATE 9,11:PRINT"4:-BUSCA D
E INFORMAÇÕES"
80 LOCATE 9,13:PRINT"5:-GRAVAR
UM ARQUIVO"
90 LOCATE 9,15:PRINT"6:-CARREGA
R UM ARQUIVO"
100 LOCATE 9,17:PRINT"7:-FIM DE
 PROGRAMA"
110 LOCATE 14,19:PRINT"OPÇÃO: "
;
120 IN$=INKEY$:IF IN$<"1"ORIN$>
"7"THEN120
130 IFIN$<>"1"ANDIN$<>"6"ANDIN$
<>"7"ANDR=0THEN 120
140 PRINTCHR$(7) :CLS:IN=VAL(IN
$)
150 ON IN GOSUB 1000,2000,6000,
5000,7000,8000,9000
160 GOTO 30
1000 LOCATE 7,0:PRINT"MONTAR UM
 NOVO ARQUIVO":LOCATE 7,14:PRIN
T"Você tem certeza (S/N)?";
1010 IN$=INKEY$:IFIN$<>"S"ANDIN
$<>"N"THEN 1010
1020 IFIN$<>"S"THEN RETURN
1030 IFR>0THEN RUN 9200
1040 CLS:LOCATE 7,0:PRINT"MONTA
R UM NOVO ARQUIVO"
1050 LOCATE 4,2:PRINT"Número de
 campos (1-8) ";:INPUT A:A=ABS(
INT(A))
1060 IF A>8 OR A<1 THEN1050
1070 DIM A(A),N$(A)
1080 LOCATE 0,6:FOR N=1TOA
1090 PRINT:PRINT"Nome do campo
";N;"? ";:LINEINPUTN$(N):N$(N)=
LEFT$(N$(N),10)
1100 PRINT"Tamanho do campo ";N
;:INPUTA(N):A(N)=ABS(INT(A(N)))
1110 IFA(N)>250RA(N)<1THEN1100
1120 TS=TS+A(N)
1130 NEXT:R=INT(3000/(TS+3*A)):
PRINT:PRINT"Número máximo de re
gistros =>";R
1140 DIMA$(R+1,A):FORI=1TO1000:
NEXT:RETURN

2000 G=0
2010 IFNR=RTHEN2190
2020 NR=NR+1
2030 CLS:PRINTNR-1;" de ";R;" r
egistros em uso"
2040 FORN=1TOA:LOCATE0,N*2+2:PR
INTN$(N);": ":LOCATE0,22:PRINTC
HR$(5);:LOCATE0,23:PRINTCHR$(5)
;
2050 LOCATE0,23:PRINT"Até ";A(N
);" caracteres";
2060 LOCATE0,21:PRINTCHR$(21);:
LINEINPUTA$(NR,N)
2070 IFA$(NR,N)=""ANDN=1THENN=A
:G=1:GOTO2090
2080 A$(NR,N)=LEFT$(A$(NR,N),A(
N)):LOCATE12,N*2+2:PRINTA$(NR,N
)
2090 NEXT
2100 IFG=1THEN2170
2110 C=NR:FORF=1TO500:NEXT:IFNR
=1THEN2160
2120 IFA$(C,1)>=A$(C-1,1)THEN21
60
2130 FORN=1TOA:SWAPA$(C,N),A$(C
-1,N):NEXT:C=C-1
2140 IFC=1THEN2160
2150 GOTO2120
2160 GOTO2010
2170 NR=NR-1
2180 RETURN
2190 CLS:LOCATE10,15:PRINT"ARQU
IVO CHEIO!":FORF=1TO1000:NEXT
2200 RETURN
3000 RETURN:REM LINHA TEMPORÁRI
A
4000 RETURN:REM LINHA TEMPORÁRI
A
5000 RETURN:REM LINHA TEMPORÁRI
A
6000 D=1
```

```
6010 IFNR<1THEN6170
6020 GOSUB8500
6030 LOCATE3,22:PRINT"[P]rosseg
ue   [R]etorna   [M]enu   [C]o
rrige   [A]paga   [I]mprime"
;
6040 IN$=INKEY$:IFIN$=""THEN604
0
6050 IN=INSTR(1,R$,IN$)
6060 ON IN GOTO 6080,6080,6090,
6100,6110,6120,6130
6070 GOTO6030
6080 D=D+1:GOTO6140
6090 D=D-1:GOTO6140
6100 RETURN
6110 GOSUB3000:GOTO6020
6120 GOSUB4000:GOTO6010
6130 GOSUB10000:GOTO6020
6140 IFD>NRTHEND=1
6150 IFD<1THEND=NR
6160 GOTO 6010
6170 CLS:LOCATE10,15:PRINT"ARQU
IVO VAZIO!"
6180 FORF=1TO500:NEXT:RETURN
7000 MOTOR:CLS:LOCATE0,5:PRINT"
Posicione a fita e tecle [retur
n]"
7010 IN$=INKEY$:IFIN$<>CHR$(13)
THEN7010
7020 MOTOR:PRINT:PRINT:PRINT"Co
loque o gravador em modo de gra
vação e pressione [return]"
7030 IN$=INKEY$:IFIN$<>CHR$(13)
THEN7030
7040 PRINT:INPUT"Nome do arquiv
o ";FL$ :F$="CAS:"+FL$
7050 LOCATE10,22:PRINT"Gravando
...";
7060 MOTOR:FORI=1TO1000:NEXT
7070 OPEN FL$ FOR OUTPUT AS#1
7080 PRINT #1,FL$;",";R,A,NR
7090 FORN=1TOA:PRINT#1,N$(N);",
";A(N):NEXT
7100 C=1
7110 IFA$(C,1)=""THEN7140
7120 FORN=1TOA:PRINT#1,A$(C,N):
NEXT
7130 C=C+1:GOTO7110
7140 CLOSE#1:RETURN
8000 CLS:LOCATE10,15:PRINT"Você
 tem certeza? (S/N) "
8010 IN$=INKEY$:IFIN$<>"S"ANDIN
$<>"N"THEN8010
8020 IFIN$="N"THENRETURN
8030 MOTOR:CLS:LOCATE0,3:PRINT"
Posicione a fita e tecle [retur
n]"
8040 IN$=INKEY$:IFIN$<>CHR$(13)
THEN8040
8050 MOTOR:LOCATE0,7:PRINT"Colo
que o gravador em modo de repro
dução e tecle [return]"
8060 IN$=INKEY$:IFIN$<>CHR$(13)
THEN8060
8070 IFR>0THENRUN9210
8080 PRINT:INPUT"Nome do arquiv
o ";FL$:F$="CAS:"+FL$
8090 LOCATE10,22:PRINT"Procuran
do..."
8100 OPEN F$ FOR INPUT AS#1
8110 INPUT #1,FL$
8120 CLS:LOCATE 10,15:PRINT"Ach
ei ";FL$
8130 INPUT#1,R,A,NR
8140 DIMA(A),N$(A),A$(R,A)
8150 FORN=1TOA:INPUT#1,N$(N),A(
N):NEXT
8160 C=1
8170 IFEOF(1)THEN8200
8180 FORN=1TOA:INPUT#1,A$(C,N)
8190 NEXT:C=C+1:GOTO8170
8200 CLOSE#1:RETURN
8500 CLS:PRINT"Número do regist
ro =>";D:FORN=1TOA:LOCATE0,N*2+
2:PRINTN$(N);" :";:LOCATE13:PRI
NTA$(D,N):NEXT:RETURN
9000 CLS:LOCATE10,15:PRINT"Você
 tem certeza? (S/N) "
9010 IN$=INKEY$:IFIN$<>"S"ANDIN
$<>"N"THEN9010
9020 IFN$="N" THENRETURN
9030 CLS:END
9200 GOSUB1040:GOTO9220
9210 GOSUB8080
9220 R$="P RMCAI"
10000 CLS:LOCATE10,15:PRINT"CHE
QUE A IMPRESORA"
10010 PRINT:LOCATE12:PRINT"ALIN
HE O PAPEL"
10020 IN$=INKEY$:IFINKE$=""THEN
10020
10040 LPRINT:LPRINT"Registro nú
mero ";D
10050 FORX=1TOA:LPRINT:LPRINT N
$(X);" :";A$(D,X):NEXT
10060 LPRINT:LPRINT:LPRINT:LPRI
NT
10070 CLS:RETURN
```

```
20  TEXT : CLEAR :D$ =  CHR$ (4
)
25  ONERR  GOTO 11000
30  HOME : VTAB 1: HTAB 6: INVE
RSE : PRINT " M E N U    P R I
N C I P A L ": NORMAL
40  VTAB 5: HTAB 10: PRINT "1:-
ABRIR UM ARQUIVO"
50  PRINT : HTAB 10: PRINT "2:-
ENTRADA DE REGISTROS"
60  PRINT : HTAB 10: PRINT "3:-
LISTAR REGISTROS"
70  PRINT : HTAB 10: PRINT "4:-
BUSCA DE INFORMACOES"
80  PRINT : HTAB 10: PRINT "5:-
GRAVAR O ARQUIVO"
90  PRINT : HTAB 10: PRINT "6:-
CARREGAR UM ARQUIVO"
100  PRINT : HTAB 10: PRINT "7:
-FIM DE PROGRAMA"
110  VTAB 20: HTAB 15: PRINT "O
PCAO: ";
120  GET IN$
130  IF IN$ < "1" OR IN$ > "7"
THEN 110
135  IF R = 0 AND IN$ <  > "1"
AND IN$ <  > "6" AND IN$ <  > "
7" THEN 110
140  PRINT  CHR$ (7): HOME :IN
=  VAL (IN$)
150  ON IN GOSUB 1000,2000,6000
,5000,7000,8000,6520
160  GOTO 30
1000  HTAB 8: INVERSE : PRINT "
 MONTAR UM NOVO ARQUIVO ": VTAB
 15: HTAB 10: PRINT " VOCE CONF
IRMA? (S/N) ";
1005  NORMAL
1010  GET IN$: IF IN$ <  > "S"
AND IN$ <  > "N" THEN 1010
1020  IF IN$ <  > "S" THEN  RET
URN
1030  IF R > 0 THEN  CLEAR :D$
=  CHR$ (4):IN = 1: HOME : GOTO
 150
1040  HOME : HTAB 8: INVERSE :
PRINT " MONTAR UM NOVO ARQUIVO
": NORMAL
1050  VTAB 3: HTAB 5: PRINT "NU
MERO DE CAMPOS (1-8): ";: INPUT
 A
1060  IF A > 8 OR A < 1 THEN 10
50
1070  DIM A(A),N$(A)
1080  VTAB 7: FOR N = 1 TO A
1090  PRINT "NOME DO CAMPO ";N;
" ";: INPUT N$(N):N$(N) =  LEFT
$ (N$(N),10)
1100  PRINT "TAMANHO DO CAMPO "
;N;" ";: INPUT A(N):A(N) =  ABS
 ( INT (A(N)))
1110  IF A(N) < 1 OR A(N) > 25
THEN 1100
1120 T$ = T$ + A(N): PRINT
1130  NEXT :R =  INT (( FRE (0)
 - 5000) / (T$ + 2 * A)): IF R
> 4000 THEN R = 4000
1135 R = 3: REM  RETIRAR ESTA L
INHA!!!!!!!!!!!!!!!!************
************
1140  PRINT : PRINT "NUMERO MAX
IMO DE REGISTROS: ";R
1150  DIM A$(R,A): FOR I = 1 TO
 2000: NEXT : RETURN
2000 G = 0
2010  IF NR = R THEN 2180: REM
 CHECK GOTO
2020 NR = NR + 1
2030  HOME : PRINT NR - 1;" DE
";R;" REGISTROS EM USO"
2040  FOR N = 1 TO A: VTAB N *
2 + 2: PRINT N$(N);":": VTAB 23
: CALL  - 958
2050  VTAB 24: PRINT "ATE ";A(N
);" CARACTERES";: VTAB 23: HTAB
 1: INPUT A$(NR,N)
2060  IF A$(NR,N) = "" AND N =
1 THEN NR = NR - 1: RETURN
2070 A$(NR,N) =  LEFT$ (A$(NR,N
),A(N)): VTAB N * 2 + 2: HTAB 1
3: PRINT A$(NR,N)
2080  NEXT
2090  IF A$(NR,1) >  = A$(NR -
1,1) THEN 2150
2100  FOR C = NR TO 2 STEP  - 1
2110  FOR N = 1 TO A:X$ = A$(C,
N):A$(C,N) = A$(C - 1,N):A$(C -
 1,N) = X$: NEXT
2120  NEXT
2150  FOR F = 1 TO 1000: NEXT :
 GOTO 2010
2180  VTAB 3: HTAB 1: CALL  - 9
58: VTAB 15: HTAB 10: FLASH : P
RINT " ARQUIVO CHEIO! "
2190  PRINT  CHR$ (7): FOR F =
1 TO 1000: NEXT : PRINT  CHR$ (
7): RETURN
3000  RETURN : REM  LINHA TEMPO
```

```
RARIA
4000  RETURN : REM  LINHA TEMPO
RARIA
5000  RETURN : REM  LINHA TEMPO
RARIA
6000  IF NR < 1 THEN  VTAB 15:
HTAB 12: FLASH : PRINT " ARQUIV
O VAZIO! ": FOR F = 1 TO 1000:
NEXT : RETURN
6010 D = 1
6020  POKE 35,22: VTAB 23: INVE
RSE : HTAB 4: PRINT "P";: HTAB
18: PRINT "R";: HTAB 32: PRINT
"M"
6030  VTAB 24: HTAB 8: PRINT "C
";: HTAB 20: PRINT "A";: HTAB 3
0: PRINT "I";: NORMAL
6040  VTAB 23: HTAB 5: PRINT "R
OSSEGUE";: HTAB 19: PRINT "ETOR
NA";: HTAB 33: PRINT "ENU"
6050  VTAB 24: HTAB 9: PRINT "O
RRIGE";: HTAB 21: PRINT "PAGA";
: HTAB 31: PRINT "MPRIME";
6060  GOSUB 6500
6070  GET IN$
6080  IF IN$ =  CHR$ (32) OR IN
$ = "P" THEN D = D + 1: IF D >
NR THEN D = 1
6090  IF IN$ = "R" THEN D = D -
 1: IF D < 1 THEN D = NR
6100  IF IN$ <  > "I" AND IN$ <
  > "A" AND IN$ <  > "C" AND IN
$ <  > "M" THEN 6060
6110  POKE 35,24
6115  IF IN$ = "M" THEN  RETURN

6120  IF IN$ = "I" THEN  GOSUB
10000: GOTO 6020
6130  IF IN$ = "A" THEN  GOSUB
4000: GOTO 6020
6140  IF IN$ = "C" THEN  GOSUB
3000: GOTO 6020
6150  GOTO 6060
6500  VTAB 1: HTAB 1: CALL  - 9
58: PRINT "NUMERO DO REGISTRO =
>";: INVERSE : PRINT D: NORMAL
6510  FOR I = 1 TO A: VTAB I *
2 + 2: PRINT N$(I);: HTAB 13: P
RINT A$(D,I): NEXT : RETURN
6520  END
7000  HOME : VTAB 10: HTAB 10:
PRINT "GRAVAR O ARQUIVO"
7005  PRINT : PRINT : HTAB 5: I
NPUT "NOME DO ARQUIVO =>";AR$
7010  PRINT : HTAB 5: PRINT "CO
NFERE ? ";: GET IN$
7015  IF IN$ <  > "S" THEN  RET
URN
7020  PRINT : PRINT D$;"OPEN";A
R$: PRINT D$;"DELETE";AR$
7030  PRINT D$;"OPEN";AR$
7040  PRINT D$;"WRITE";AR$
7050  PRINT R: PRINT A: PRINT N
R
7055  FOR I = 1 TO A: PRINT N$(
I): PRINT A(I): NEXT
7060  FOR J = 1 TO NR
7070  FOR I = 1 TO A
7080  PRINT A$(J,I): NEXT : NEX
T
7090  PRINT D$;"CLOSE";AR$
7100  RETURN
8000  CLEAR :D$ =  CHR$ (4)
8010  VTAB 10: HTAB 10: PRINT "
CARREGAR DADOS"
8020  PRINT : HTAB 5: INPUT "NO
ME DO ARQUIVO =>";AR$
8025  HTAB 5: PRINT "CONFERE ?
";: GET IN$
8027  IF IN$ <  > "S" THEN  GOT
O 30
8030  PRINT D$;"OPEN";AR$
8040  PRINT D$;"READ";AR$
8050  INPUT R,A,NR: DIM A$(R,A)
,N$(A),N(A)
8060  FOR I = 1 TO A: INPUT N$(
I),A(I): NEXT
8066  FOR J = 1 TO NR
8069  FOR I = 1 TO A
8071  INPUT A$(J,I): NEXT : NEX
T
8073  PRINT D$;"CLOSE";AR$
8078  GOTO 30
10000  HOME : VTAB 23: HTAB 9:
PRINT " CHEQUE A IMPRESSORA! "
10010  HTAB 12: PRINT " ALINHE
O PAPEL ";
10015  GET IN$
10020  VTAB 1: PRINT : PRINT  C
HR$ (4);"PR#1"
10030  PRINT : PRINT "REGISTRO
NUMERO ";D
10040  FOR X = 1 TO A: PRINT :
PRINT N$(X);" :"; TAB( 15);A$(D
,X): NEXT
10050  PRINT : PRINT : PRINT :
PRINT : PRINT
10055  PRINT  CHR$ (4);"PR#0"
10060  HOME : RETURN
11000  PRINT D$;"CLOSE": GOTO 2
0
```

O QUE É UMA BASE DE DADOS?

O programa de arquivamento de dados listado neste artigo e no próximo pode ser utilizado em muitos tipos de aplicações, pois o usuário tem a oportunidade de definir, ele mesmo, os **campos** *ou diferentes itens de informação que constituirão a ficha de cadastramento a ser usada.*

Em computação, esse tipo de sistema flexível, ou programável pelo usuário final, é comumente chamado de **sistema gerenciador de base de dados***: além de possuir as funções essenciais de qualquer sistema de cadastramento, esse sistema também tem um módulo de criação do* **lay out** *do arquivo, conforme o uso que ele terá.*

Uma base de dados, na definição mais estrita do termo, é um conjunto de arquivos de computador, que tem a mesma finalidade do ponto de vista do armazenamento e recuperação de informações. Normalmente, portanto, uma dessas bases inclui mais de um arquivo: por exemplo, em uma base de dados bibliográficos (artigos de revistas), teríamos o arquivo-mestre, ou arquivo-tombo, o arquivo de índice por autores, o arquivo de índice por assuntos, o arquivo de nomes de revistas, e assim por diante. Todos eles podem ser interrelacionados por meio de um ou mais campos em comum (por exemplo, o número de tombo do artigo, e o código da revista).

Entretanto, podemos considerar, de forma simplificada, que uma base de dados com apenas um arquivo também pode cair nessa definição. É o caso do programa listado em BASIC neste artigo.

A única condição essencial para que ele continue sendo chamado de um sistema gerenciador de base de dados é que o usuário possa definir livremente os campos de informação que quer ter em cada ficha (desde que, é claro, sejam respeitados os limites intrínsecos do programa).

MODELOS DE BASES DE DADOS

Existem diversas formas de organização da informação em uma base de dados. Normalmente, todas as formas têm pelo menos uma característica em comum: a subdivisão do arquivo em registros individuais.

A forma pela qual esses registros (ou os campos do registro) são organizados é que difere de caso para caso. Na verdade, a organização de uma base de dados é ditada pelo programa aplicativo que a controla.

A forma de organização mais comum para uma base de dados é a chamada **estrutura relacional**. *Ela é bastante fácil de entender e muito intuitiva, pois se equivale a uma tabela, ou planilha de dupla entrada, com linhas e colunas. Essa tabela é chamada de* **relação**, *daí o nome da estrutura. Como você já deve ter percebido, o programa listado neste artigo é uma base de dados relacional.*

Na estrutura relacional, as linhas da tabela correspondem sempre aos registros (por exemplo, cada livro de uma biblioteca), ao passo que as colunas correspondem aos diversos campos (tais como o nome do autor, o título da revista, o ano de publicação, etc.).

AS PLACAS DE SINALIZAÇÃO

Universalmente adotados, os sinais de trânsito servem para orientar o fluxo do tráfego. No mundo dos micros, os comandos GOTO e GOSUB cumprem uma função semelhante.

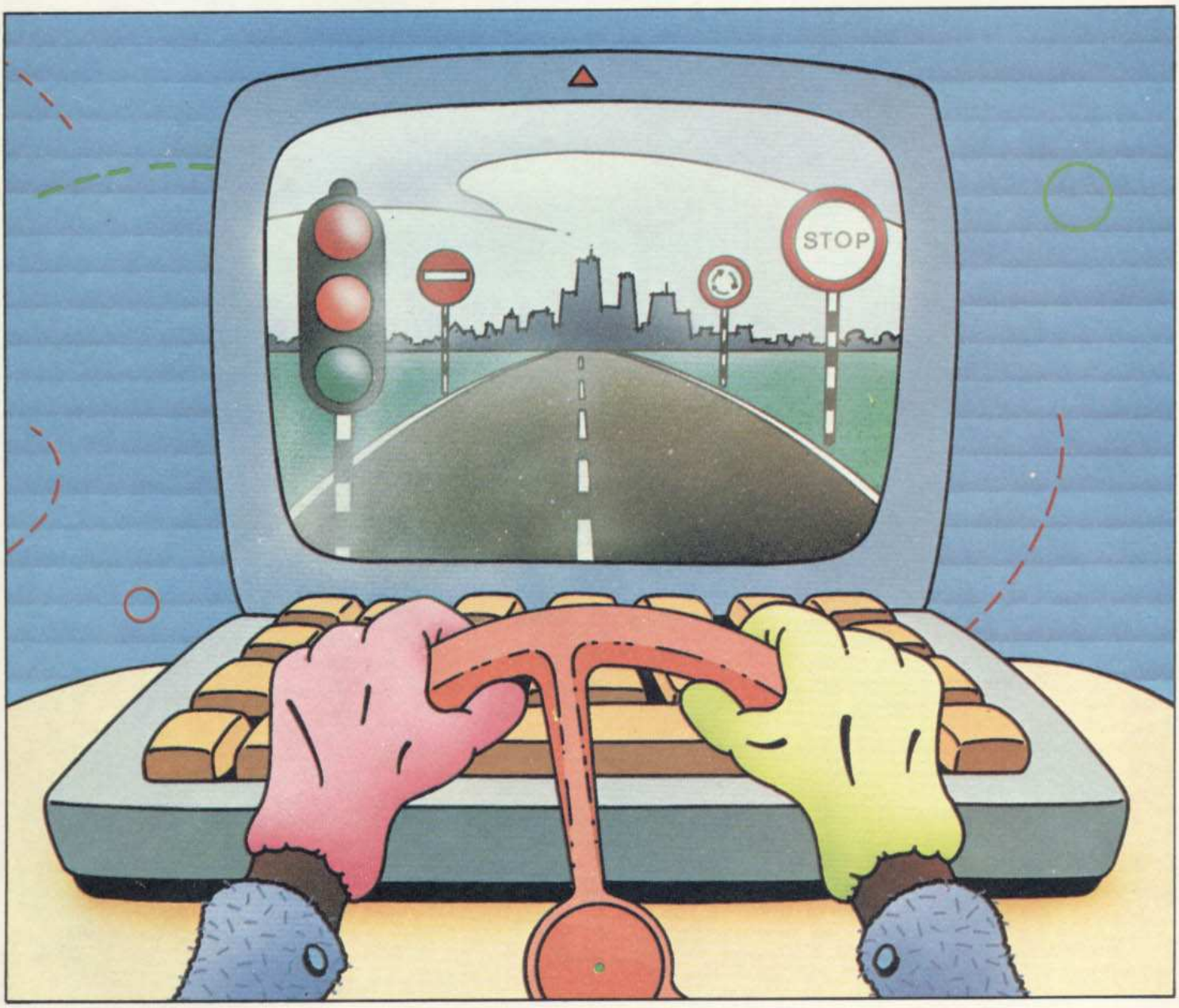
Uma das instruções mais importantes em programação BASIC é o **GOTO** ("vá para", em inglês). Sua função é alterar o fluxo de execução de um programa, de modo que, ao invés de simplesmente executar as linhas do programa em ordem linear, o computador salta para a linha indicada na declaração **GOTO**.

Embora possa aparecer na tela como duas palavras separadas, **GOTO** normalmente é digitada como uma só expressão. Nos micros da linha Sinclair, você pressiona a tecla marcada **GOTO** para entrar a instrução no computador. Em outros tipos de micros, você digita **GOTO** sem o espaço entre **GO** e **TO**.

A palavra **GOTO** deve ser sempre seguida pelo número da linha para onde você quer que o programa salte, como no exemplo abaixo:

```
30 GOTO 125
```

Em alguns computadores, como os da linha Sinclair, em vez de um número, podemos ter uma expressão ou equação de cálculo (inclusive o nome de uma variável), que assume um valor numérico quando o programa é rodado.



- GOTO E GOSUB NA PRÁTICA
- PROGRAMAS PARA CÁLCULO, ADIVINHAÇÃO DE NOMES E JOGOS DE DADOS
- QUANDO E COMO CRIAR DESVIOS EM UM PROGRAMA
- UTILIZAÇÃO EM DESVIOS COMPLEXOS
- COMO ACELERAR AS SUB-ROTINAS


```
10 LET A=10
20 GOTO A+5  equivale a  GOTO 15
```

As declarações **GOTO** permitem também que você salte para trás em um programa, criando um laço. Esse tipo de laço é semelhante àquele que aprendemos a montar com as declarações **FOR...NEXT**, embora não haja, neste caso, um limite para o número de vezes que o programa o executa. O programa abaixo calcula o valor da hipotenusa de um triângulo retângulo (note de que maneira os valores em A e B são elevados ao quadrado: multiplicando-os por si mesmo, o que é mais rápido do que usar a operação de exponenciação).

```
10 PRINT "COMPRIMENTO DOS LADOS
 A,B EM CM"
15 INPUT A,B
20 LET C=SQR(A*A+B*B)
30 PRINT"O COMPRIMENTO DO LADO
C E";C;"CM"
40 GOTO 10
```

Na linha 20, a função **SQR** é utilizada para extrair a raiz quadrada (**SQR** é a abreviatura de *square root*, expressão inglesa que quer dizer exatamente "raiz quadrada") do que está entre parênteses, ou seja, a expressão toda significa: "extraia a raiz quadrada de A ao quadrado mais B ao quadrado, e armazene o resultado em C". (Esta é a fórmula de Pitágoras, $\sqrt{A^2 + B^2} = C^2$, para calcular o comprimento C da hipotenusa de um triângulo retângulo, cujos catetos são A e B.) A linha 30 imprime o valor resultante em C.

Em virtude da linha 40, o programa se auto-repetirá, pois cada vez que ele a alcança a declaração **GOTO** manda-o de volta à linha 10, e o programa é executado outra vez.

A única maneira de sair desse ciclo é interromper externamente o programa, através da tecla <**BREAK**> ou <**CTRL**> <**C**>, <**CTRL**> <**STOP**>, etc., dependendo do computador, ou então desligá-lo e começar de novo (pois o programa na memória se perde).

SALTOS PARA A FRENTE

A declaração **GOTO** pode ser utilizada para saltar para a frente, por sobre um bloco do programa. É o que acontece no programa a seguir, que simula um jogo de cara ou coroa:

```
5 FOR F=1 TO 500:NEXT
F:CLS
10 PRINT "ESTOU JOGANDO
A MOEDA ..."
20 FOR J=1 TO 3
30 FOR F=1 TO 250:NEXT F
40 PRINT".";
50 NEXT J
60 PRINT
70 IF RND(0)<.5 THEN GOTO 100
80 PRINT"E DEU COROA!"
90 GOTO 5
100 PRINT"E DEU CARA!"
110 GOTO 5
```

Digite o programa abaixo usando somente maiúsculas, se o seu computador for compatível com o ZX-81:

```
 5  PAUSE 50: CLS
10 PRINT "Estou jogando a moe
da...";
20 FOR j=1 TO 3
```

```
 30 PAUSE 25
 40 PRINT ".";
 50 NEXT j
 60 PRINT
 70 IF RND<.5 THEN  GOTO 100
 80 PRINT "E' coroa!!!"
 90 GOTO 5
100 PRINT "E' cara!!!"
110 GOTO 5
```

```
5 FOR F=1 TO 500 : NEXT
6 CLS
10 PRINT "ESTOU JOGANDO A
MOEDA..."
20 FOR J=1 TO 3
30 FOR F=1 TO 250 : NEXT F
40 PRINT ".";
50 NEXT J
60 PRINT
70 IF RND(1)<.5 THEN GOTO 100
80 PRINT "E DEU COROA !"
90 GOTO 5
100 PRINT "E DEU CARA !"
110 GOTO 5
```

A função **RND** na linha 70 sorteia um número aleatório entre 0 e 1. Aqui, a declaração **GOTO** faz parte da declaração **IF**. Se o número selecionado pelo computador for menor do que 5, o computador saltará para a frente, para a linha 100. Se ele não satisfizer essa condição, o computador executará normalmente a linha seguinte do programa — a linha 80 —, e quaisquer outros comandos porventura existentes na linha 70 serão desprezados.

Essa condição significa que a linha 70 constrói uma bifurcação no programa. Existem, neste caso, duas alternativas: ou o computador executa as linhas 70, 80 e 90, para exibir na tela a mensagem "E DEU CARA?", ou, então, executa as linhas 70, 100 e 110, para exibir "E DEU COROA!". Isso é feito repetidas vezes, de forma totalmente aleatória, como uma espécie de lançamento rápido de uma "moeda eletrônica".

As linhas 90 e 100 também contêm declarações **GOTO**. Qualquer que seja o desvio tomado pelo programa, elas o mandarão de volta ao início, na linha 5, para começar outro sorteio.

Mais uma vez, você notará que este programa não termina nunca. A única maneira de pará-lo é pressionar a tecla de interrupção do seu computador, ou desligá-lo.

DESVIOS MAIS COMPLEXOS

Nos computadores da linha Sinclair, a declaração **GOTO** não precisa ser seguida de uma constante numérica. Uma variável servirá: **GOTO A**, por exemplo, ou **GOTO (100 ÷ INT (RND*6))**. Isso significa que a declaração **GOTO** pode proporcionar um tipo de desvio mais complexo ao seu programa; veja a seguir.

Digite o programa abaixo usando somente maiúsculas, se o seu computador for compatível com o ZX-81.

```
100 PRINT "Ola, qual e o seu n
ome?"
110 INPUT a$
120 GOTO (130+INT (RND*4)*10)
130 PRINT "E' um bonito nome,
";a$: GOTO 170
140 PRINT "E' um nome gozado,
";a$: GOTO 170
150 PRINT "Prazer em conhece-l
o, ";a$: GOTO 170
160 PRINT "Ola, ";a$;", eu sou
 seu computador."
170 STOP
```

A declaração **GOTO** na linha 120 dá um salto aleatório para a frente, em direção a qualquer uma das quatro linhas com as alternativas do programa, em função do resultado da expressão numérica. O número 130 na expressão pode ser somado a 0, 10, 20 ou 30, dependendo do resultado da expressão com RND. Assim, o programa poderá saltar para as linhas 130, 140, 150 ou 160.

O **GOTO 170**, após cada alternativa, faz com que o programa salte para a frente, ignorando as linhas intermediárias. Note que não necessitamos de um **GOTO 170** após a linha 160, pois o programa vai para a linha 170 de qualquer maneira (é a última alternativa de **GOTO** na linha 120).

Assim, este programa roda apenas uma vez, pois todos os **GOTO** instruem o computador para saltar para a instrução **STOP** na linha 170, sem formar laços de retorno.

A INSTRUÇÃO ON...GOTO

Os computadores da linha TRS, MSX e Apple têm um equivalente ao **GOTO** variável existente apenas nos computadores da linha Sinclair. É a declaração **ON...GOTO**. Esta última é bastante poderosa e útil em programação, e tem a seguinte forma (um exemplo):

```
ON A GOTO 100,200,300,400
```

Neste exemplo, quando a variável A for igual a 1, o programa saltará para o primeiro número de linha na lista que se segue à palavra **GOTO**, ou seja, a linha de número 100. Se A for igual a 2, o programa irá para a linha 200, e assim por diante. A lista de linhas após **GOTO** pode ter qualquer número de elementos, com um mínimo de 2, e um máximo determinado pelo número máximo de caracteres por linha de programa que o seu computador aceita (normalmente 254 ou 255 caracteres).

Novamente, esta declaração permite a programação de desvios complexos. O programa anterior passa a ser assim.

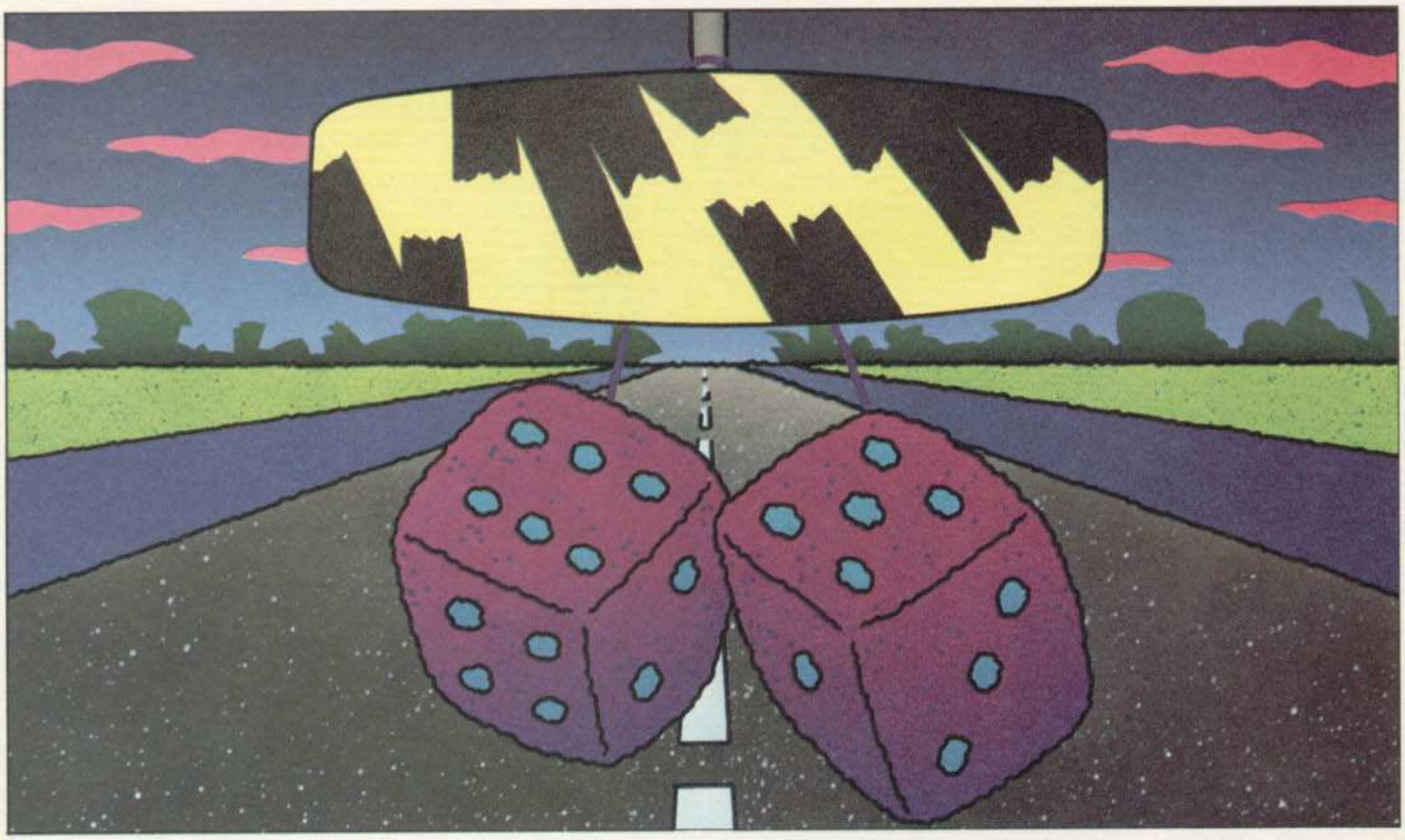
```
100 PRINT"OLA, QUAL E O SEU NOM
E?"
110 INPUT A$
120 ON RND(4) GOTO 130,140,150,
160
130 PRINT"E UM BONITO NOME, ";A
$:GOTO 170
140 PRINT"E UM NOME GOZADO, ";A
$:GOTO 170
150 PRINT"PRAZER EM CONHECE-LO,
 ";A$:GOTO 170
160 PRINT "OLA, ";A$;", EU SOU
O SEU COMPUTADOR."
170 END
```

```
100 PRINT "OLA, QUAL E' O SEU
NOME ?"
110 INPUT A$
120 ON INT(RND(1)*4)+1 GOTO
130, 140, 150, 160
130 PRINT "E' UM BONITO NOME, "
;A$:GOTO 170
140 PRINT "E' NOME GOZADO, "
;A$:GOTO 170
150 PRINT "PRAZER EM CONHECE-
LO, " ;A$:GOTO 170
160 PRINT "OLA, ";A$;", EU SOU
O SEU COMPUTADOR"
170 END
```

EVITE MÁS TÉCNICAS DE PROGRAMAÇÃO

O excessivo emprego de **GOTO** em um programa é considerado um mau estilo de programação. Uma razão para isso é que, mesmo em programas simples, uma declaração **GOTO** que o manda de volta para uma linha precedente pode criar um laço sem fim. Ora, esse laço só poderá ser interrompido por meio do teclado, pressionando-se, para isso, as teclas **<BREAK>**, **<STOP>**, **<CTRL><C>**, etc. — ou desligando-se o computador. Isso é inconveniente, principalmente para o usuário do programa.

Mas a principal razão é que, permitindo que você salte, para trás ou para a frente, para qualquer ponto do programa, ao inteiro sabor de sua vontade, o uso do **GOTO** tende a quebrar a estrutura lógica do programa. Isso pode não parecer muito importante, se você estiver lidando com programas de 5 ou 10 linhas, mas pode ser vital, se você estiver escrevendo programas de 100 ou 1000 linhas.



Um bom estilo de programação requer que os programas sejam desenvolvidos em módulos lógicos, cada um desempenhando uma tarefa bem separada. Isso ajuda muito quando você tem que localizar erros de programação que ocorrem aleatoriamente, no momento em que o programa é rodado. Esse estilo de programação (conhecido como estruturado) o ajudará também a fazer programas mais simples, que poderão ser facilmente modificados no futuro.

COMO UTILIZAR O GOSUB

Existe outra ferramenta de programação que pode ser utilizada para substituir em grande parte o **GOTO**, quando este é problemático: é o **GOSUB**. Essa instrução é digitada com uma palavra e deve ser seguida por um número de linha (ou ainda por uma expressão numérica, em micros da linha Sinclair).

O **GOSUB** envia o computador para uma sub-rotina, que se inicia na linha especificada por ele. Uma sub-rotina é um conjunto de operações dentro de um programa que podem ser colocadas à parte, em uma espécie de "tijolo" lógico. Ela é freqüentemente utilizada quando uma operação deve ser repetida várias vezes ao longo de um programa. Assim, ao invés de escrever esta rotina toda vez que ela ocorre no programa, podemos colocá-la em um ponto qualquer, e simplesmente direcionar o computador para ela, quando necessário, e quantas vezes desejarmos.

A diferença decisiva entre um **GOTO** e um **GOSUB** é que no final de uma sub-rotina deve constar a palavra **RETURN** ("retornar", em inglês). Para entrar esta palavra, nos computadores da linha Sinclair deve-se simplesmente pressionar a tecla marcada com **RETURN**. Nos outros micros, digita-se a palavra por extenso, pressionando-se <**ENTER**> ou <**RETURN**> depois (não confunda a tecla de retorno de carro, ou transmissão, que é sempre indicada no texto de forma diferente, com a instrução **RETURN**).

A declaração **RETURN** envia o fluxo do programa de volta à linha ou ao comando que se segue ao **GOSUB** que solicitou ("chamou", na gíria dos programadores) a sub-rotina. Assim, o computador sempre "se lembra" de que ponto do programa foi efetuado o desvio para a sub-rotina e sempre volta a ele, para continuar o fluxo normal do programa, depois que a sub-rotina é completada. Desse modo, podemos colocar chamadas à mesma sub-rotina em qualquer ponto do programa.

No programa seguinte exemplificamos o uso do **GOSUB**. Ele simula um jogo de dados muito simples, uma espécie de "crepe". No jogo, um par de dados é arremessado, e o total dos pontos obtidos é anotado. Se os totais de dois arremessos sucessivos forem iguais, o jogo terminará. Caso contrário, os dados serão arremessados novamente.

TT

```
20 LET A=1
30 REM..PRIMEIRO  LANCAMENTO..
40 GOSUB 150
50 LET T1=T
60 REM..SEGUNDO LANCAMENTO..
70 GOSUB 150
80 LET T2=T
90 IF T1=T2 THEN GOTO 120
100 LET A=A+1
110 GOTO 40
120 PRINT"CONTAGENS IGUAIS A ";
T1;" APOS ";A;" LANCAMENTOS"
130 END
140 REM..SUBROTINA
150 LET D1=RND(6)
160 LET D2=RND(6)
170 LET T=D1+D2
180 RETURN
```

SS

Digite o programa abaixo usando somente maiúsculas, se o seu computador for compatível com o ZX-81.

```
20 LET a=1
```

```
 30 REM primeira jogada
 40 GOSUB 150
 50 LET t1=T
 60 REM segunda jogada
 70 GOSUB 150
 80 LET t2=T
 90 IF t1=t2 THEN  GOTO 120
100 LET a=a+1
110 GOTO 40
120 PRINT "Os dados deram ";t1
;" na jogada";a
130 GOTO 200
140 REM sobroutine
150 LET d1=INT (RND*6)+1
160 LET d2=INT (RND*6)+1
170 LET T=d1+d2
180 RETURN
```

```
10 CLS: LET D1=RND(-TIME)
20 LET A=1
40 GOSUB 150
50 LET T1=T
70 GOSUB 150
80 LET T2=T
90 IF T1=T2  THEN GOTO 120
100 LET A=A+1
110 GOTO 40
120 PRINT "CONTAGENS IGUAIS A "
T1;" APOS ";A;" LANCAMENTOS"
130 END
140 REM ... SUBROTINA
150 LET D1=INT(RND(1)*6)+1
160 LET D2=INT(RND(1)*6)+1
170 LET T=D1+D2
175 PRINT D1;D2;T
180 RETURN
```

O arremesso do dado deve ser realizado duas vezes; por isso, é entregue a uma sub-rotina, que vai das linhas 150 a 180. O **GOSUB** nas linhas 40 e 70 envia o computador ao ponto de início da sub-rotina (linha 150). O **RETURN** na linha 180 manda o micro voltar à linha 50, caso o chamado à sub-rotina tenha partido do **GOSUB** da linha 40, ou à linha 80, se tiver partido da linha 70. A linha 140 é usada para identificar o início da sub-rotina, usando a declaração **REM**, que significa apenas um comentário feito pelo programador.

Note a declaração **END** na linha 130 de todos os programas (exceto os do Sinclair, que usam um **GOTO**). Se ela não estivesse aí, o computador passaria a executar a sub-rotina diretamente e, ao atingir a declaração **RETURN**, assinalaria um erro, pois não haveria indicação prévia sobre para que linha retornar.

NÚMEROS DE LINHA INEXISTENTES

Como os micros da linha Sinclair não têm declaração **END**, utilizamos um **GOTO** seguido de um número de linha que não existe no programa, ou seja, 200.

```
130 GOTO 200
```

Em outros micros, isto daria uma mensagem de erro. O mesmo, entretanto, não acontece com a linha Sinclair: ao perceber que a linha 200 não existe, o computador interromperá providencialmente o programa, pronto para começar de novo.

Para tais micros, é melhor usar esses expedientes do que colocar

```
130 STOP
```

que é aceito pelo computador, mas que gera uma mensagem de advertência na tela. Isso poderia levar o usuário a acreditar que houve um erro de execução do programa, especialmente porque a linha 130 não é a última *linha física* do programa, pois está no meio dele, antes da sub-rotina.

Mas seja o mais cuidadoso possível quando utilizar, nesses computadores, o comando **GOTO** seguido por um número de linha inexistente.

Após **GOTO 200**, por exemplo, os micros da linha Sinclair pesquisarão todas as linhas seguintes. E se encontrarem uma instrução em alguma delas, eles a executarão (ou tentarão executá-la), independentemente do conteúdo dessa instrução. Portanto, para ficar mais seguro, o melhor é enviar o computador para a última linha física possível, que pode ser a de número 9999, nos micros da linha Sinclair.

E, para que não haja nenhuma confusão quanto ao que você está fazendo, coloque:

```
9999 REM END
```

CHAMADAS MÚLTIPLAS

É possível uma sub-rotina chamar outras, e estas, por sua vez, outras tantas, gerando assim uma estrutura em diversos níveis. Assim como no jogo citado o dado deve ser lançado duas vezes, a sub-rotina de arremesso pode ser constituída como uma sub-rotina dentro de outra, desde que se mude as últimas linhas do programa, da seguinte forma:

```
150 GOSUB 190
155 LET D1=D
160 GOSUB 190
165 LET D2=D
170 LET T=D1+D2
180 RETURN
190 LET D=RND(6)
195 RETURN
```

Digite o programa abaixo usando somente maiúsculas, se o seu computador for compatível com o ZX-81:

```
150 GOSUB 190
155 LET d1=d
160 GOSUB 190
165 LET d2=d
170 LET t=d1+d2
180 RETURN
190 LET d=INT (RND*6)+1
195 RETURN
```

```
150 GOSUB 190
155 LET D1=D
160 GOSUB 190
165 LET D2=D
170 LET T=D1+D2
180 RETURN
190 LET D=INT(RND(1)*6)+1
195 RETURN
```

Aqui, as linhas 150 e 160 enviam o computador para a sub-rotina na linha 190, que faz o dado rolar (isto é, sorteia um número aleatório inteiro, entre 1 e 6), enquanto as linhas 155 e 165 armazenam os resultados de cada lançamento separado.

Por uma questão de concisão, o computador realiza apenas o lançamento de um dado, e não de dois, separadamente. Veja a seguir:

```
150 LET T=RND(6)+RND(6)
```

```
150 LET T=INT(6*RND+1) +
    INT(6*RND+1)
```

```
150 LET T=INT(6*RND(1)+1)+
    INT(6*RND(1)+1
```

UTILIZE O COMANDO ON...GOSUB

Assim como **GOTO**, a declaração **GOSUB** pode ser utilizada para chamadas complexas a um determinado número de sub-rotinas, começando em linhas diferentes. Nos microcomputadores da linha Sinclair (compatíveis com Spectrum e ZX-81), o **GOSUB** pode ser seguido de uma expressão numérica, ou nome de variável, exatamente como vimos anteriormente para o **GOTO**.

No caso dos computadores de outras linhas, como TRS-80, Apple e MSX, existe a declaração **ON...GOSUB**, que pode ser utilizada para o mesmo efeito.

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
O que são variáveis em promoção BASIC? Cordões (*strings*) e cordões vazios, como utilizá-los em jogos eletrizantes.

CÓDIGO DE MÁQUINA
Aprenda a entrar um programa em linguagem de máquina no computador, usando um programa em BASIC e o comando **POKE**.

PROGRAMAÇÃO BASIC
Saia do mundo preto e branco e entre nos arco-íris, acrescentando cores aos desenhos nos micros TRS-Color.

APLICAÇÕES
Como encontrar nomes e endereços de pessoas a partir de poucas informações.