CURSO PRÁTICO 3 DE PROGRAMAÇÃO DE COMPUTADORES

INPUT

PROGRAMAÇÃO BASIC - PROGRAMAÇÃO DE JOGOS - CÓDIGO DE MÁQUINA

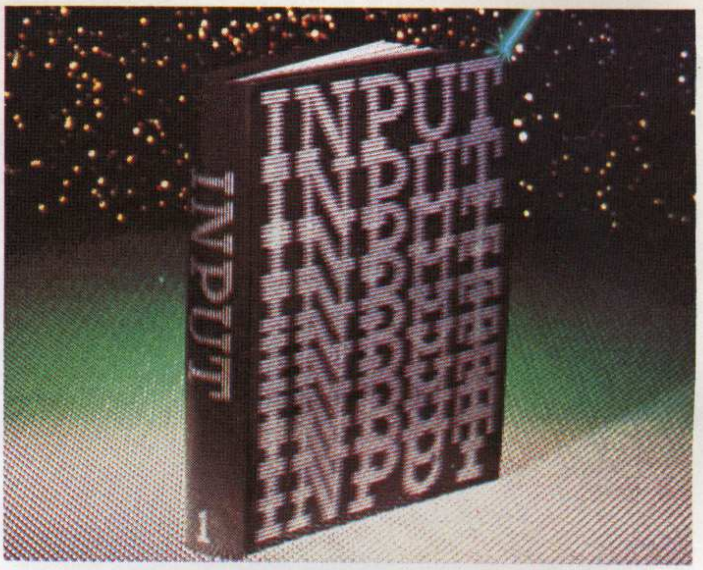
NOVA CULTURAL

Cz$ 20,00

INPUT

Vol. 1 N.º 3

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 n.º (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

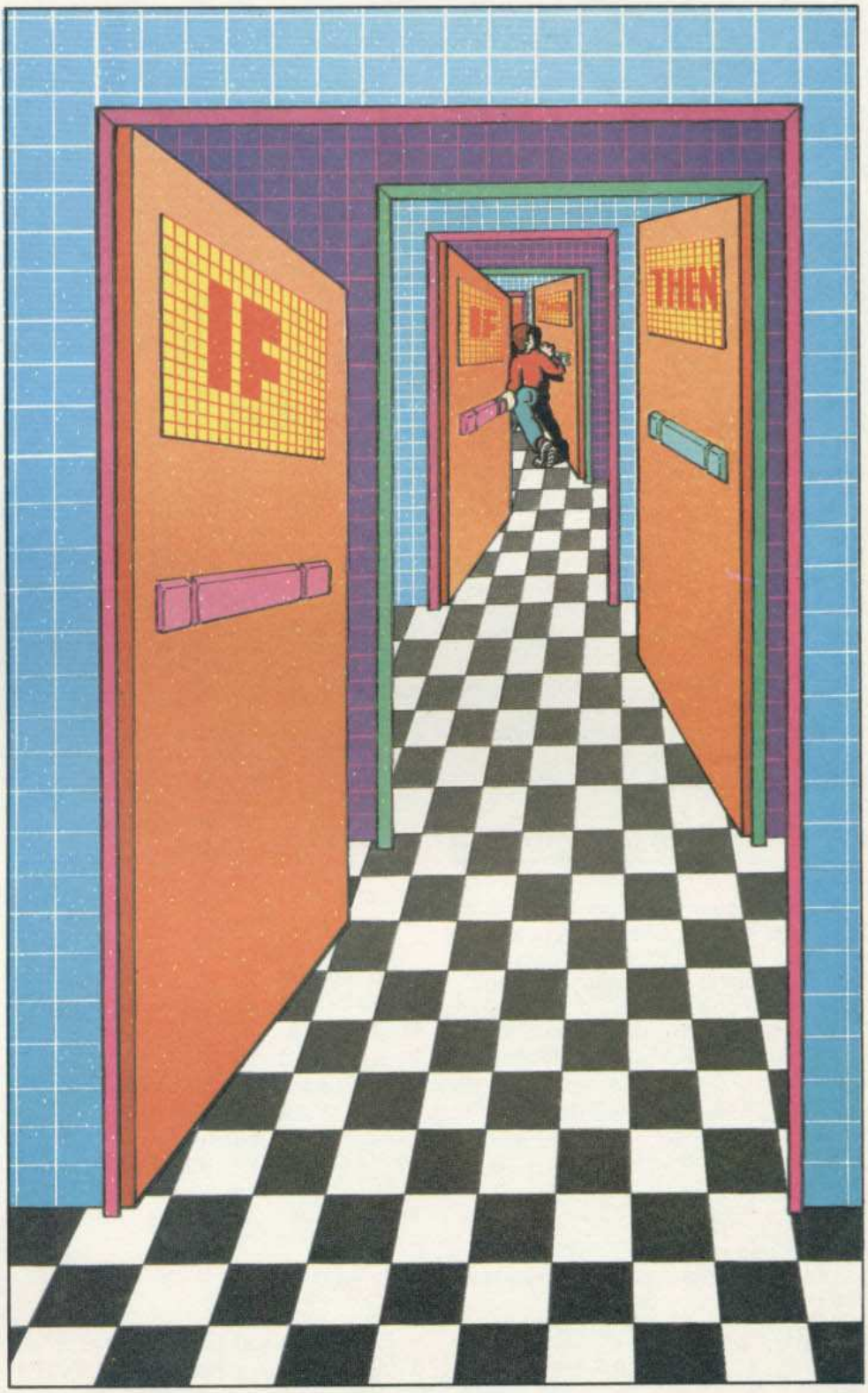
Preparadores de Texto: Ana Maria Dilguerian, Antonio Francelino de Oliveira, Karina Ap. V. Grechi, Levon Yacubian, Maria Teresa Galluzzi, Paulo Felipe Mendrone
Revisor/Coordenador: José Maria de Assis
Revisoras: Conceição Aparecida Gabriel, Isabel Leite de Camargo, Ligia Aparecida Ricetto, Maria do Carmo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

Edição organizada pela Editora Nova Cultural Ltda.
(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.

ENSINE SEU MICRO A TOMAR DECISÕES

- ESCOLHA O CAMINHO A SEGUIR
- DECISÕES MAIS COMPLICADAS
- APRENDA A UTILIZAR O COMANDO **IF...THEN...ELSE**

Embora não tenha cérebro, o computador tem capacidade para decidir de forma lógica. Aprenda a trabalhar com a instrução IF...THEN e ensine seu micro a tomar decisões corretamente.



O que torna um computador infinitamente superior a uma máquina de calcular é sua capacidade de tomar decisões baseadas em raciocínios lógicos. Um desses raciocínios é o comando **IF...THEN**. O computador reage a essa declaração de modo semelhante a um ser humano: **SE (IF)** isto é verdadeiro, **ENTÃO (THEN)** faça tal coisa. Eis um exemplo:

IF A<18 (ou seja, SE A é menor que 18) **THEN PRINT** "menor de idade" (ENTÃO IMPRIMA "menor de idade")

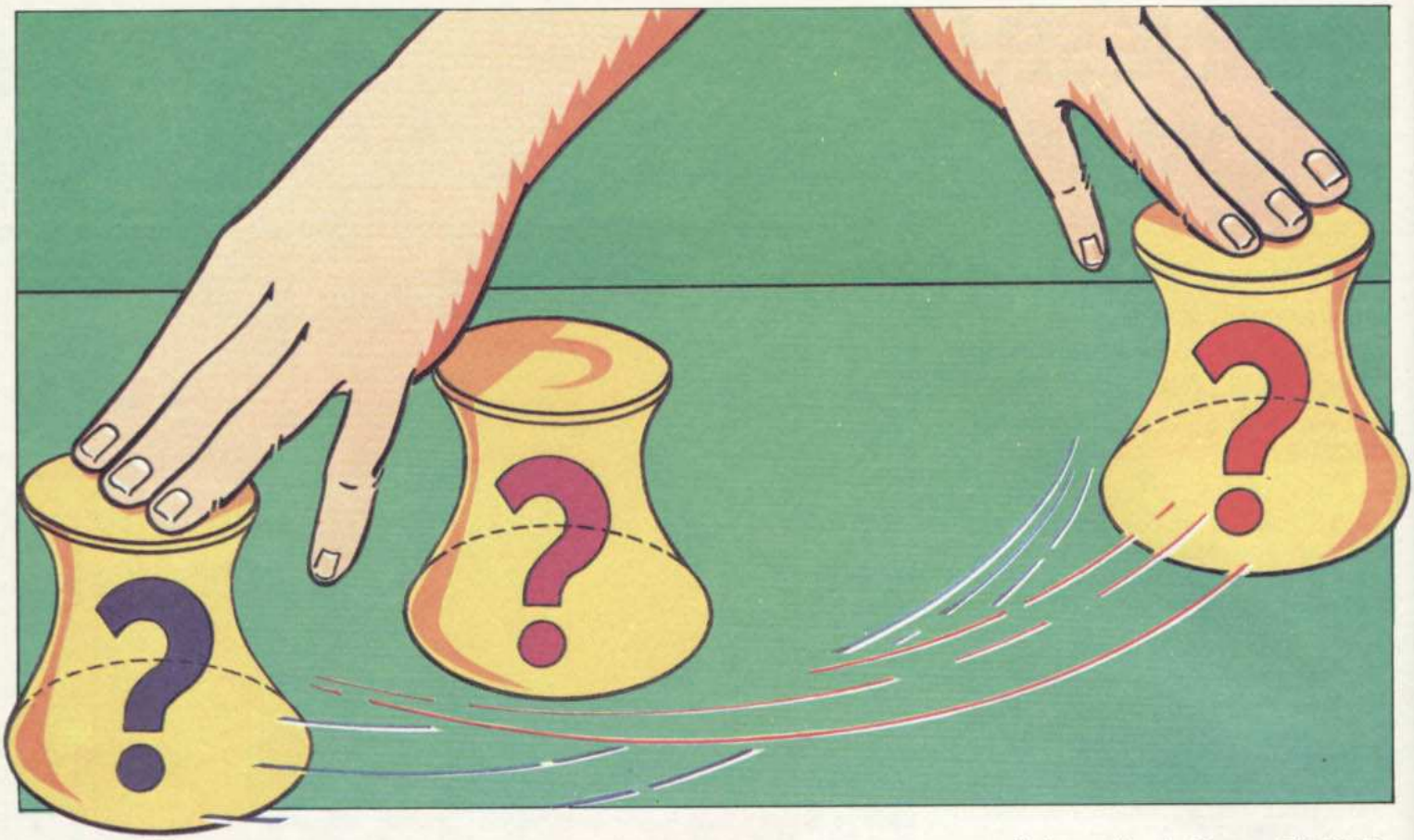
Ao encontrar a declaração **IF**, o computador verifica se a proposição que a segue é verdadeira. Se for, ele executa tudo o que estiver após o **THEN**. Se a proposição é falsa, ele ignora o resto e passa para a próxima linha do programa.

Você verá como funciona tudo isso no programa a seguir.

```
10 PRINT "Digite a lista das no
tas"
20 PRINT "Digite -99 para termi
nar"
30 INPUT N
40 IF N=-99 THEN PRINT "Média =
 ";T/C:STOP
50 LET T=T+N
60 LET C=C+1
70 GOTO 30
```

A mensagem no início do programa instrui o usuário a digitar primeiro uma lista de notas e por fim o número -99. As linhas 25 e 26 zeram o valor das variáveis do total (**T**) e do contador (**C**). A linha 30 dá entrada ao valor que você digita. A linha 50 adiciona esse valor ao total corrente, e a linha 60 conta quantos números você digitou, adicionando 1 à variável do contador cada vez que um novo número é dado.

Enquanto você estiver digitando no-



tas reais, o computador ignorará a linha 40 e seguirá em frente, mas, quando você digitar -99, a condição da linha 40 N = -99 será satisfeita, e o computador imprimirá a média das notas (**T/C**). Números como -99 são chamados de fictícios, ou terminais, e são úteis para controlar o que acontece em um programa. Este é um recurso muito usado para interromper partes de um programa.

UMA ESCOLHA COM TRÊS CAMINHOS

E se você quiser escolher entre três ou mais alternativas, de modo a colocar o computador em diferentes cursos de ação? É simples: o computador sorteia um número que você tem de adivinhar; ao comparar o seu palpite com o número sorteado, ele elabora três alternativas: o palpite foi correto, foi muito baixo, ou muito alto. Tudo se passa, portanto, como num jogo de advinhação.

TT

```
5 CLS
10 LET N=RND(20)
20 PRINT "ACABO DE IMAGINAR UM
NUMERO..."
30 PRINT "TENTE ADIVINHA-LO!"
40 INPUT G
50 IF G=N THEN PRINT "MUITO BEM
!":FOR D=1 TO 2000:NEXT D:GOTO
10
60 IF G<N THEN PRINT "ESTA BAIX
O, TENTE OUTRA VEZ!"
70 IF G>N THEN PRINT "MUITO ALT
O! TENTE DE NOVO."
80 GOTO 40
```

SS

Atenção: digite o programa abaixo em maiúsculas, se quiser executá-lo nos compatíveis com ZX-81:

```
 5 CLS
10 LET N=INT (RND*20)+1
20 PRINT "Eu acabo de imagina
r um numero."
30 PRINT "Tente adivinha-lo."
40 INPUT G
50 IF G=N THEN  PRINT "Muito
bem!": PAUSE 100: GOTO 10
60 IF G<N THEN  PRINT "Esta b
aixo; tente outra vez!"
70 IF G>N THEN  PRINT "Muito
alto! Tente de novo."
80 GOTO 40
```

MX

O programa abaixo, como está, rodará apenas nos micros tipo MSX. Para rodá-lo no Apple II e no TK-2000, elimine a linha 3 e substitua o comando da linha 5 por **HOME**.

```
3 R=RND(-TIME)
5 CLS
10 LET N=INT(RND(1)*21)
20 PRINT "Acabo de pensar e um
número"
30 PRINT "...será que você cons
egue advinhá-lo ?"
40 INPUT G
50 IF G=N THEN PRINT "Certo, mu
ito bem!":FOR D=1 TO 2000:NEXT
D:GOTO 5
60 IF G<N THEN PRINT "Muito bai
xo, tente outra vez."
70 IF G>N THEN PRINT "Alto dema
is, tente outra vez."
80 GOTO 40
```

A linha 10 escolhe um número ao acaso entre 1 e 20, e as linhas 20 a 40 convidam você a adivinhá-lo. Qualquer que seja sua resposta, o computador verifica, nas linhas 50, 60 e 70, que proposição é verdadeira.

Suponha que seu palpite seja um número abaixo do sorteado. Neste caso, o computador vai primeiro para a linha 50; mas P = N é falso, de modo que ele ignora o resto da linha e passa para a

seguinte, a linha 60. Agora, a proposição é verdadeira, visto que P<N. Então, ele imprime a mensagem "Muito baixo, tente novamente". A seguir, ele passa assim mesmo para a próxima linha, onde a proposição é falsa, de forma que ele a ignora e vai para a 80, que leva você de volta para outra tentativa.

Este programa funciona muito bem, mas tem uma desvantagem: ele continua a pedir um novo palpite ao jogador, sem se importar se este quer ou não continuar jogando. Seria interessante fazer com que o computador preguntasse se você quer fazer uma nova tentativa.

As próximas linhas fazem exatamente isso. O comando **IF...THEN** será novamente utilizado, só que desta vez o computador estará comparando letras e não números.

Antes de entrá-las em adição ao programa anterior, não esqueça de mudar a linha 50 para:

```
50 IF G=N THEN PRINT "CERTO,
    MUITO BEM!":GOTO 100
```

```
100 PRINT "QUER JOGAR DE NOVO?(
S/N)"
110 LET A$=INKEY$:IF A$=""THEN
GOTO 110
120 IF A$="S" THEN RUN
```

```
100 PRINT "Voce quer tentar ou
tra vez(S/N)?"
110 LET A$=INKEY$: IF A$<>"S"
AND A$<>"N" THEN  GOTO 110
120 IF A$="S" THEN  RUN
```

Atenção: para poder rodar o programa abaixo em computadores tipo Apple e TK-2000, substitua a linha

```
110 GET A$
```

```
100 PRINT ,,"Quer jogar outra v
ez (S/N)?"
110 LET A$=INKEY$: IF A$="" THE
N GOTO 110
120 IF A$="S" OR A$="s" THEN GO
TO 5
```

A linha 110 espera que você acione uma tecla. Caso seja o "S" maiúsculo, o programa continuará. Se qualquer outra tecla for acionada, o programa terminará.

CHECAGEM DUPLA

Para fazer o computador testar duas ou mais condições, antes de decidir que caminho tomar, use palavras especiais chamadas *operadores lógicos:*

```
100 IF D$="DOMINGO" AND H=20
    THEN PRINT "E' HORA DO
    FANTASTICO"
```

Quando você usa **AND** (a conjunção *e*, em inglês) entre duas condições, ambas devem ser verdadeiras para que o computador execute o resto da linha. Se uma delas ou as duas forem falsas, ele passará para a próxima linha. Nesse exemplo é necessário que sejam 20 horas de um domingo para que o computador imprima mensagem. Veja outro exemplo:

```
200 IF P$="SAGU" OR
    P$="TAPIOCA" THEN PRINT
    "NAO ESTOU COM FOME"
```

Essa linha usa **OR** (*ou*, em inglês) e o computador imprimirá a mensagem no caso de pelo menos uma das comparações ser verdadeira.

O teste pode vir a ser muito complicado se houver uma quantidade muito grande de condições para verificar. Se você tem vários AND e OR juntos, terá que usar parênteses para que o computador saiba o que verificar primeiro.

Uma linha de um jogo de aventuras pode se parecer com isto:

```
2000 IF P=14 AND (C$="ESPADA"
     OR C$="FACA") THEN PRINT
     "VOCE DESTRUIU O GREMLIN"
```

Isso só é verdadeiro se você estiver na posição 14 E com uma espada OU uma faca. Mude os parênteses:

```
2000 IF (P=14 AND C$="ESPADA")
     OR C$="FACA" THEN PRINT
     "VOCE DESTRUIU O GREMLIN"
```

Agora ela será verdadeira se você estiver na posição 14 com uma espada OU se estiver em qualquer lugar com uma faca.

TIRE A SORTE GRANDE

Agora, usando os conceitos que aprendemos, vamos programar um joguinho que simula uma máquina caça-níqueis. Ela faz um bom uso de **AND** e **OR**. Tente tirar a sorte grande!

```
20 LET M=50
30 CLS
40 LET M=M-5
50 IF M<0 THEN PRINT "DESCULPE.
.. VOCE ESTA DURO!":END
60 LET A=RND(12)+192
70 LET B=RND(12)+192
80 LET C=RND(12)+192
210 PRINT @237,CHR$(A):PRINT@23
9,CHR$(B):PRINT @241,CHR$(C)
220 IF A=B AND B=C THEN PRINT @
258,"VOCE GANHOU $50!!!":LET M=
M+50
230 IF (A=B  OR B=C) AND A<>C T
HEN PRINT @361,"VOCE GANHOU $10
!":LET M=M+10
240 FOR D=1 TO 500:NEXT
250 PRINT @323,"QUER JOGAR OUTR
A VEZ?(S/N)":PRINT @361,"VOCE T
EM $";M
```

MICRO DICAS

OS OPERADORES LÓGICOS

Se você não está acostumado com os símbolos usados em matemática para expressar relações lógicas, como "maior que" e "menor que" — os chamados *operadores* — pode achá-los meio confusos, no início.

Assim, pense neles como uma espécie de cunha. No símbolo > o lado mais aberto, à esquerda, é *maior do que* o lado mais agudo (a ponta). No símbolo < o lado mais estreito, á esquerda, é *menor do que* o lado direito. Deste modo, a expressão A>B deve ser lida simplesmente: A é maior do que B. Colocar o sinal de igualdade (=) ao lado de um dos operadores prévios significa também que pode ser "maior ou igual a" ou "menor ou igual a".

Eis aqui a lista de todas as combinações possíveis:

```
A=B   : A é igual a B
A>B   : A é maior que B
A<B   : A é menor que B
A>=B  : A é maior ou igual a B
A<=B  : A é menor ou igual a B
A<>B  : A é diferente de B
```

Alguns computadores exigem que a ordem de digitação dos operadores compostos seja sempre a mesma. Por exemplo, não se pode digitar >= ou => indistintamente: eles não vão aceitar. Em outros micros, o esquema é mais liberal.

Nos computadores da linha Sinclair os operadores compostos são digitados através de uma única tecla. Dessa forma, se você tentar entrar o sinal < seguido do sinal =, por exemplo, a linha gera um erro de sintaxe.

Finalmente, alguns computadores aceitam também o sinal # (sustenido), para significar "diferente de".

Posso combinar dois ou mais IF...THEN em uma única linha?

Sim, mas esta não é uma idéia muito boa, pois dificulta muito a compreensão de como funciona um programa e pode gerar erros lógicos.

O princípio básico da declaração **IF...THEN** é que o que vem depois do **THEN** na linha do **IF**, só será executado se a proposição for verdadeira. Assim, na linha:

```
70 IF X=Y THEN PRINT "NAVE
   DESTRUIDA"  :   LET NAVES=
   NAVES-1:GOTO 30
```

Nenhuma das instruções após o **THEN** será executada, se X for diferente de Y. Mas, algumas vezes o **IF** composto pode ser útil:

```
70 IF X=Y THEN PRINT "NAVE
   DESTRUIDA": IF NAVES>0 THEN
   LET NAVES=NAVES-1

80 IF X=Y AND NAVES=0 THEN
   PRINT "FIM DO JOGO":END
```

Note como os dois **IFs** na linha 70 simplificam a programação.

Por que ocorrem erros quando tento rodar programas digitados?

Pode ser que existam falhas no programa em si, mas os erros mais comuns são de digitação, quando se está copiando. Eis alguns deles (lembre-se de checá-los um a um, quando estiver copiando programas):

— Confundir a letra maiúscula I ou a letra L minúscula com o número 1; ou ainda a letra O maiúscula com o zero.

— Confundir parênteses com os sinais de maior (>) e menor (<). Confundir o cifrão ($) com a letra S, ou omiti-los.

— Omitir as aspas no começo ou no fim, em comando **PRINT** ou **LET**. Omitir os dois pontos (:) entre instruções na mesma linha.

— Esquecer a vírgula entre dois ítens de uma declaração **DATA**, o que pode determinar um número muito grande para o computador, ou a falta de um item quando o computador chega ao final dos **DATA**.

— Omitir um sinal de menos (em um programa que gera gráficos na tela, isso pode resultar em uma tentativa de desenhar além dos seus limites).

— Omitir um sinal de ponto e vírgula no final de uma linha **PRINT** ou colocar um número de espaços em branco maior ou menor do que o previsto.

```
260 LET K$=INKEY$:IF K$="" THEN
 GOTO 260
270 IF K$="S" THEN GOTO 30
280 END
```

```
 20 LET M=50
 30 CLS
 40 LET M=M-5
 50 IF M<0 THEN  PRINT "Desculpe... Voce esta duro!": STOP
 60 LET A=INT (RND*12)+130
 70 LET B=INT (RND*12)+130
 80 LET C=INT (RND*12)+130
210 PRINT  PAPER 0; INK 4;AT 10,14;CHR$ A;AT 10,16;CHR$ B;AT 10,18;CHR$ C
220 IF A=B AND B=C THEN PRINT AT 13,2;"Voce ganhou $50!!!": LET M=M+50
230 IF (A=B OR B=C) AND A<>C THEN  PRINT AT 13,9;"Voce ganhou $10!": LET M=M+10
240 PAUSE 25
250 PRINT AT 15,6;"Quer jogar outra vez?(S/N)": PRINT TAB 10;"Voce tem $";M
260 IF INKEY$="" THEN  GOTO 260
270 IF INKEY$="n" THEN  STOP
280 GOTO 30
```

```
20 LET M=50
30 CLS
40 M=M-5
50 IF M<0 THEN PRINT "Desculpe, seu dinheiro acabou.":END
60 LET A=INT(RND(1)*5+66)
70 LET B=INT(RND(1)*5+66)
80 LET C=INT(RND(1)*5+66)
90 LOCATE 17,10:PRINT CHR$(1)+CHR$(A);" ";CHR$(1)+CHR$(B);" ";CHR$(1)+CHR$(C)
220 IF A=B AND B=C THEN LOCATE 2,16:PRINT " Você ganhou o prêmio máximo ... $50":LET M=M+50
230 IF (A=B OR B=C) AND A<>C THEN LOCATE 10,16:PRINT "Você ganhou $10":LET M=M+10
240 FOR D=1 TO 500:NEXT
250 LOCATE 5,18:PRINT " Quer tentar outra vez ? (S/N)"
260 LOCATE 12,20:PRINT "Você tem $";M;"    ";
270 LET K$=INKEY$:IF K$="" THEN GOTO 270
280 IF K$="s" OR K$="S" THEN GOTO 30
290 END
```

Como o Apple II não tem caracteres gráficos pré-definidos, a versão a seguir faz na tela blocos coloridos (símbolos do caça-níqueis):

```
20  LET M = 50
30  HOME : GR
40  LET M = M - 5
50  IF M < 0 THEN  VTAB 22: PRINT "DESCULPE, VOCE ESTA DURO": END
60  LET A =  INT ( RND (1) * 12 + 4)
70  LET B =  INT ( RND (1) * 12 + 4)
80  LET C =  INT ( RND (1) * 12 + 4)
85  COLOR= A: VLIN 0,39 AT 16
90  COLOR= B: VLIN 0,39 AT 18
95  COLOR= C: VLIN 0,39 AT 20
110  IF A = B AND B = C THEN  VTAB 22: PRINT "VOCE  GANHOU $50": LET M = M + 50
120  IF (A = B OR B = C) AND A <  > C THEN  VTAB 22: PRINT "VOCE GANHOU $10": LET M = M + 10
130  FOR I = 1 TO 500: NEXT I
140  VTAB 22: PRINT "OUTRA VEZ? (S/N)?": PRINT "VOCE TEM $";M
150  GET A$
160  IF A$ = "N" THEN  END
170  GOTO 30
```

A versão para o TK-2000 usa os caracteres do teclado. Só as linhas que serão substituídas no programa acima aparecem abaixo:

```
30  HOME
60  LET A =  INT ( RND (1) * 12 ) + 222
70  LET B =  INT ( RND (1) * 12 ) + 222
80  LET C =  INT ( RND (1) * 12 ) + 222
90  VTAB 10
100  PRINT  TAB( 18); CHR$ (242 ); CHR$ (A); TAB( 209; CHR$ (242); CHR$ (B); TAB( 22); CHR$ (242); CHR$ (C)
```

O programa usa várias linhas **IF...THEN**. A primeira, de número 50, checa se você tem dinheiro para jogar. Se você tem, a linha é ignorada, mas, se não, uma mensagem é mostrada, e o jogo termina.

As linhas 60 a 80 escolhem três números aleatórios, e a linha 210 converte esses números em caracteres (no caso dos computadores das linhas TRS-Color, Spectrum e MSX, naipes de baralho) e os coloca no centro da tela. O Sinclair e o TRS-Color convertem esses números em caracteres. Os outros necessitam de ajustes.

Na linha 220, se os três caracteres são iguais, você ganha o *jack pot* (maior prêmio) e seu dinheiro é aumentado em $50. Na linha 230 você ganha $10 se dois caracteres adjacentes são iguais (A = B ou B = C), mas não ganha nada se os dois caracteres das extremidades são iguais. Se você não ganhar nada, o computador ignora as linhas 220 e 230 e pas-

sa para a 240, que provoca uma breve pausa.

As linhas seguintes são uma outra versão da rotina "sim/não" que checam se você quer uma nova jogada.

IF...THEN...ELSE

Em alguns computadores você pode escrever a instrução **IF** de uma forma muito mais poderosa, chamada de **IF...THEN...ELSE**. Veja um exemplo:

```
10 INPUT IDADE
20 IF IDADE<18 THEN PRINT
   "MENOR DE IDADE" ELSE PRINT
   "ELEGIVEL"
```

Esta linha faz exatamente o que diz. Se você tem menos de 18 anos, o computador imprime "MENOR DE IDADE", mas, se você tiver 18 ou mais, ele imprimirá "ELEGÍVEL"

Dos computadores que apresentam a declaração **IF...THEN...ELSE** em seu repertório de BASIC apenas os das linhas TRS-80 e MSX estão ilustrados aqui.

Com a instrução **IF...THEN...ELSE** o programa fica mais fácil de ler e escrever. No entanto, ela não é essencial. Você pode escrever programas sem ela se o seu computador usa apenas **IF...THEN**. Existem duas maneiras de contornar o problema. Uma emprega **IF...THEN** seguido de **GOTO** para pular para a parte requerida do programa:

```
10 INPUT IDADE
20 IF IDADE<18 THEN PRINT
   "MENOR DE IDADE":GOTO 30
25 PRINT "ELEGIVEL"
30 ...restante do programa
```

Esse tipo de programação serve para os micros que permitem *comandos múltiplos por linha*, isto é, a colocação de mais de uma instrução na linha do **IF**. Os compatíveis com o ZX-81 não contam com essa possibilidade. A segunda solução do problema apóia-se num **IF...THEN** extra para assegurar que todas as possibilidades sejam cobertas:

```
10 INPUT IDADE
20 IF IDADE<18 THEN PRINT
   "MENOR DE IDADE"
25 IF IDADE >=18 THEN PRINT
   "ELEGIVEL"
30 ...restante do programa
```

Como dois **IF**s demoram mais tempo do que um só para serem executados (e isso é importante quando o mesmo trecho de programa deve ser executado muitas vezes), outro estilo de programação de alternativas é:


```
10 INPUT IDADE
20 IF IDADE>=18 THEN GOTO 25
22 PRINT "MENOR DE IDADE"
24 GOTO 30
25 PRINT "ELEGIVEL"
30...resto do programa
```

Nas duas últimas programações, é mais difícil entender o fluxo de desvios do que o programa com apenas uma linha **IF...THEN...ELSE**.

DIVIRTA-SE COM LABIRINTOS

Jogos com labirintos complicados exigem programas muito longos. Mas você pode construir alguns bem simples com pouco mais do que um laço e declarações DATA.

Jogos com labirintos exerceram sempre um grande fascínio sobre usuários de computadores, e variações do tão conhecido "Come-come" (*PacMan*) continuam a aparecer constantemente no mercado.

Este artigo lhe ensina a entrar na onda e fazer um emocionante jogo de labirinto.

O labirinto, nesta primeira tentativa, não inclui "inimigos" ou obstáculos, que exigiriam um programa bem maior. Mas ele mostra como fazer para que os personagens do seu jogo não atravessem as paredes e inclui também cronometragem, manutenção de um placar e uma rotina de registro de recordes, para dar mais competitividade ao jogo.

A maneira mais fácil de compreender como os jogos com labirinto funcionam nos micros da linha Sinclair Spectrum, como o TK90X, é ir por etapas:

```
100 FOR n=3 TO 17
110 READ a$
120 FOR m=7 TO 21
130 PRINT AT n,m;"."
140 IF a$(m-6)="p" THEN  PRINT
 PAPER 1; INK 1;AT n,m;" "
150 NEXT m
160 NEXT n
9000 DATA "pppppppppppppp"
9010 DATA "p............p"
9020 DATA "p.pp.pp.pp.pp.p"
9030 DATA "p.p........p.p"
9040 DATA "p...p.p.p...p"
9050 DATA "p.ppp.p.p.ppp.p"
9060 DATA "p.....p.p.....p"
9070 DATA "pppp.pp.pp.pppp"
9080 DATA "p.....p.p.....p"
9090 DATA "p.ppp.p.p.ppp.p"
9100 DATA "p...p.p.p...p"
9110 DATA "p.p........p.p"
9120 DATA "p.pp.pp.pp.pp.p"
9130 DATA "p............p"
9140 DATA "pppppppppppppp"
```

As linhas 100, 120, 150 e 160, usando um par dos nossos já conhecidos laços **FOR...NEXT**, determinam os limites do labirinto. A linha 130 imprime um ponto em cada posição da tela dentro desses limites.

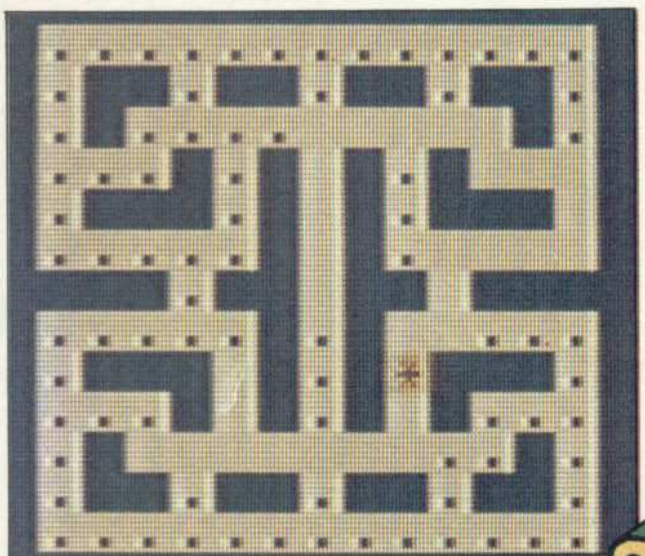
As linhas 110 e 140 lêem os dados das linhas 9000 a 9140 e substituem o ponto por um espaço em fundo azul, onde o padrão dos dados mostrar um "p".

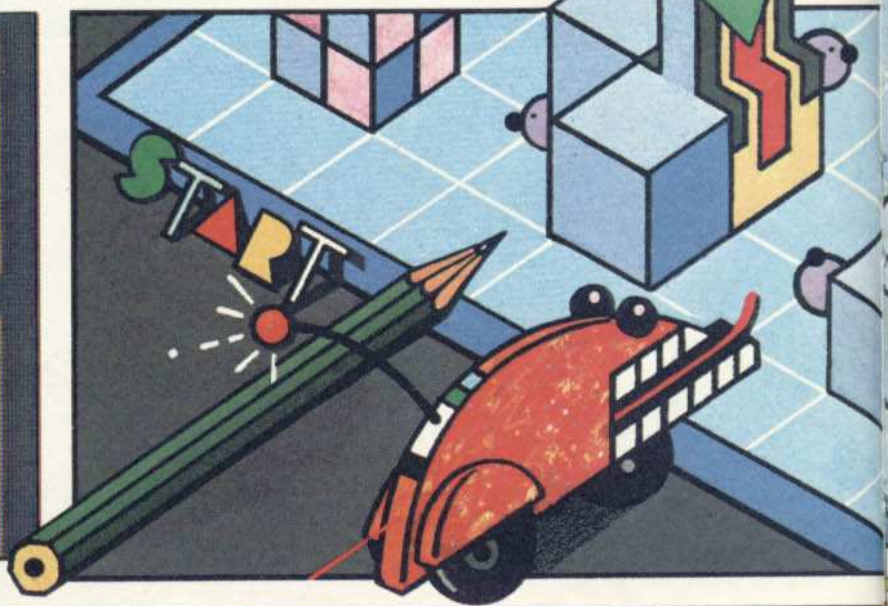
Isto é possível porque o TK90X nunca coloca dois caracteres na mesma posição; então, o último toma o lugar do primeiro.

O "COME-COME"

Um labirinto é inútil se não há ninguém ou algo para andar dentro dele. Então execute o programa acima para ver como fica e depois acione as seguintes linhas:

```
  50 LET x=10
  60 LET y=14
1000 PRINT  PAPER 6; INK 2;AT x
,y;"*"
1010 LET xx=x
1020 LET yy=y
1030 IF INKEY$="" THEN  GOTO 10
30
1040 IF INKEY$="w" AND ATTR (x-
1,y)<>9 THEN  LET x=x-1
1050 IF INKEY$="z" AND ATTR (x+
1,y)<>9 THEN  LET x=x+1
1060 IF INKEY$="a" AND ATTR (x,
```





Labirinto no TK-90X: com um programa bem curto.

- OS PRINCÍPIOS DA CRIAÇÃO DE UM LABIRINTO
- CONSTRUA UM LABIRINTO COM LAÇOS E COMANDOS **DATA**
- CONTROLE O TEMPO E O PLACAR
- COMO MOVER UMA FIGURA PELO LABIRINTO
- FAÇA PAREDES INTRANSPONÍVEIS

```
y-1)<>9 THEN  LET y=y-1
1070 IF INKEY$="s" AND ATTR (x,
y+1)<>9 THEN  LET y=y+1
1080 PRINT  INK 7;AT xx,yy;" "
1090 GOTO 1000
```

Como todos os personagens famosos dos jogos, nosso amigo asterisco é posicionado em x,y e se movimenta, aos seus comandos, por uma série de linhas usando **INKEY$**.

Aqui, no entanto, há uma determinada diferença vital: ele só pode mover-se para uma posição se esta não for azul. Ou seja, se **ATTR** não for igual a 9, sendo 9 o valor numérico da cor com a qual você imprimiu as linhas do seu labirinto.

No caso de você querer criar um labirinto de outra cor, veja como chegar ao valor de **ATTR**:

1. Tome o valor da cor do caractere (**INK**), como está no teclado do computador — neste caso 1.

2. Tome o valor da cor da tela (**PAPER**) — neste caso 1 — e multiplique por 8. Resultado, 8.

3. Adicione 64 se a área em questão é clara (**BRIGHT**). Neste caso, resultado 0.

4. Adicione 128 se a área está piscando (**FLASH**). No caso, resultado 0.

Some então todos os números. O resultado será o valor de **ATTR**.

A função **ATTR** no TK90X tem, entretanto, outros usos além de impedir que o seu personagem atravesse paredes sólidas.

Em um labirinto maior, você poderia usar a mesma idéia para fazer com que ele exploda ao entrar em uma zona proibida. Para um labirinto que use caracteres (**INK**) vermelhos sobre fundo (**PAPER**) vermelho, você necessitaria uma linha começando com:

```
IF ATTR=18 THEN...
```

As linhas 1010, 1020 e 1080 definem a posição que o personagem acabou de deixar e apagam o ponto que ele "comeu", imprimindo um espaço. Estas são linhas úteis em qualquer jogo, mas note onde elas aparecem — as primeiras linhas, antes das **INKEY$**, responsáveis pela ação, e a linha final, logo após estas.

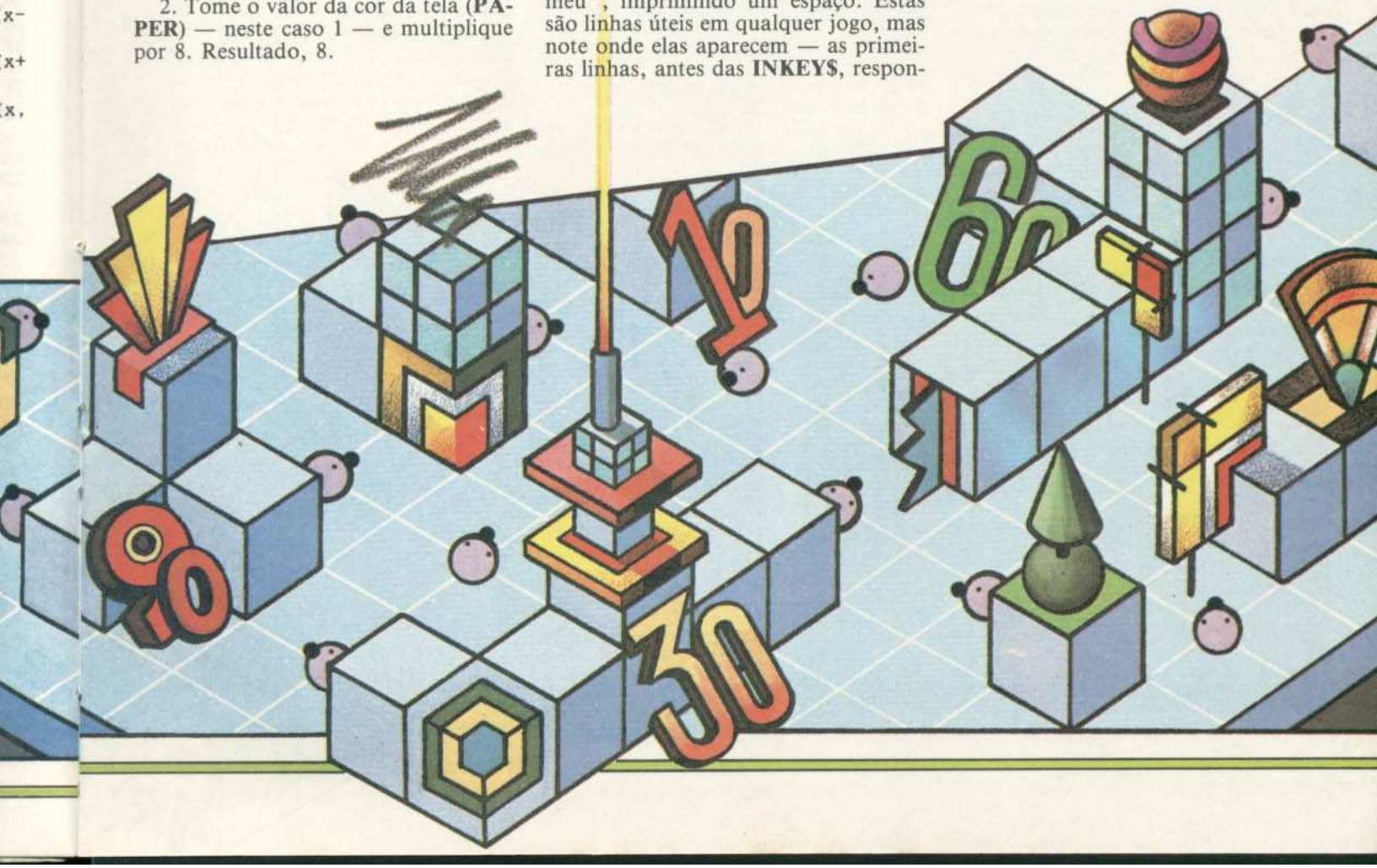
A linha 1090 é temporária e será reescrita depois. Seu propósito é permitir que você rode o programa e o teste. Sem fechar o laço, o caractere se moveria apenas uma vez.

TEMPO E PLACAR

Para que esse jogo possa ser jogado, sem contar com "inimigos" que exigiriam um programa muito complexo, o melhor é uma rotina de cronometragem e placar.

Adicione estas linhas:

```
  10 LET bt=100000
  40 LET s=0
 990 POKE 23672,0: POKE 23673,0
1025  IF s=110 THEN  GOTO 2000
1090 IF ATTR (x,y)<>63 THEN  LE
T s=s+1: SOUND .005,10
```



```
2000 LET t=(PEEK 23672+256*PEEK
 23673)/50
2010 PRINT AT 1,6;t;" SEGUNDOS
"
2020 IF t<bt THEN  LET bt=t
2030 PRINT AT 19,4;"Melhor temp
o: ";bt;" segundos"
```

Acrescentando uma linha temporária. Lembre-se de apagá-la depois:

```
1100 GOTO 1000
```

O placar é muito simples. A linha 40 coloca a contagem inicial em zero. A linha 1090 adiciona 1 cada vez que o "Come-come" devora mais um ponto, ou melhor, sempre que ele cruza um espaço em que **INK** e **PAPER** são ambos brancos — **ATTR** novamente.

E a linha 1025, que entra em ação quando todos foram comidos, faz com que o computador verifique quanto tempo o jogador levou para chegar a esse ponto.

Já a cronometragem é um pouco mais complicada. Até que você compreenda os comandos **PEEK** e **POKE**, que serão explicados num artigo próximo, procure assimilar o procedimento que se segue.

Em resumo, a linha 990 zera o relógio do TK90X ao jogar o valor 0 num endereço de memória apropriado. A linha 2000 conta o número de linhas formadas pela tela desde que o jogo começou e então divide por 50 — o número de linhas por segundo em um televisor.

Ao mesmo tempo, a linha 10 determina o "melhor tempo" inicial em 100000, muito mais que qualquer um levaria, de forma que ele será batido. A linha 2020 compara o novo tempo com o "melhor tempo".

OUTRA VEZ?

Para dar ao jogador a chance de outra tentativa, use estas novas linhas. Primeiro, pressione <**CAPS SHIFT**> e <**BREAK**> e, então, <**ENTER**>:

```
2040 PRINT AT 20,2;"Pressione q
ualquer tecla para  jogar de n
ovo.";
2050 IF INKEY$<>"" THEN  GOTO 2
050
2060 IF INKEY$="" THEN  GOTO 20
60
2070 RESTORE
2080 GOTO 40
```

OUTROS LABIRINTOS

Se você quiser tentar outros labirintos do mesmo tamanho, pode fazê-lo, mudando o padrão de letras "p" das linhas **DATA**, da linha 9000 em diante. Se fizer isso, assegure-se de ter dados suficientes para completar o labirinto ou terá uma mensagem de erro antes que apareça na tela.

Estas primeiras linhas formam o labirinto propriamente dito:

```
30 CLS4
40 LET P=297
1000 FOR N=0 TO 15
1010 READ A$
1020 PRINT @(N*32)+6,A$;
1030 FOR M=0 TO 18
1040 IF PEEK(1024+(N*32)+6+M)=6
5 THEN POKE(1024+(N*32)+6+M),17
5
1050 NEXT M
1060 NEXT N
2000 DATA AAAAAAAAAAAAAAAAAAAA
2010 DATA A.......AAA.......A
2020 DATA A.AA.AA.....AA.AA.A
2030 DATA A.A.....AAA.....A.A
2040 DATA A...A.A.....A.A...A
2050 DATA A.AAA.AA.A.AA.AAA.A
2060 DATA A...A.........A...A
2070 DATA AAA...A.AAA.A...AAA
2080 DATA AAA...A.AAA.A...AAA
2090 DATA A...A.........A...A
2100 DATA A.AAA.AA.A.AA.AAA.A
2110 DATA A...A.A.....A.A...A
2120 DATA A.A.....AAA.....A.A
2130 DATA A.AA.AA.....AA.AA.A
2140 DATA A.......AAA.......A
2150 DATA AAAAAAAAAAAAAAAAAAAA
```

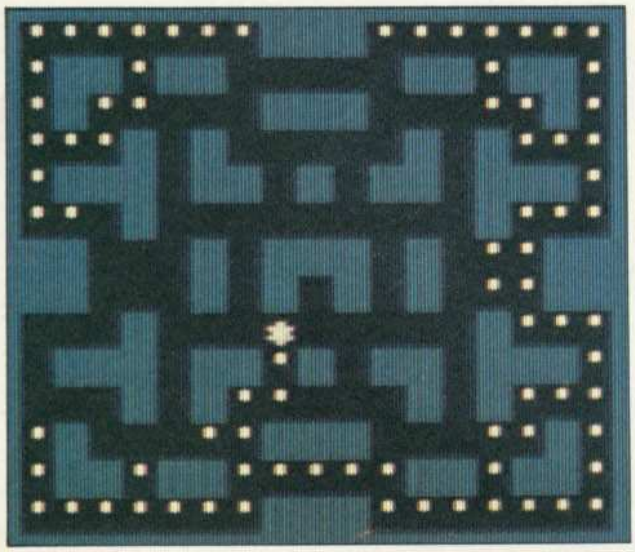
A forma do labirinto está contida nas linhas de dados (**DATA**), de 2000 a 2150. Nesse estágio os limites dele são representados por letras "A".

Antes que os dados sejam lidos, a linha 30 limpa a tela e a coloca na cor vermelha, cujo código é 4. A linha 40, por sua vez, coloca o asterisco na posição inicial.

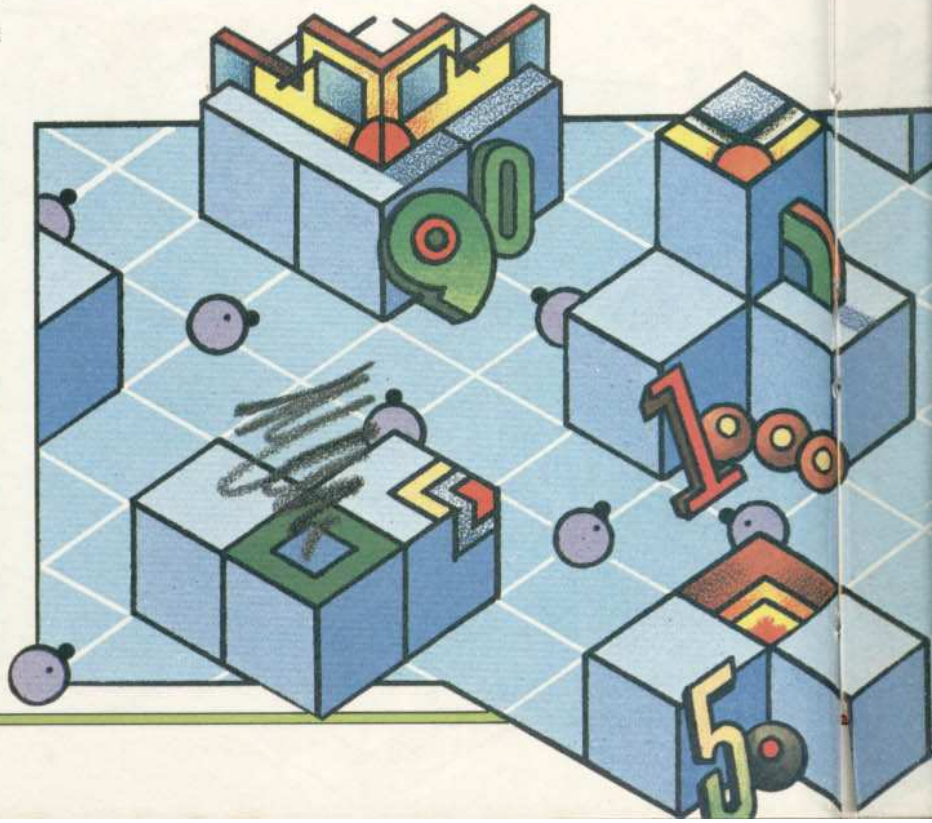
As linhas de 1000 a 1060 desenham o labirinto na tela. Ao mesmo tempo, sempre que o laço **FOR...NEXT** das linhas 1000 e 1060 é executado, o programa lê a próxima linha de dados (**DATA**).

A linha 1010 lê os dados e os chama de A$. A linha 1020 coloca-os na tela.

A linha 1040 verifica a porção da memória da máquina que corresponde a cada posição de tela. Se o endereço de memória contém o número 65, temos um "A" na tela, e ele é mudado para um bloco azul (código 175). As linhas 1030 e 1050 fornecem o *loop* com que



A versão para o TRS-Color mostra o tempo empregado.



a linha 1040 verifica todas as posições da tela.

Agora adicione essas linhas e você poderá deslocar um homenzinho pelo labirinto.

```
1080 PRINT @P,"*";
1090 LET LP=P
1100 IF PEEK(340)=247 AND PEEK(
1023+P)<>175 THEN LET P=P-1
1110 IF PEEK(338)=247 AND PEEK(
1025+P)<>175 THEN LET P=P+1
1120 IF PEEK(338)=251 AND PEEK(
992+P)<>175 THEN LET P=P-32
1130 IF PEEK(342)=253 AND PEEK(
1056+P)<>175 THEN LET P=P+32
1150 PRINT @LP," ";
1170 GOTO 1080
```

Esta parte do programa permite que você mova o seu caractere pelo labirinto. Ela tem algumas particularidades. O homem só pode ser movimentado se o ponto para onde se quer levá-lo não contém um bloco azul. A linha 1100 verifica o conteúdo de duas posições de memória: a que corresponde à tecla "Z" e a que corresponde à posição de tela em que o homem vai entrar. Se a tecla "Z" (código 223) está sendo pressionada e o endereço de tela não contém um bloco azul (código 175), então ele pode se movimentar.

As teclas para controle de movimento são as mesmas já utilizadas: "Z" para a esquerda, "X" para a direita, "P" para cima e "L" para baixo. Tente andar com o asterisco pelo labirinto de forma que os pontos sejam apagados. Acrescente estas linhas e marque o tempo necessário para "comer" os pontos:

```
20 LET FA=999999
50 RESTORE
60 LET D=0
1070 TIMER=0
1140 IF PEEK(1024+P)=110 THEN P
LAY "T80B":LET D=D+1
1160 IF D=153 THEN GOTO 1180
1180 PRINT @P," ";
1190 TI=TIMER/50
1200 PRINT @52,"TEMPO=";
1210 PRINT @84,TI;"SEG ";
1220 IF FA>TI THEN FA=TI
1230 PRINT @212,"MELHOR TEMPO";
1240 PRINT @244,FA;"SEG ";
1250 PRINT @340,"PRESSIONE";
1260 PRINT @372,"<ENTER>";
1270 PRINT @404,"PARA COMECAR";
1280 PRINT @436,"DE NOVO";
1290 LET IN$=INKEY$:IF IN$="" T
HEN GOTO 1290
1300 IF IN$=CHR$(13) THEN GOTO
40
1310 GOTO 1290
```

Quando você executa este programa pode verificar à direita do labirinto o tempo gasto para remover os pontos e também o melhor tempo.

Inicialmente, o melhor tempo é colocado em 999999 segundos pela linha 20. A linha 60 zera o placar, e a linha 1070 zera o "cronômetro".

A linha 1140 examina a posição em que o asterisco está, para ver se há um ponto. Em caso positivo, o comando **PLAY** é usado para emitir um "*blip*" enquanto o ponto é apagado. Ela também adiciona um ponto a D, que contém o número de pontos "comidos".

A linha 1160 verifica se não há mais pontos na tela. Se estes terminaram, o programa pula para a linha 1180, que apaga o asterisco.

A declaração **TI = TIMER/50** na linha 1190 pára o relógio e divide o que está sendo lido por 50, para transformá-lo em segundos.

As linhas 1200 e 1210 mostram o tempo na tela.

A linha 1220 checa se o melhor tempo (FA) é maior que o obtido na última contagem. Se for, ele é igualado a este. As linhas 1230 e 1240 mostram o melhor tempo, antes que uma mensagem oriente o jogador a teclar <**ENTER**> para recomeçar.

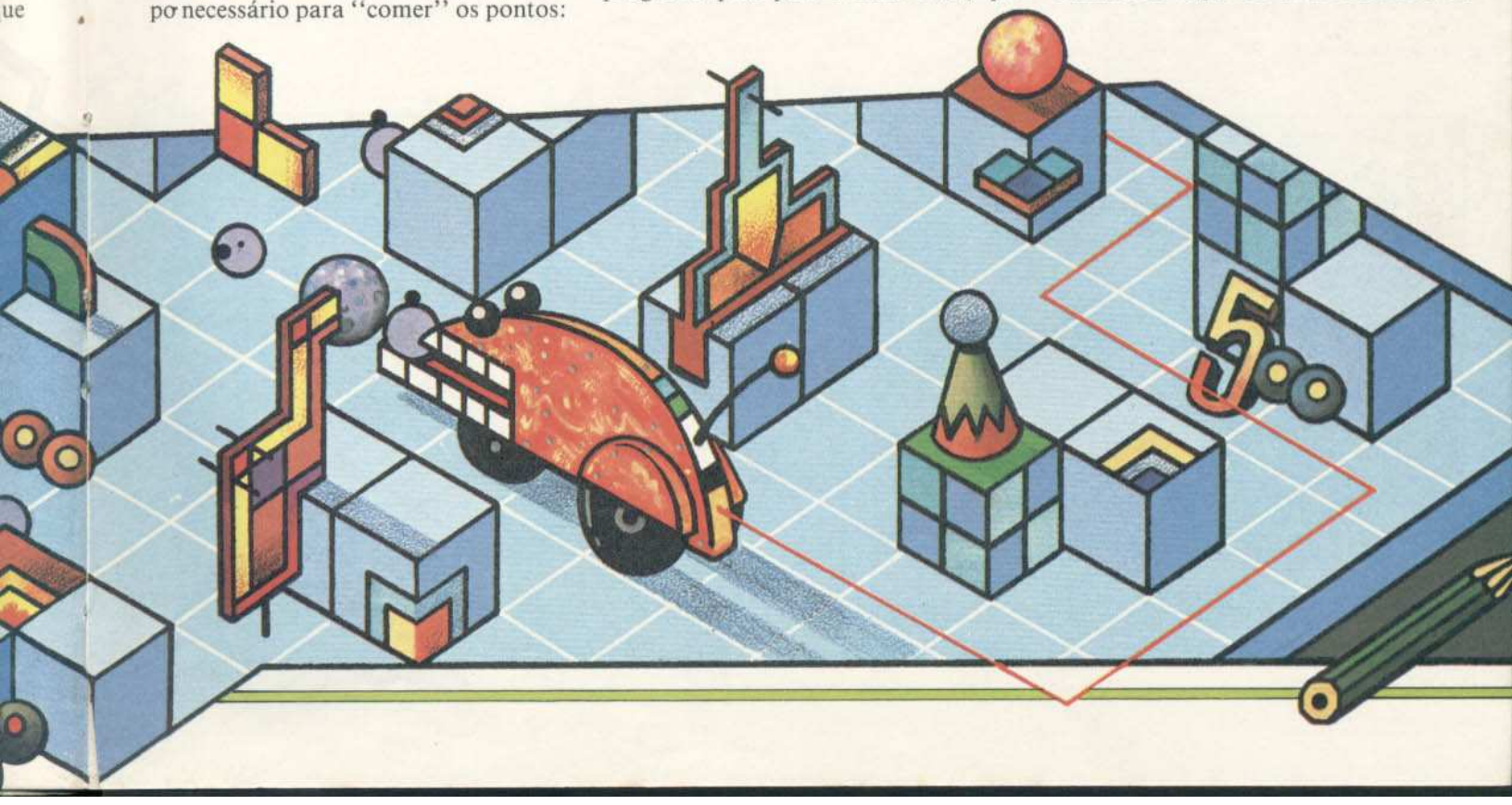
A linha 1290 aguarda, até que uma tecla seja pressionada, e a 1300 verifica se foi <**ENTER**>. Neste caso o jogo recomeça. O **RESTORE** na linha 50 permite que os dados (**DATA**) possam ser lidos novamente. Se <**ENTER**> não foi teclada, o computador continua esperando outra entrada.

A tecla <**BREAK**>, finalmente, termina o jogo.

CONSTRUA SEUS PROPRIOS LABIRINTOS

Use papel quadriculado para desenhar seus labirintos. Tenha sempre em mente o tamanho da tela do computador: 32 por 16 caracteres.

Desenhado o novo labirinto, altere o conteúdo das linhas **DATA**. Se o novo labirinto for maior ou menor que o antigo, você terá que modificar o tamanho do laço **FOR...NEXT** (linhas 1000 e 1030). Conte quantas linhas o novo labirinto ocupa e subtraia 1. Coloque o resultado no lugar do último número da



linha 1000. Da mesma forma, faça a contagem do número de posições da primeira linha e subtraia 1. Coloque, em seguida, esse número no lugar de 18 no fim da linha 1030.

Finalmente, conte quantos pontos existem no seu novo labirinto. Se encontrar um número diferente de 153, altere a linha 1160 também, substituindo 153 pelo novo número.

Estas primeiras linhas constituem o labirinto propriamente dito:

```
10 SCREEN 1:COLOR 1,3,3
40 LET P=247
70 VPOKE BASE(6)+5,26
80 VPOKE BASE(6)+1,102
90 LOCATE 0,3
100 FOR N=0 TO 15
110 READ A$
120 PRINT TAB(4);A$
130 FOR M=0 TO 18
140 IF VPEEK(BASE(5)+102+N*32+M
)=65 THEN VPOKE BASE(5)+102+N*3
2+M,8
150 NEXT M
160 NEXT N
1000 DATA AAAAAAAAAAAAAAAAAAA
1010 DATA A.......AAA.......A
1020 DATA A.AA.AA.....AA.AA.A
1030 DATA A.A.....AAA.....A.A
1040 DATA A...A.A.....A.A...A
1050 DATA A.AAA.AA.A.AA.AAA.A
1060 DATA A...A.........A...A
1070 DATA AAA...A.AAA.A...AAA
1080 DATA AAA...A.AAA.A...AAA
1090 DATA A...A.........A...A
1100 DATA A.AAA.AA.A.AA.AAA.A
1110 DATA A...A.A.....A.A...A
1120 DATA A.A.....AAA.....A.A
1130 DATA A.AA.AA.....AA.AA.A
1140 DATA A.......AAA.......A
1150 DATA AAAAAAAAAAAAAAAAAAA
```

As linhas de dados (**DATA**), de 1000 a 1150 encerram a forma do labirinto. Os limites deste são representados por letras "A".

Antes que os dados sejam lidos, a linha 10 limpa a tela, introduzindo-a no modo gráfico de 32 colunas (comando **SCREEN**), e fixa a cor de fundo e de frente (comando **COLOR**). A linha 40 define a posição inicial do "Come-come" (que será representado por um asterisco).

As linhas de 100 a 160 desenham o labirinto na tela. Sempre que o laço **FOR...NEXT** das linhas 100 e 160 é executado, o programa lê a próxima linha de dados (**DATA**). A linha 110 lê os dados, denominando-os de A$. A linha 120 coloca-os na tela.

Todos esses pontos e letras "A" combinados não formam, contudo, um labirinto de verdade. Assim, a linha 140 verifica a porção da memória da máquina que corresponde a cada posição de tela. Isso é feito com a função **VPOKE BASE (5)**, que torna acessível a memória dedicada de vídeo do MSX, na chamada página 5. Se o endereço de memória contém o número 65, isso quer dizer que temos um "A" na tela e ele é mudado para um bloco azul (código 8). As linhas 130 e 150 fornecem o laço que permite que a linha 140 verifique todas as posições da tela.

Agora adicione estas linhas e desloque um homenzinho pelo labirinto.

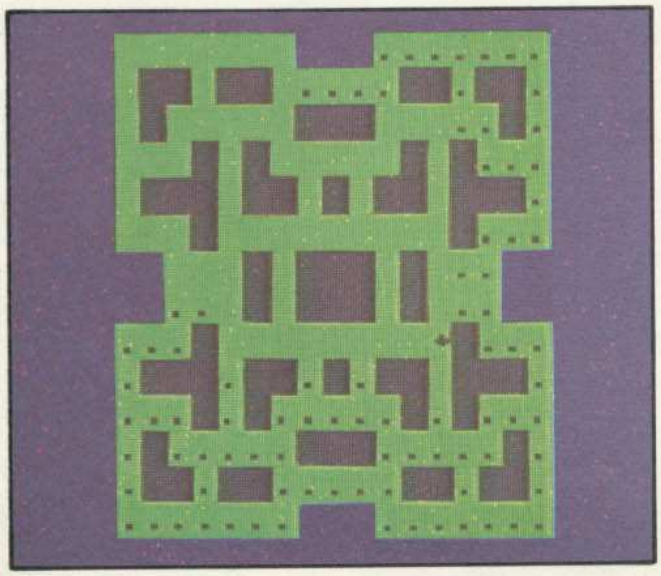
```
60 VPOKE BASE(6)+31,170
180 VPOKE BASE(5)+P,ASC("*")
190 LET LP=P
195 K$=INKEY$:IF K$="" THEN 195
200 IF K$=CHR$(29) AND VPEEK(BA
SE(5)+P-1)<>8 THEN LET P=P-1
210 IF K$=CHR$(28) AND VPEEK(BA
SE(5)+P+1)<>8 THEN LET P=P+1
220 IF K$=CHR$(30) AND VPEEK(BA
SE(5)+P-32)<>8 THEN LET P=P-32
230 IF K$=CHR$(31) AND VPEEK(BA
SE(5)+P+32)<>8 THEN LET P=P+32
250 VPOKE BASE(5)+LP,255
270 GOTO 180
```

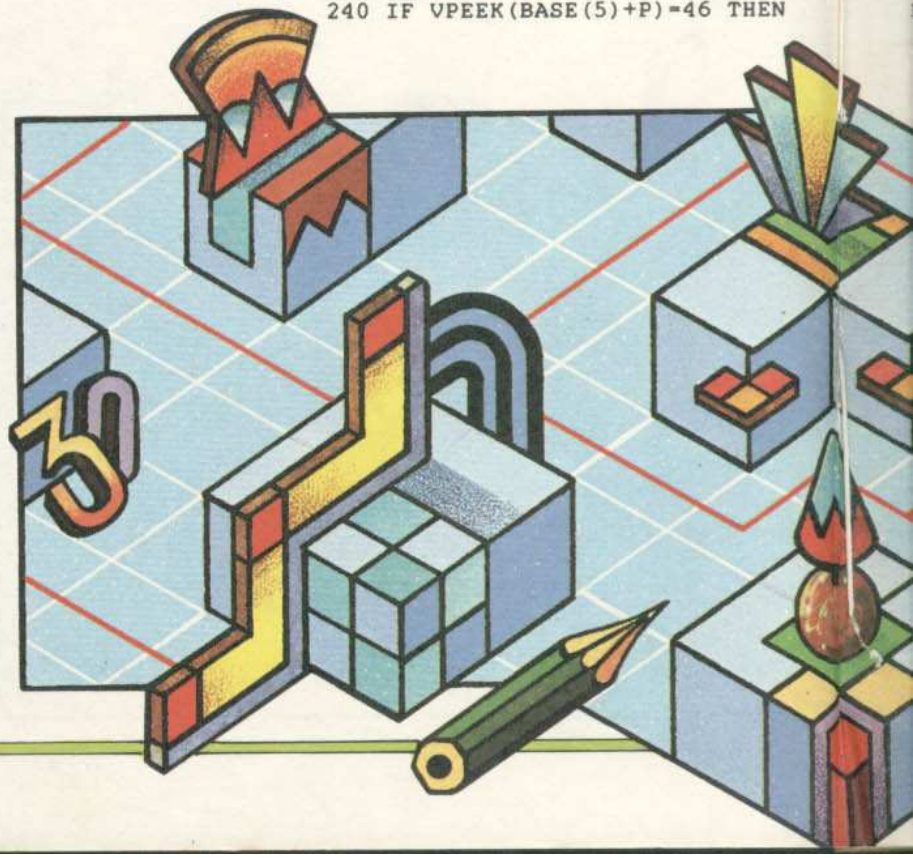
O movimento do caractere pelo labirinto só é possível se o ponto para onde se quer levá-lo não contém um bloco azul. A linha 200 verifica se uma tecla foi apertada e qual é o conteúdo de duas posições de memória: a que corresponde à tecla de movimentação do cursor "flecha para baixo" e a que corresponde à posição de tela em que o homem vai entrar. Se a tecla (código 29) está sendo pressionada, e o endereço de tela não contém um bloco azul (código 8), ele pode se movimentar.

As teclas para controle de movimento são as usadas para a movimentação do cursor (marcadas com as flechas nas quatro direções). Experimente deslocar o asterisco e "comer" (apagar) os pontinhos.

Adicione estas linhas e marque o tempo para "comer" os pontos:

```
20 FA=999999!
30 RESTORE
50 LET D=0
170 TIME=0
240 IF VPEEK(BASE(5)+P)=46 THEN
```





Um labirinto bem fácil para o MSX.

```
 SOUND 7,7:SOUND 6,0:SOUND 8,15
:SOUND 8,0:LET D=D+1
260 IF D=153 THEN GOTO 280
280 VPOKE BASE(5)+P,255
290 LET TI=INT(TIME/50)
300 LOCATE 7,1:PRINT "TEMPO=";T
I;"SEG  ";
310 IF FA>TI THEN FA=TI
320 LOCATE 3,20:PRINT "MELHOR T
EMPO=";FA;"SEG  ";
330 LOCATE 7,21:PRINT "Aperte R
ETURN ";
390 LET K$=INKEY$:IF K$="" THEN
 390
400 IF K$=CHR$(13) THEN GOTO 30
410 GOTO 390
```

O tempo gasto para remover os pontos, assim como o melhor tempo, é mostrado à direita do labirinto, quando você executa este programa. De início, o melhor tempo é colocado em 999999 segundos pela linha 20. A linha 50 zera o placar, e a linha 170 zera o "cronômetro".

A linha 240 examina a posição em que o asterisco está, para ver se há um ponto. Em caso positivo, o comando **SOUND** é usado para emitir um "*blip*" enquanto o ponto é apagado. Ela também acrescenta um ponto a D, que contém o número de pontos "comidos". A linha 260 verifica se não há mais pontos na tela. Se estes terminaram, o programa pula para a linha 280, que apaga o asterisco. Acionamos aqui o comando **VPOKE**, como acima.

A declaração **TI = INT(TIME/50)**, agora na linha 290, pára o relógio e divide o que está sendo lido por 50 para transformá-lo em segundos. A linha 300 mostra o tempo na telą.

A linha 310, por sua vez, checa se o melhor tempo (FA) é maior que o tempo obtido na última contagem. Se foi, ele é igualado a este último. A linha 320 mostra o melhor tempo, antes que uma mensagem diga ao jogador para recomeçar, teclando <**ENTER**>.

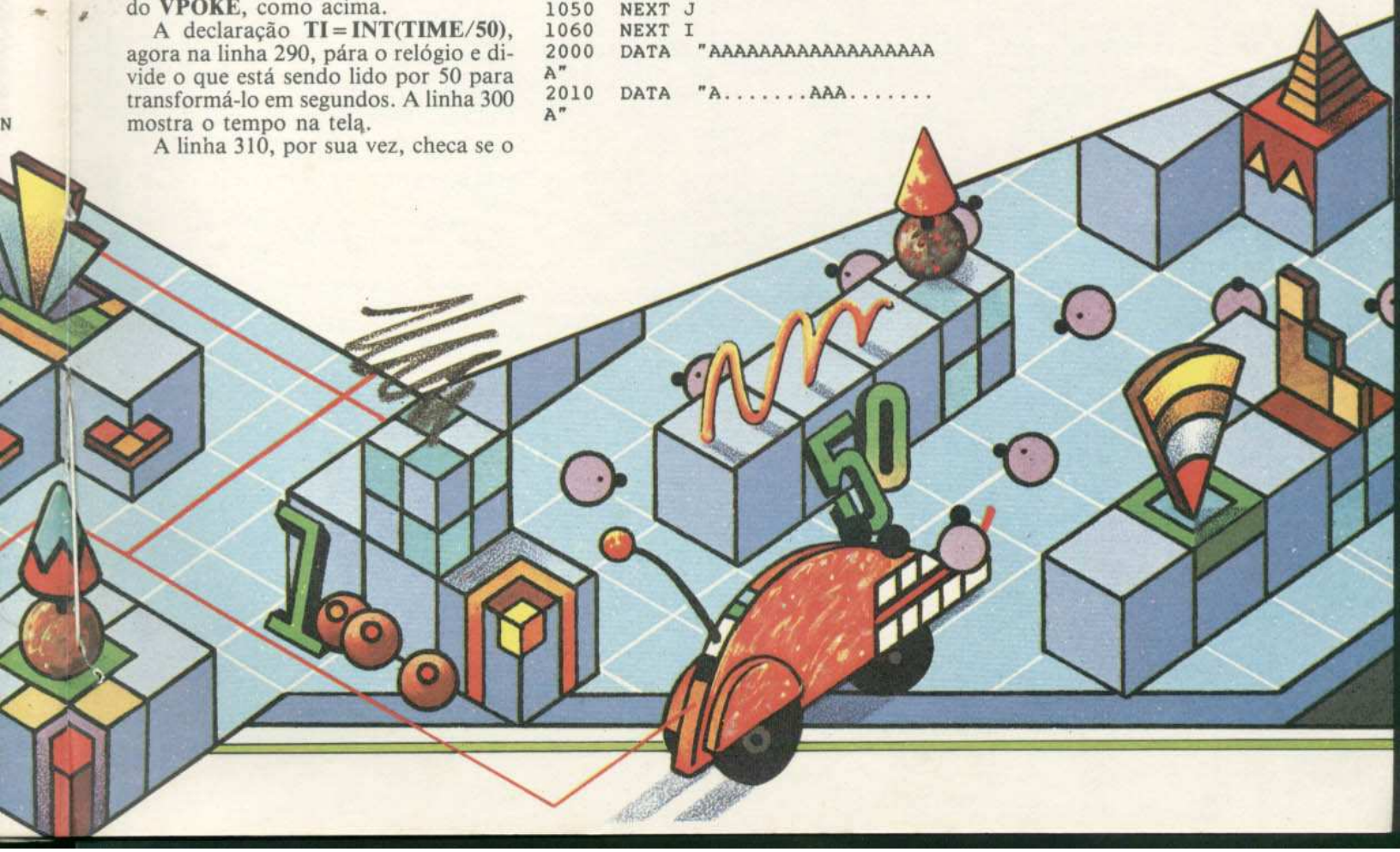
A linha 390 aguarda até que uma tecla tenha sido pressionada, e a linha 400 verifica se foi <**ENTER**>. Neste caso, o jogo recomeça. O **RESTORE** na linha 30 permite que os dados (**DATA**) possam ser lidos novamente. Se alguma tecla diferente de <**ENTER**> foi pressionada, o computador continua esperando outra entrada.

O comando <**CTRL**><**STOP**> encerra o jogo.

Estas primeiras linhas formam o labirinto propriamente dito:

```
30  HOME : GR
40  LET M = 18: LET N = 4
1000  FOR I = 0 TO 15
1010  READ A$
1020  FOR J = 1 TO 19
1025  IF  MID$ (A$,J,1) = "A" T
HEN  COLOR= 3: PLOT 2 * J,2 * I
: PLOT 2 * J + 1,2 * I: PLOT 2
* J,2 * I + 1: PLOT 2 * J + 1,2
 * I + 1
1030  IF  MID$ (A$,J,1) = "." T
HEN  COLOR= 12: PLOT 2 * J,2 *
I
1050  NEXT J
1060  NEXT I
2000  DATA  "AAAAAAAAAAAAAAAAAAA
A"
2010  DATA  "A.......AAA.......
A"
2020  DATA  "A.AA.AA.....AA.AA.
A"
2030  DATA  "A.A.....AAA.....A.
A"
2040  DATA  "A...A.A.....A.A...
A"
2050  DATA  "A.AAA.AA.A.AA.AAA.
A"
2060  DATA  "A...A.........A...
A"
2070  DATA  "AAA...A.AAA.A...AA
A"
2080  DATA  "AAA...A.AAA.A...AA
A"
2090  DATA  "A...A.........A...
A"
2100  DATA  "A.AAA.AA.A.AA.AAA.
A"
2110  DATA  "A...A.A.....A.A...
A"
2120  DATA  "A.A.....AAA.....A.
A"
2130  DATA  "A.AA.AA.....AA.AA.
A"
2140  DATA  "A.......AAA.......
A"
2150  DATA  "AAAAAAAAAAAAAAAAAAA
A"
```

Os limites do labirinto são representados, neste estágio, tal como em exemplos anteriores, por letras "A". Da mesma forma, seus contornos estão contidos nas linhas de dados (**DATA**), de 2000 a 2150. Portanto, o programa só



funcionará se forem adicionadas essas linhas. É relativamente fácil alterar a forma para um novo labirinto: basta modificar as linhas de 2000 a 2150.

Antes que os dados sejam lidos, a linha 30 limpa a tela e a coloca num modo gráfico de baixa resolução. A linha 40 coloca o asterisco na posição incial.

As linhas de 1000 a 1060 desenham o labirinto na tela. A cada vez que o laço **FOR...NEXT** das linhas 1000 e 1060 é executado, o programa lê a próxima linha de dados (**DATA**). A linha 1010 lê os dados, e os chama A$. A linha 1025 coloca-os na tela, na forma de quadradinhos coloridos (cor = 3 onde houver a letra A, e cor = 12 em outros pontos).

Agora adicione estas linhas e você poderá deslocar, mais uma vez, um homenzinho pelo labirinto.

```
1080  COLOR= 1: PLOT 2 * M,2 *
N: PLOT 2 * M + 1,2 * N: PLOT 2
 * M,2 * N + 1: PLOT 2 * M + 1,
2 * N + 1
1090  LET PM = M: LET PN = N
1100  IF  PEEK ( - 16384) = 218
 AND  SCRN( 2 * (M - 1),2 * N)
<  > 3 THEN  LET M = M - 1
1110  IF  PEEK ( - 16384) = 216
 AND  SCRN( 2 * (M + 1),2 * N)
<  > 3 THEN  LET M = M + 1
1120  IF  PEEK ( - 16384) = 208
 AND  SCRN( 2 * M,2 * (N - 1))
<  > 3 THEN  LET N = N - 1
1130  IF  PEEK ( - 16384) = 204
 AND  SCRN( 2 * M,2 * (N + 1))
<  > 3 THEN  LET N = N + 1
1150  COLOR= 0: PLOT 2 * PM,2 *
 PN: PLOT 2 * PM + 1,2 * PN: PL
OT 2 * PM,2 * PN + 1: PLOT 2 *
PM + 1,2 * PN + 1
1170  GOTO 1080
```

Esta parte do programa dá a você a possibilidade de movimentar o caractere pelo labirinto. Para que haja movimento, porém, o ponto para onde se quer levar o caractere (nosso homenzinho) não deve conter nenhum bloco de cor 12.

A linha 1100, por exemplo, verifica o conteúdo de duas posições de memória: a que corresponde à tecla "Z" e a que corresponde à posição de tela em que o homem vai entrar. Se a tecla "Z" (código 218) está sendo pressionada, e o endereço de tela não contém um bloco gráfico, então ele pode se movimentar.

As teclas para controle de movimento são as mesmas já utilizadas anteriormente: "Z" para a esquerda, "X" para a direita, "P" para cima e "L" para baixo.

Marque o tempo necessário para "comer" todos os pontos, acrescentando estas linhas:

```
20  LET FA = 99999
50  RESTORE
60  LET D = 0
1070  LET TI = 0
1140  IF  SCRN( 2 * M,2 * N) =
12 THEN  LET W =  PEEK ( - 1633
6): LET D = D + 1
1150  COLOR= 0: PLOT 2 * PM,2 *
 PN: PLOT 2 * PM + 1,2 * PN: PL
OT 2 * PM,2 * PN + 1: PLOT 2 *
PM + 1,2 * PN + 1
1160  IF D = 153 THEN  GOTO 118
0
1170  GOTO 1080
1180  COLOR= 0: PLOT 2 * M,2 *
N: PLOT 2 * M + 1,2 * N: PLOT 2
 * M,2 * N + 1: PLOT 2 * M + 1,
2 * N + 1
1200  HTAB 9: VTAB 21: PRINT "S
eu tempo foi de ";TI / 5;" seg"
1210  IF FA > TI THEN  LET FA =
 TI
1220  HTAB 11: VTAB 22: PRINT "
Melhor tempo: ";FA / 5;" seg"
1230  HTAB 6: VTAB 23: PRINT "P
ressione RETURN para continuar"
1240  GET T$: IF  ASC (T$) <  >
 13 THEN  GOTO 1210
1250  GOTO 40
```

De início, o melhor tempo é colocado em 999999 segundos pela linha 20. A linha 60 zera o placar e a linha 1070 zera o "cronômetro". A linha 1095 aumenta uma posição no cronômetro. A linha 1140 examina a posição em que o asterisco está. Ela também adiciona um ponto a D, que contém o número de pontos que foram "comidos".

A linha 1160 verifica se não há mais pontos na tela (em número de 153). Se estes terminaram, o programa pula para a linha 1180, que apaga o asterisco.

A linha 1200 mostra o tempo na tela (que é convertido para segundos dividindo-se FA por 50).

A linha 1210 checa se o melhor tempo (FA) é maior que o tempo obtido na última contagem. Em caso positivo, ele é igualado a este último. A linha 1220 mostra o melhor tempo, antes que uma mensagem diga ao jogador para teclar <**ENTER**> para recomeçar (na linha 1230).

A linha 1290 aguarda até que uma tecla tenha sido pressionada, e a linha 1300 verifica se foi <**ENTER**>. Neste caso, o jogo recomeça. O **RESTORE** na linha 50 permite que os dados (**DATA**) possam ser lidos novamente. Caso tenha sido acionada uma tecla que não <**ENTER**>, o computador continua à espera de outra entrada. A tecla <**CTRL-C**> termina o jogo.

MICRO DICAS

CONVERSÃO DO COLOR PARA O TRS-80

Os micros TRS-80 e TRS-Color têm interpretadores BASIC similares, no que diz respeito ao núcleo básico de comandos, funções e declarações. Por isso, é fácil converter programas de um modelo para outro. As maiores diferenças residem nos comandos de alta resolução gráfica, cor e som. Os comandos **CLS** (limpa telas), **SET** e **POINT** (pontos gráficos de resolução média) e **PRINT@** são semelhantes nas duas linhas. As diferenças de sintaxe do TRS-80 em relação ao Color são: o **CLS**, o **SET** e o **POINT** não têm parâmetro indicativo de cor de fundo (tela de texto), e as locações de tela usadas com o **PRINT@** produzem resultados diferentes.

A tela do Color tem dezesseis linhas por 32 colunas, e a do TRS-80, dezesseis linhas por 64 colunas. Como as posições de tela usadas pelo **PRINT@** são numeradas seqüencialmente, um **PRINT@ 162**, por exemplo, escreverá na segunda coluna da sexta linha do TRS-Color, e na 32.ª posição da terceira linha, no TRS-80.

Há uma forma simples de converter comandos **PRINT@** do Color para o TRS-80: se **N** for a locução de tela do Color, sua correspondente no TRS-80 será

```
INT(N/32)*64+N-32
```

para a segunda linha em diante (a que começa na posição 32 do Color). Se N < 32, não há necessidade da fórmula acima: a locução de tela do TRS-80 será o próprio N.

COMO DESCOMPLICAR SAVEs E LOADs

- APRENDA A CONECTAR O GRAVADOR
- COMANDO DE GRAVAÇÃO
- VERIFIQUE SUAS FITAS
- COMANDO DE LEITURA

Qualquer criança é capaz de fazer funcionar um gravador. Quando se trata de conectá-lo ao computador, porém, as coisas podem se complicar. Saiba como sair dessa, começando por ajustar os controles.

Alguns problemas podem surgir quando se quer gravar ou carregar programas com um gravador cassete. As causas desses problemas nem sempre são identificáveis. Mas é possível estabelecer algumas rotinas que diminuam as margens de erro.

SAIBA CONECTAR CORRETAMENTE

Alguns micros empregam um gravador especial do tipo digital, que tem uma ligação única e direta com a máquina; essa característica elimina qualquer problema na instalação. Mas a maioria dos computadores pode ser ligada a um gravador cassete, freqüentemente utilizando mais que um tipo de cabo. Normalmente, a ligação é feita por meio de um plugue do tipo DIN ligado a três plugues de pino (*jaques*), mas pode-se encontrar também um conector DID em cada uma das pontas, com ou sem jaques em paralelo. Se você comprar um cabo ou usar um próprio para áudio, assegure-se de que ele é adequado: caso não seja, aparecerão problemas como interferências em cabos não blindados.

Se você usa o gravador com freqüência, é aconselhável deixar o cabo permanentemente ligado dos dois lados.

AJUSTE OS CONTROLES

O passo inicial e mais importante na adequação com o computador é ajustar os controles de volume, tonalidade e outras características do gravador. Se você tem um gravador especial para computadores, siga as instruções que o acompanham.

Se seu gravador é do tipo comum, comece desligando todos os controles especiais, como filtros e sistemas de redução de ruído (*Dolby*), etc. Ajuste o controle de tonalidade para o máximo de agudos (a tonalidade é controlada ou por um dial numerado, ou por um interruptor de duas posições grave/agudo). Deixe a tonalidade assim sempre que você usá-lo com o computador. Escolha um volume médio (entre 50 e 60% do máximo) e tente carregar (com os comandos **LOAD**, **CLOAD**, etc., dependendo do computador) um programa já gravado, como por exemplo o da fita de demonstração que acompanha sua máquina. Se o programa não carrega, aumente o volume pouco a pouco e repita a operação, até conseguir um bom resultado.

Se você alcançar o volume máximo ainda sem sucesso, volte o seletor até um ponto levemente abaixo daquele em que você começou e repita toda a operação, diminuindo o volume aos poucos.

Caso o programa não carregue em nenhum nível de volume, verifique as conexões novamente. Tente outro cabo ou outra fita, ou peça emprestado um gravador adequado ao seu tipo de computador. Se, depois disso, ele continuar sem funcionar, procure o seu revendedor.

Quando tiver encontrado um volume que permita o carregamento, observe a posição do controle, estabeleça os limites superior e inferior com os quais você consegue carregar o programa e marque o ponto médio para uso posterior. Você pode usar o mesmo volume para gravação, mesmo porque muitos gravadores têm um controle automático desse volume.

Para testar se o volume é adequado para gravação, limpe a memória do computador (digite **NEW** e pressione <**ENTER**> a seguir) e digite este pequeno programa.

MTS

```
10 REM TESTE DE GRAVACAO
20 REM
30 REM
40 REM
50 REM FIM DO TESTE
```

MICRO DICAS

EXTERMINADOR DE PROBLEMAS

— Utilize um gravador cassete monoaural (isto é, que não seja estéreo), de marca confiável e, se possível, reserve-o para uso exclusivo com o computador. Ligue-o sempre à rede elétrica, ao invés de usar pilhas, para assegurar uma rotação uniforme do motor.

— Evite usar aparelhos estereofônicos, a não ser que todos os seus recursos especiais possam ser desligados.

— Use fitas magnéticas para áudio de boa qualidade ou especiais para dados. Uma fita de má qualidade não resistirá por muito tempo a várias regravações.

— Procure instruções específicas na tela; o gravador deve ficar com a tecla **PLAY** acionada até que a operação de carga esteja completada.

— Altere as características de tonalidade, volume e outros controles se a carga do programa não for bem-sucedida.

— Afaste o gravador da TV ou monitor se um programa que já carregou uma vez não funcionar. Tente também uma outra gravação do programa para ver se funciona. Em caso afirmativo, a primeira fita deve ter sido danificada.

— Se for necessário ajustar o volume com freqüência, de programa para programa, marque o volume adequado na etiqueta da fita.

— Guarde suas fitas em lugar seco, livre de poeira e longe de fontes eletromagnéticas e do calor.

Siga então a rotina para gravação no computador, acionando **SAVE** ou **CSAVE**. Ao carregá-lo (comando **LOAD** ou **CLOAD**) e listá-lo (**LIST**) você obterá o programa de volta.

O COMANDO DE GRAVAÇÃO

O comando de gravação de fita não é um comando padrão do BASIC, variando muito entre os computadores. A maneira correta de manipulá-lo é geralmente explicada no manual que acompanha o micro. Alguns passos iniciais, porém, são importantes.

Antes de gravar qualquer programa, coloque uma fita de boa qualidade no gravador, tendo o cuidado de rodá-la um pouco para que a guia de plástico transparente, existente no começo (*leader*), não fique em contato com o cabeçote.

A seguir, escolha um nome adequado para seu programa. Os dados são armazenados na forma de "arquivos" de um tipo ou outro, independentemente do método real de armazenamento. Mas, para gravações feitas em fita, nomes especiais são empregados com o comando **SAVE**, de modo a identificar na fita qual o programa gravado (isso é útil para achar automaticamente o programa, depois, através do comando **LOAD**).

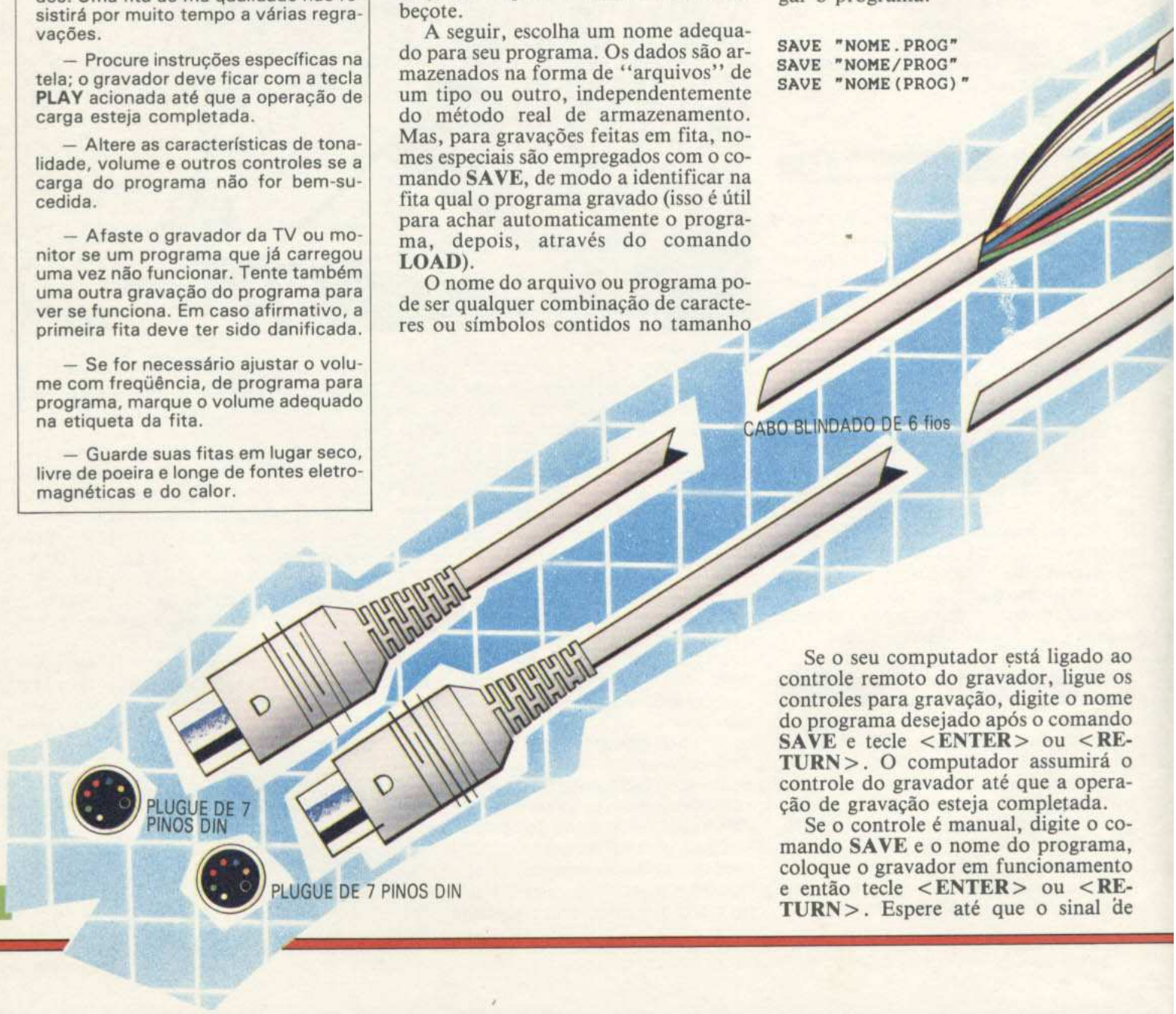
O nome do arquivo ou programa pode ser qualquer combinação de caracteres ou símbolos contidos no tamanho máximo permitido para uma linha no seu computador. Na maioria das máquinas, esse tamanho é de dez caracteres no máximo. Outros computadores, como o TRS-80, permitem apenas uma letra ou símbolo de identificação do programa: quando o nome é maior, os comandos **CSAVE** e **CLOAD** reduzem-no à primeira letra.

```
SAVE "NOMEPROG01"
SAVE "NomeProg01"
CSAVE "R"
```

Dos computadores considerados aqui, apenas os compatíveis com Apple II não exigem aspas.

Os exemplos de comandos **SAVE** acima são válidos, mas um deles usa letras maiúsculas e minúsculas. Se você misturá-las num nome de programa, deve fazer o mesmo quando quiser carregar o programa.

```
SAVE "NOME.PROG"
SAVE "NOME/PROG"
SAVE "NOME(PROG)"
```

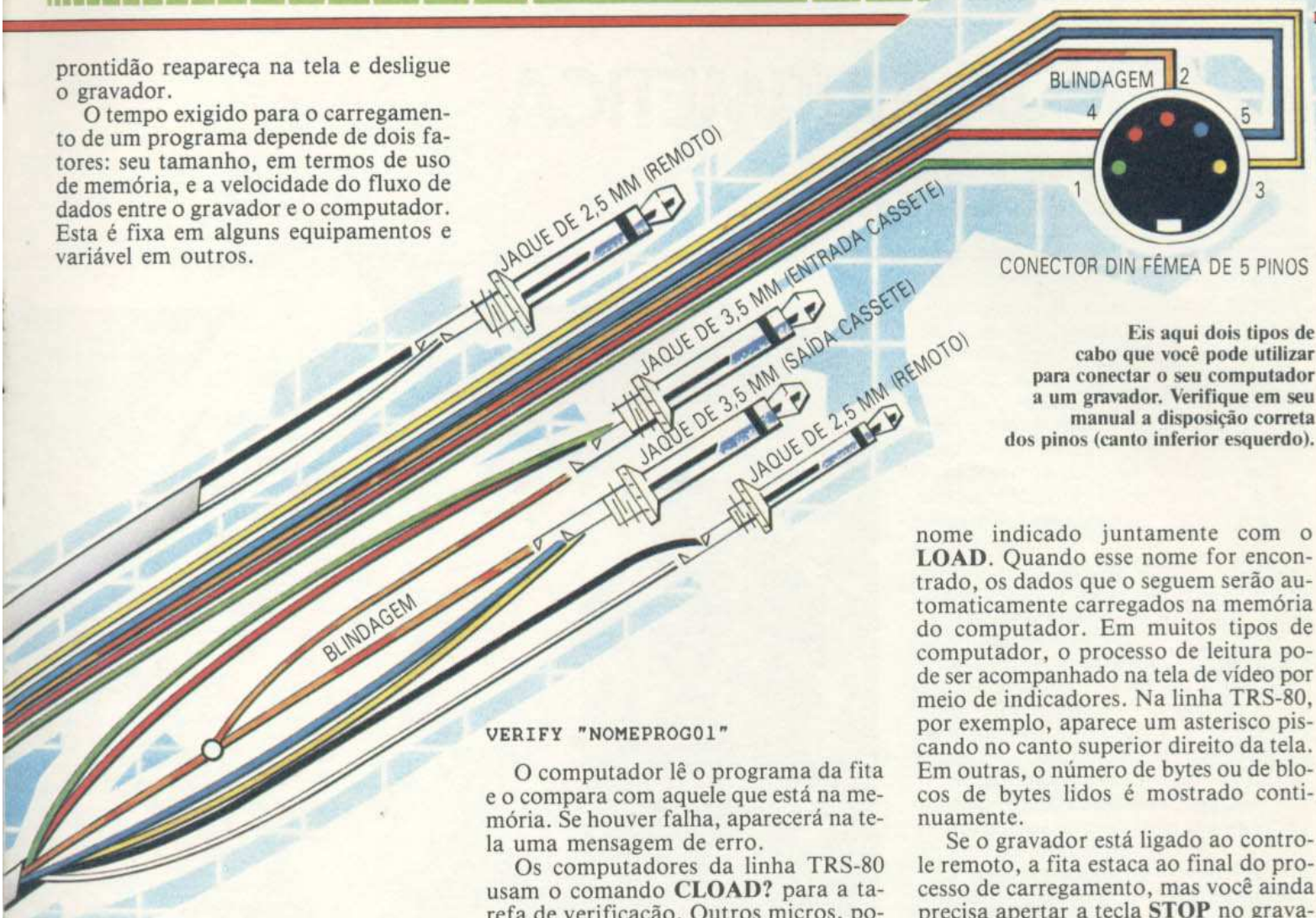


Se o seu computador está ligado ao controle remoto do gravador, ligue os controles para gravação, digite o nome do programa desejado após o comando **SAVE** e tecle <**ENTER**> ou <**RETURN**>. O computador assumirá o controle do gravador até que a operação de gravação esteja completada.

Se o controle é manual, digite o comando **SAVE** e o nome do programa, coloque o gravador em funcionamento e então tecle <**ENTER**> ou <**RETURN**>. Espere até que o sinal de

prontidão reapareça na tela e desligue o gravador.

O tempo exigido para o carregamento de um programa depende de dois fatores: seu tamanho, em termos de uso de memória, e a velocidade do fluxo de dados entre o gravador e o computador. Esta é fixa em alguns equipamentos e variável em outros.



Eis aqui dois tipos de cabo que você pode utilizar para conectar o seu computador a um gravador. Verifique em seu manual a disposição correta dos pinos (canto inferior esquerdo).

As gravações mais confiáveis são feitas na velocidade mais baixa. Altas velocidades usam menos fita e, obviamente, operam mais rápido. Nesse caso é importante usar fitas de alta qualidade.

VERIFIQUE AS GRAVAÇÕES

Uma maneira de verificar se um programa foi gravado adequadamente é carregá-lo e executá-lo. Quando a gravação é imperfeita o programa que está na memória (e que você quer gravar) é substituído por aquele que está na fita, e a gravação se perde. Se esta tiver sido bem-sucedida, não haverá problemas.

Muitos computadores contornam essa dificuldade fornecendo um comando verificador (por exemplo, **VERIFY**). O seu uso exige o rebobinamento da fita até o início do programa. Então executa-se o comando usando o mesmo nome que se deu ao programa ao gravá-lo. Por exemplo:

```
VERIFY "NOMEPROG01"
```

O computador lê o programa da fita e o compara com aquele que está na memória. Se houver falha, aparecerá na tela uma mensagem de erro.

Os computadores da linha TRS-80 usam o comando **CLOAD?** para a tarefa de verificação. Outros micros, porém (como os compatíveis com o Sinclair ZX-81), não têm qualquer função de verificação. Recorra, nesse caso, a programas de revistas especializadas ou vendidos à parte.

O COMANDO DE LEITURA

Se você começar com programas prégravados, seus primeiros problemas serão com o comando de leitura (**LOAD**). Podem existir várias formas do comando **LOAD**, mesmo para uma única máquina. Eis uma delas:

```
LOAD "NOMEPROG01"
```

Outros computadores utilizam a forma **CLOAD** (linha TRS-80), **LOADA** ou **LOADT** (TK-2000), etc.

O comando **LOAD** deve reproduzir o nome utilizado para gravação (**SAVE**) e verificação (**VERIFY**). Para que o computador leia os sinais da fita magnética, basta digitar **LOAD** e pressionar a tecla **PLAY** do gravador. Ele procurará pelo nome do programa e mostrará todos que encontrar pelo caminho, até encontrar aquele que confere com o nome indicado juntamente com o **LOAD**. Quando esse nome for encontrado, os dados que o seguem serão automaticamente carregados na memória do computador. Em muitos tipos de computador, o processo de leitura pode ser acompanhado na tela de vídeo por meio de indicadores. Na linha TRS-80, por exemplo, aparece um asterisco piscando no canto superior direito da tela. Em outras, o número de bytes ou de blocos de bytes lidos é mostrado continuamente.

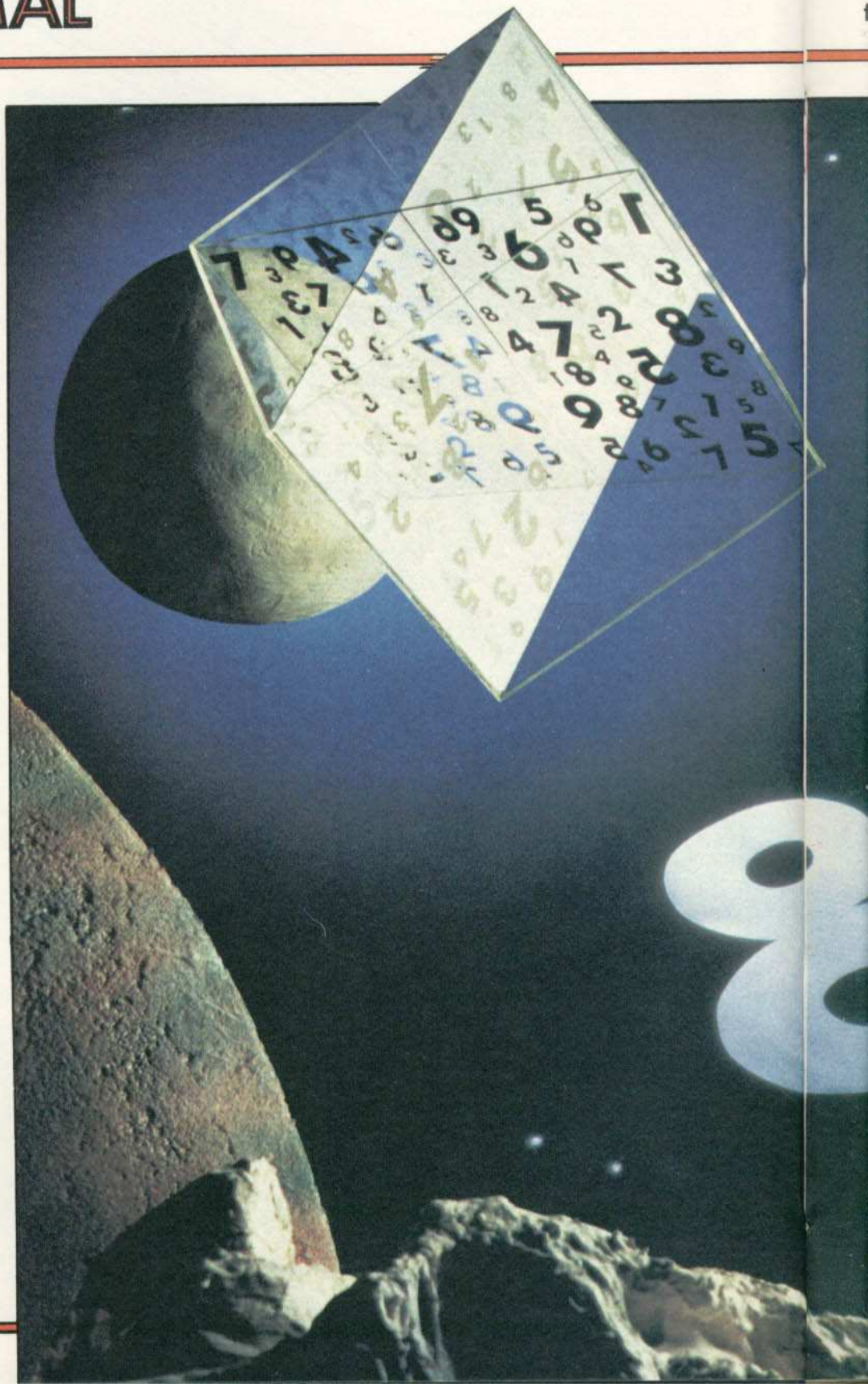
Se o gravador está ligado ao controle remoto, a fita estaca ao final do processo de carregamento, mas você ainda precisa apertar a tecla **STOP** no gravador. Se o controle é manual, simplesmente aperte **STOP** quando o cursor reaparecer na tela. Evite que a tecla **PLAY** fique acionada desnecessariamente, porque isso pode provocar o desgaste prematuro ou deformação da polia tratora.

Quando você usar o comando **LOAD** ou equivalente, o programa que for carregado tomará o lugar de qualquer outro que esteja na RAM (Memória de Acesso Randômico) do computador. Por isso, assegure-se de ter gravado o programa resistente na memória antes de carregar o próximo.

Rebobine a fita até o início do programa-teste que foi apresentado lá em cima, e tente carregá-lo (**LOAD**) usando o nome que você escolheu. Em caso de dificuldade, consulte o guia de resolução de problemas, neste artigo.

Outras formas do comando **LOAD** são necessárias para tornar acessíveis dados em código de máquina (por exemplo, **BLOAD**), para realocar programas e para juntar um programa a outro (**MERGE**). Seu primeiro contato com elas pode ter sido em instruções de jogos comprados ao acaso.

APRENDA ARITIMÉTICA HEXADECIMAL



Os circuitos internos dos computadores utilizam, como vimos, apenas números binários para realizar operações aritméticas. Ora, esse sistema de numeração, composto dos dígitos 0 e 1, cria uma série de dificuldades de compreensão e manipulação para os operadores.

Em primeiro lugar, os números binários de oito e dezesseis bits (os mais usados em microcomputadores) tornam-se confusos e difíceis de reconhecer, com tantos 0s e 1s. Digitá-los no computador é uma empresa difícil, sujeita freqüentemente a erros e incorreções.

A maneira de contornar esses problemas consiste em fazer com que o operador ou programador em nível de máquina passe a utilizar um sistema numérico com outra base, mas que ainda seja próximo, em concepção, ao sistema binário utilizado pelo computador.

O sistema universalmente adotado para isso é o *hexadecimal* (ou hexa), no qual os números utilizam a base 16. E por que o hexa funciona tão bem com computadores?

Antes de mais nada, ele está suficientemente próximo do decimal para ser tolerado por nós, simples mortais. Mas, além disso, 16 é uma potência de 2 (como 8 também o é, em outro sistema muito usado com computadores, o *octal*). Isso significa que a conversão entre os sistemas — binários e hexa —, nos dois sentidos, é bem mais simples do que entre o primeiro e o sistema decimal.

A única dificuldade é que um sistema com base maior do que 10 precisa criar novos algarismos, além dos que existem entre 0 e 9. No sistema hexadecimal utilizam-se letras adicionais (sempre em maiúsculas). O 10 decimal é representado por A, o 11 por B, o 12 por C, o 13 por D, o 14 por E e o 15 por F.

Tudo se passa como se tivéssemos oito dedos em cada mão. Essa imagem pode parecer estranha ou mesmo monstruosa, pois dificilmente suportamos alterações em nossa aparência física. Na prática, porém, é isso que acontece.

COMO CONVERTER BINÁRIO EM HEXA

A conversão de números binários de oito bits (os mais utilizados em compu-

Se todos tivéssemos dezesseis dedos, o sistema numérico mais comum seria certamente o hexadecimal. Embora isso não aconteça, esse sistema é fundamental para o código de máquina.

- POR QUE O SISTEMA HEXADECIMAL É UTILIZADO
- CONTANDO DE 16 EM 16
- A RELAÇÃO ENTRE OS SISTEMAS BINÁRIO E HEXADECIMAL
- CONVERSÕES FÁCEIS DE BINÁRIO PARA HEXA
- CONVERSÃO DE DECIMAL PARA HEXA

tadores pessoais) para o sistema hexadecimal é bastante fácil. Primeiro, você divide o número binário em duas partes de quatro bits cada. Em seguida, cada uma dessas partes é transformada em um dígito hexadecimal. O número hexa final é composto por esses dois dígitos. Da mesma forma, converte-se um número binário de dezesseis bits hexa dividindo-o em quatro segmentos de quatro bits cada.

Já a conversão de decimal para hexa é mais complicada, embora não seja tão difícil que você não possa realizá-la. Para se fazer isso, você precisa dividir o número decimal por 16, sucessivamente. Os restos dessa divisão formarão os dígitos do número em hexa.

Quando se divide o decimal 1226 por 16, o resto será 10. O 10 decimal é A em hexa. O resultado da divisão (76) é dividido, por sua vez, por 16, e dá 4, com resto de 12 (C em hexa). Finalmente, 4 dividido por 16 dá zero, com 4 de resto. Conclusão: 1226 em decimal é 4CA em hexa (note a seqüência inversa à divisão para os dígitos hexa). Para aprender melhor como funcionam todas essas conversões, digite o programa abaixo.

```
20 CLS 0
30 PRINT @11,"BIN,DEC,HEX";
40 PRINT @68,"BINARIO";
50 PRINT @196,"DECIMAL";
60 PRINT @323,"HEXADECIMAL";
70 PRINT @355,"+  +  +   = ";
80 PRINT @371,"+  +  +   = ";
90 FOR J=1 TO 15:POKE 1040+32*J
,175:NEXT
100 PRINT @450,"NUMERO HEX=
 ";
110 PRINT @227,"+   +   +    +
  +   +   + ";
120 GOTO 170
130 IN$=INKEY$:IF IN$="" THEN 1
30
140 IF IN$=" " THEN NO=NO+1:NO=
NO AND 255:GOTO 170
150 IF IN$="B" THEN NO=NO-1:NO=
NO AND 255:GOTO 170
160 GOSUB 370
170 GOSUB 190: GOSUB 270
180 GOTO 130
190 FOR X=7 TO 0 STEP-1
200 IF(NO AND 2^X)THEN N=1 ELSE
 N=0
210 PRINT @125-X*4,N;
220 IF N=1 THEN N=INT(2^X):N$=S
TR$(N):N$=MID$(N$,2,LEN(N$)-1)E
LSE N$=RIGHT$("  0",LEN(STR$(2^
X))-1)
230 PRINT @255-X*4-LEN(N$),N$;
240 NEXT
250 PRINT @279," = ";MID$(STR$(
NO)+"  ",2,3);
260 RETURN
270 FOR X=7 TO 4 STEP -1
280 PRINT @374-X*3,STR$((NO AND
 2^X)/16);
290 NEXT
300 PRINT @367,HEX$(NO/16);
310 FOR X=3 TO 0 STEP -1
320 PRINT @378-X*3,STR$(NO AND
2^X);
330 NEXT
340 PRINT @383,HEX$(NO AND 15);
350 POKE 1488,PEEK(1391):POKE 1
489,PEEK(1407)
360 RETURN
370 NU$="":PRINT @439,"?";
380 IN$=INKEY$:IF(IN$<"0" OR IN
$>"9") AND IN$<>CHR$(13) THEN G
OTO 380
390 IF IN$=CHR$(13) THEN NO=VAL
(NU$):IF NO>255 THEN 370 ELSE P
RINT @439,STRING$(5,CHR$(128));
:RETURN
400 IF IN$<>CHR$(13) AND LEN (N
U$)>2 THEN 380
410 NU$=NU$+IN$:PRINT @441,MID$
(NU$+"   ",1,3);:GOTO 380
```

```
 20 CLS
 25 PLOT 140,0: DRAW 0,160
 30 PRINT  INVERSE 1;AT 0,8;"
 BIN,DEC,HEX "
 40 PRINT  INVERSE 1;AT 4,2;"
 BINARIO:     "
 50 PRINT  INVERSE 1;AT 9,2;"
 DECIMAL:     "
 60 PRINT AT 10,5;"+   +   +
  +   +   +   +"
 70 PRINT  INVERSE 1;AT 17,2;"
 HEXADECIMAL:"
 80 PRINT AT 18,4;"+  +  +  ="
 90 PRINT AT 18,20;"+  +  +  =
"
100 LET no=0
110 GOTO 150
120 LET  a$=INKEY$ :  IF a$=""
 THEN  GOTO 120
130 IF a$=" " THEN  LET no=no+
1 : IF  no=256 THEN  LET no=0
135 IF a$="b" THEN  LET no=no-
```

```
1 : IF  no=-1 THEN  LET no=255
140 IF a$="b" OR a$=" " THEN
GOTO 150
145 INPUT "?";no
150 GOSUB 170: GOSUB 250
160 GOTO 120
170 LET nu=no: LET c=128
175 FOR x=0 TO 7
180 LET n=0: IF nu>=c THEN
LET n=1 : LET nu=nu-c
190 LET c=c/2
200 PRINT AT 5,2+4*x;n
210 IF n=1 THEN  PRINT AT 10,2
+4*x;c* 2
220 IF n=0 THEN  PRINT AT 10,2
+4*x;"0  "
230 NEXT x
235 PRINT AT 13,6;"TOTAL DECIM
AL= ";n o;"  "
240 RETURN
250 LET hi=INT (no/16): LET hh
=hi
260 LET lo=(no-hi*16): LET ll=
lo: IF  lo>9 THEN  LET lo=lo+7
265 IF hi>9 THEN  LET hi=hi+7
270 LET hi=hi+48: LET lo=lo+48
280 PRINT AT 18,14;CHR$ hi;AT
18,30;  CHR$ lo
290 LET c=8
300 FOR x=0 TO 3
310 LET n=0: IF hh>=c THEN
LET n=c : LET hh=hh-c
```

MICRO DICAS

Nos micros TRS-Color, TRS-80 (com BASIC em disquete) e MSX, existem funções específicas para converter decimais em hexas. Essas funções são (*d* corresponde a um número ou expressão em decimal):

TT MSX

HEX$ (d)

Se você quiser incluir números em hexadecimal como parte de um programa (por exemplo, dentro de declarações **DATA**), também não há problema para muitos computadores. Ao utilizar essas constantes em hexa para cálculos subseqüentes, o computador os converterá internamente em decimal (ou, mais propriamente, em binário).

TT MSX

Coloque &H antes do número. Exemplo: &HF3

Coloque $ antes do número. Exemplo: $F6.

```
315 LET m=0: IF 11>=c THEN
LET m=c : LET 11=11-c
320 LET c=c/2
330 PRINT AT 18,2+x*3;n;AT 18,
18+x*3; m
340 NEXT x
400 PRINT AT 21,6;"TOTAL HEX=
";CHR$  hi;CHR$ lo
500 RETURN
```

S

```
  20 CLS
  30 PRINT AT 0,9;"BIN,DEC,HEX"
  40 PRINT AT 4,4;"BINARIO:"
  50 PRINT AT 9,4;"DECIMAL:"
  60 PRINT AT 10,5;"+   +   +
+   +   +   "
  70 PRINT AT 17,4;"HEXADECIMAL:
"
  80 PRINT AT 18,4;"+  +  +  ="
  90 PRINT AT 18,20;"+  +  +  ="
 100 LET NO=0
 110 GOTO 150
 120 LET A$=INKEY$
 125 IF A$="" THEN GOTO 120
 130 IF A$<>"F" THEN GOTO 135
 131 LET NO=NO+1
 132 IF NO=256 THEN LET NO=0
 135 IF A$<>"B" THEN GOTO 140
 136 LET NO=NO-1
 137 IF NO=-1 THEN LET NO=255
 140 IF A$="B" OR A$="F" THEN GO
TO 150
 145 INPUT NO
 150 GOSUB 170
 155 GOSUB 250
 160 GOTO 120
 170 LET NU=NO
 171 LET C=128
 175 FOR X=0 TO 7
 180 LET N=0
 185 IF NU>=C THEN LET N=1
 186 IF NU>=C THEN LET NU=NU-C
 190 LET C=C/2
 200 PRINT AT 5,2+4*X;N
 210 IF N=1 THEN PRINT AT 10,2+4
*X;C*2
 220 IF N=0 THEN PRINT AT 10,2+4
*X;"0  "
 230 NEXT X
 235 PRINT AT 13,6;"TOTAL DECIMA
L= ";NO;"  "
 240 RETURN
 250 LET HI=INT (NO/16)
 255 LET HH=HI
 260 LET LO=(NO-HI*16)
 261 LET LL=LO
 270 LET HI=HI+28
 275 LET LO=LO+28
 280 PRINT AT 18,14;CHR$ HI;AT 1
8,30;CHR$ LO
 290 LET C=8
 300 FOR X=0 TO 3
 310 LET N=0
 311 IF HH>=C THEN LET N=C
 312 IF HH>=C THEN LET HH=HH-C
 315 LET M=0
 316 IF LL>=C THEN LET M=C
 317 IF LL>=C THEN LET LL=LL-C
 320 LET C=C/2
 330 PRINT AT 18,2+X*3;N;AT 18,1
8+X*3;M
 340 NEXT X
 400 PRINT AT 21,6;"TOTAL HEX= "
;CHR$ HI;CHR$ LO
 500 RETURN
```

MSX

```
5 KEY OFF
10 SCREEN0
15 COLOR 1,9
20 CLS
30 LOCATE 10,0:PRINT" BIN,DEC,H
EX"
40 LOCATE 2,4:PRINT"  BINARIO:
 "
50 LOCATE 2,9:PRINT"  DECIMAL:
 "
60 LOCATE 3,10:PRINT"+   +   +
  +   +   +   +"
70 LOCATE 2,17:PRINT"  HEXADECI
MAL:"
80 LOCATE 4,18:PRINT "+  +  +
="
90 LOCATE 20,18:PRINT"+  +  +
="
100 LET NO=0
110 GOTO 150
120 LET A$=INKEY$:IF A$="" THEN
 GOTO 120
130 IF A$=" " THEN LET NO=NO+1:
IF NO=256 THEN LET NO=1
135 IF A$="B" THEN LET NO=NO-1:
IF NO=-1 THEN LET NO=255
140 IF A$="B" OR A$=" " THEN GO
TO 150
145 LOCATE 30,0:INPUT NO
150 GOSUB 170:GOSUB 250
160 GOTO 120
170 LET NU=NO:LET C=128
175 FOR X=0 TO 7
180 LET N=0:IF NU>=C THEN LET N
=1:LET NU=NU-C
190 LET C=C/2
200 LOCATE 2+4*X,5:PRINT USING
"#";N;
210 IF N=1 THEN LOCATE 4*X,10:P
RINT USING "###";C*2
220 IF N=0 THEN LOCATE 4*X,10:P
RINT USING "###";0
230 NEXT X
235 LOCATE 6,13:PRINT "TOTAL DE
CIMAL=";NO;"  "
240 RETURN
250 LET HI=INT(NO/16):LET HH=HI
260 LET LO=(NO-HI*16):LET LL=LO
:IF LO>9 THEN LET LO=LO+7
265 IF HI>9 THEN HI=HI+7
270 LET HI=HI+48:LET LO=LO+48
280 LOCATE 14,18:PRINT CHR$(HI)
;
285 LOCATE 30,18:PRINT CHR$(LO)
;
290 LET C=8
300 FOR X=0 TO 3
310 LET N=0:IF HH>=C THEN LET N
=C:LET HH=HH-C
315 LET M=0:IF LL>=C THEN LET M
=C:LET LL=LL-C
320 LET C=C/2
330 LOCATE 2+X*3,18:PRINT USING
"c";N?
```

```
335 LOCATE 18+X*3,18:PRINT USIN
G"#";M;
340 NEXT LOCATE
400 LOCATE 6,21:PRINT"TOTAL HEX
= ";CHR$(HI);CHR$(LO);
500 RETURN
```

```
20  HOME
25  POKE 34,22
30  INVERSE : HTAB 14: PRINT "B
IN,DEC,HEX"
40  VTAB 6: HTAB 5: PRINT "BINA
RIO"
50  VTAB 10: HTAB 5: PRINT "DEC
IMAL"
60  VTAB 16: HTAB 5: PRINT "HEX
ADECIMAL"
70  VTAB 22: HTAB 3: PRINT "NUM
ERO HEX=      "
80  VTAB 17: PRINT "   +   +
+   =   ";: HTAB 21: PRINT "
+   +   +   =   "
90  VTAB 11: PRINT "    +    +
   +    +    +    +    +    "
110  GOTO 170
120  VTAB 23: CALL  - 958: HTAB
 30: GET IN$
130  IF IN$ =  CHR$ (32) THEN N
O = NO + 1: IF NO = 256 THEN NO
 = 0
140  IF IN$ =  CHR$ (66) THEN N
O = NO - 1: IF NO =  - 1 THEN N
O = 255
150  IF IN$ <  >  CHR$ (32) AND
 IN$ <  >  CHR$ (66) THEN  GOSU
B 410
170  GOSUB 190: GOSUB 270
180  GOTO 120
190 NU = NO:C = 128: FOR X = 7
TO 0 STEP  - 1
200 N = 0: IF NU >  = C THEN N
= 1:NU = NU - C
205 C = C / 2
210  VTAB 7: HTAB  ABS (X - 7)
* 5 + 1: PRINT "  ";N;" "
220  IF N = 1 THEN N = 2 ^ X:N$
 =  STR$ (N): GOTO 230
225 N$ = "  0"
230  VTAB 11: HTAB  ABS (X - 7)
 * 5 + 1 + (3 -  LEN (N$)): PRI
NT N$
240  NEXT
250  VTAB 13: HTAB 5: PRINT "TO
TAL DECIMAL= ";NO;" ";: CALL  -
 868
260  RETURN
270 HI =  INT (NO / 16):HH = HI
280 LO = (NO - HI * 16):LL = LO
: IF LO > 9 THEN LO = LO + 7
290  IF HI > 9 THEN HI = HI + 7
300 HI = HI + 48:LO = LO + 48
310  VTAB 17: HTAB 18: PRINT  C
HR$ (HI);: HTAB 38: PRINT  CHR$
 (LO)
320 C = 8
330  FOR X = 0 TO 3
340 N = 0: IF HH >  = C THEN N
= C:HH = HH - C
350 M = 0: IF LL >  = C THEN M
= C:LL = LL - C
```

```
360 C = C / 2
370  VTAB 17: HTAB X * 4 + 2: P
RINT N;: HTAB X * 4 + 22: PRINT
 M
380  NEXT
390  VTAB 22: HTAB 15: PRINT  C
HR$ (HI); CHR$ (LO)
400  RETURN
410  VTAB 23: HTAB 30: INPUT NO
420  IF NO > 255 OR NO < 0 THEN
 410
430  NORMAL : VTAB 23: HTAB 30:
 PRINT "       ": INVERSE : RETU
RN
```

Uma vez digitado o programa, rode-o com o comando **RUN**. Aparecerão na tela três linhas: uma contendo os dígitos do número em binário; mais abaixo, a linha do número em decimal; e, por último, os dois segmentos do número em hexadecimal.

Inicialmente, todos esses números estão zerados. Ao pressionar a tecla de espaçamento (a tecla **F** na versão para o ZX-81), o número binário no topo da tela será incrementado de 1 e seus equivalentes em decimal e hexadecimal aparecerão na parte de baixo da tela. O equivalente decimal é calculado somando-se os valores (potências de 2) correspondentes às posições onde um dígito binário é igual a 1. Exemplo:

BINÁRIO:
0 0 1 0 0 1 1 0

DECIMAL:
$0 + 0 + 32 + 0 + 0 + 4 + 2 + 0 = 38$

POTÊNCIAS DE 2:
$2^7 2^6 2^5 2^4 2^3 2^2 2^1 2^0$

O mesmo método de conversão é utilizado para o sistema hexadecimal; apenas, neste caso, formam-se quatro grupos de quatro bits de cada vez.

Pressionando a tecla <**B**>, você diminuirá 1 do número registrado na tela. Ao mesmo tempo, se você pressionar qualquer outra tecla, além da barra de espaços ou da letra B, poderá ordenar rapidamente ao computador que converta para binário e hexadecimal um número qualquer em decimal entre 0 e 255. Ao pressionar qualquer tecla, aparecerá em um canto da tela um sinal de interrogação. Digite então o número desejado e acione <**ENTER**> ou <**RETURN**>. Ato contínuo, os números equivalentes em binário e hexa serão mostrados no vídeo.

Note que o número máximo que pode ser representado por um byte de oito bits em binário é 11111111. Isso equivale a 255 em decimal e a FF em hexadecimal. Qualquer número armazenado em um byte na memória do seu computador pode ser representado por um valor em hexa com dois dígitos.

Com o tempo, você verificará que essas conversões não são tão difíceis quanto parecem a princípio.

O sistema hexadecimal parece difícil e complicado. Não seria possível passar sem ele? Afinal, é realmente necessário aprendê-lo para operar em linguagem de máquina?

Sim. Se você tem a intenção de aprender seriamente a programar em código de máquina, um bom conhecimento sobre como trabalhar com o sistema hexadecimal é imprescindível.

Existem programas prontos para fazer a conversão automática de códigos de operação em números binários (Assembler) ou a conversão no sentido inverso (Disassembler). Além disso, eles aceitam números expressos no sistema decimal. No entanto, mesmo nesses casos, o programador precisa recorrer constantemente aos seus equivalentes hexadecimais. Muitas vezes, certas listagens emitidas por tais programas (DUMP) contêm apenas números hexadecimais. E você não saberia trabalhar com eles se não conhecesse o sistema de base 16.

Em lições posteriores, aprenderemos a fazer Assembly manualmente: o conhecimento do sistema hexadecimal será um pré-requisito para entendê-las.

Com o tempo e a prática, você verá que a dificuldade desse sistema é apenas aparente. Na verdade, ele é tão simples e fácil quanto o sistema decimal ou qualquer outro.

Como devo fazer para reconverter de hexa para decimal?

Cada dígito de um número em hexa vale 16 vezes o dígito à sua direita. Para converter um número em hexa, como F6DA, por exemplo, deve-se tomar inicialmente o primeiro dígito da direita e convertê-lo para notação decimal. No exemplo, A equivale a 10. O próximo dígito à esquerda vale 16 vezes mais; assim, convertemos D para decimal (o que dá 13), e realizamos a operação: 13 × 16 = 208. O próximo dígito à esquerda vale 16 vezes mais; assim, multiplicamos 6 × 16 × 16 = 1536. O último dígito do exemplo precisa também ser multiplicado por 16. F vale 15, assim temos 15 × 16 × 16 × 16 = 61440. Somando todos os valores obtidos temos 10 + 208 + 1536 + 61440 = 63194, em decimal.

Ou então use o programa listado aqui para converter o número hexa de dois em dois dígitos de cada vez, e depois multiplique o par mais à esquerda por 256.

NÚMEROS MAIORES

Como um computador de oito bits representa números maiores do que 255? Simplesmente quebrando-os em duas partes de oito bits cada e colocando-os em duas memórias adjacentes. Assim, o computador armazena qualquer número até FFFF em hexa, ou 65 535 em decimal.

O FFFF é um valor importante em computadores pessoais de oito bits, pois é o número máximo de posições endereçáveis de memória RAM.

Números maiores ainda do que esses podem ser armazenados através do mesmo estratagema: dividi-los em três, quatro ou até mais porções de oito bits.

A maneira como esses bytes são organizados depende do computador. A maior parte dos micros que usam o BASIC armazenam os bytes de menor valor (chamados LSB, ou *least significant byte*) nas memórias de endereço mais baixo, e os bytes de maior valor (MSB, ou *most significant byte*), nas memórias de endereço mais alto.

Outros computadores, como os compatíveis com o TRS-Color, fazem exatamente o contrário.

Suponhamos agora que você queira encontrar o número decimal equivalente a dois hexadecimais armazenados em compartimentos vizinhos de memória (um número maior do que FF, quebrado em dois).

Nos microcomputadores das linhas Sinclair, TRS-80, MSX e Apple II basta converter os dois hexadecimais (H1 e H2) para decimal (D) e depois fazer a seguinte operação:

$$D = H1 - 256 + H2$$

Nos micros compatíveis com o TRS-Color, a expressão usada é a mesma, mas a ordem de H1 e H2 é inversa; ou seja, o byte de maior valor aparece antes do menor byte.

```
BIN. DEC. HEX.
BINARIO
1   1   0   0   1   0   0   1
DECIMAL
128 + 64 + 0 + 0 + 8 + 0 + 0 + 1
DECIMAL TOTAL = 201
HEXADECIMAL
8 + 4 + 0 + 0 = C  8 + 0 + 0 + 1 = 9
HEX TOTAL = C9
```

Veja como o programa de conversão entre bases aparece na tela dos microcomputadores. A disposição da tela é bastante parecida a esta, nos micros de outras linhas. Agora fica mais fácil entender como os três sistemas numéricos funcionam. Quando você manda executar o programa, as três linhas são zeradas. Pressione a letra B para diminuir o valor de 1 na tela, e a linha de binário ficará totalmente cheia de 1s. Logo abaixo, a linha decimal irá completar-se com todas as potências de 2. Da direita para a esquerda você terá os valores: 1 (que é 2), 2 (que é 2), 4 (ou 2), etc. A linha hexadecimal, mais abaixo, funciona da mesma forma, só que os dois dígitos hexa são computados independentemente.

O SISTEMA OCTAL

Existem apenas quatro sistemas de numeração que podem ser utilizados com vantagem na programação e operação de computadores digitais. Você já conhece três deles: o **sistema binário**, *que é o natural para um computador digital; o* **sistema decimal**, *espontaneamente utilizado pelo ser humano; e o* **sistema hexadecimal**, *cuja base (16) é uma potência de 2. Essa correspondência facilita enormemente sua utilização pelo programador.*

O quarto sistema de numeração que pode ser usado com computadores digitais é o **sistema octal**, *ou seja, de base 8, que é também uma potência do número 2. Apesar disso, o octal está entrando em desuso, pois não foi universalmente adotado pelos fabricantes de microcomputadores.*

O octal tem 8 dígitos: 0, 1, 2, 3, 4, 5, 6, 7 e 8. O sistema de conversão de um número em octal para decimal é semelhante ao usado com o hexadecimal, só que o multiplicador é 8. Por exemplo:

$$423 = 4 \times 64 + 2 \times 8 + 3 = 275$$

Alguns computadores pessoais permitem representar constantes em octal **(&0)**, *ou têm funções de conversão* **(OCT$)**.

CLASSIFICAÇÃO DOS MICROCOMPUTADORES POR LINHA

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

CLASSIFICAÇÃO DOS MICROCOMPUTADORES POR NOME

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

UM LOGOTIPO PARA CADA MODELO DE COMPUTADOR

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


Sinclair ZX-81


TRS-80


TK-2000


MSX


Spectrum


TRS-Color


Apple II

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

NO PRÓXIMO NÚMERO

PROGRAMAÇÃO DE JOGOS
Aprenda a fazer contagem de tempo e de pontos no computador, tornando seus jogos ainda mais emocionantes.

APLICAÇÕES
Como utilizar o computador para organizar seu arquivo e manter em ordem as suas coleções. Armazene informações.

PROGRAMAÇÃO BASIC
O que são, na prática, as instruções **GOTO** e **GOSUB**. Programas para cálculo, adivinhação e jogos de dados.