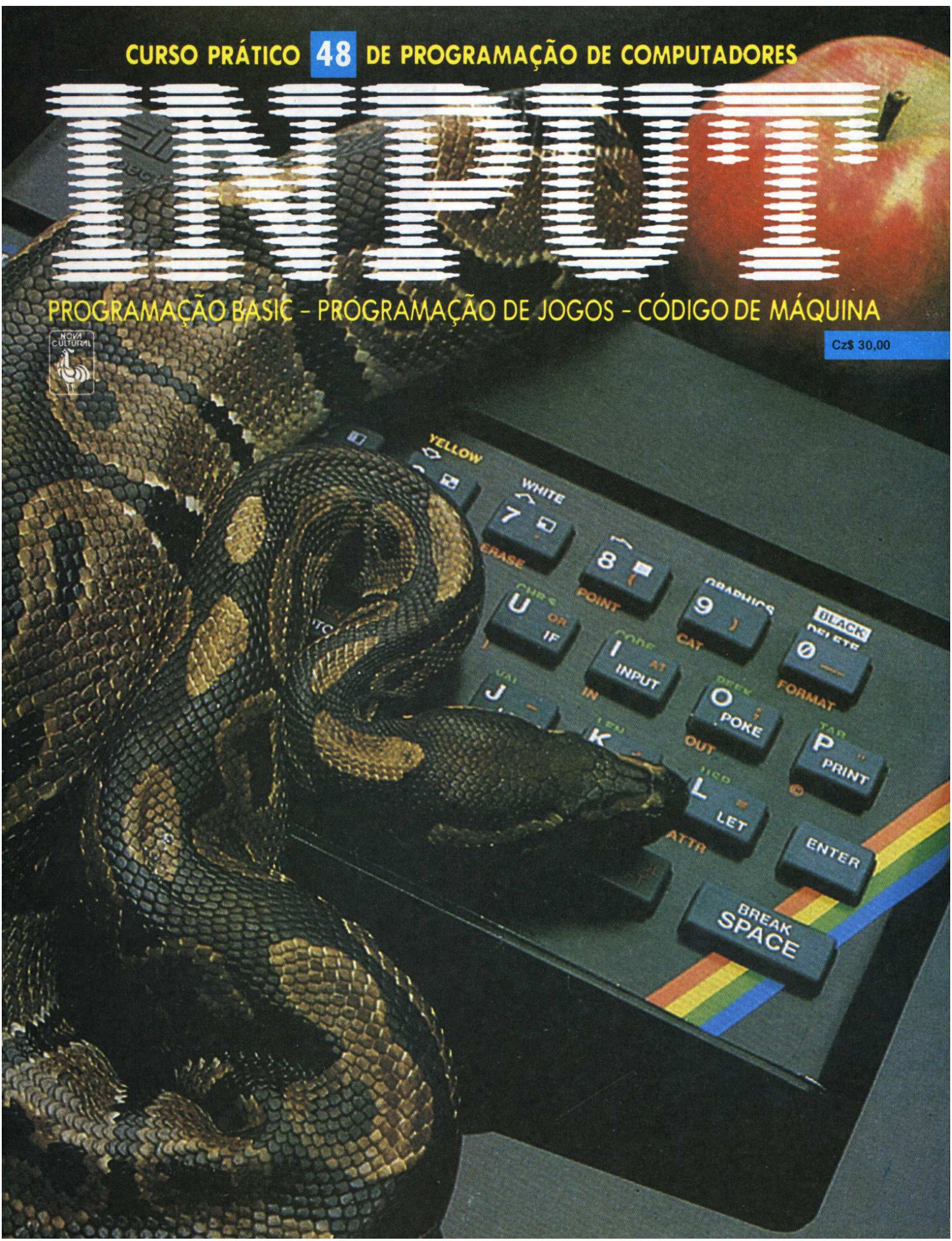
CURSO PRÁTICO 48 DE PROGRAMAÇÃO DE COMPUTADORES

TRATO

PROGRAMAÇÃO BASIC - PROGRAMAÇÃO DE JOGOS - CÓDIGO DE MÁQUINA

NOVA CULTURAL

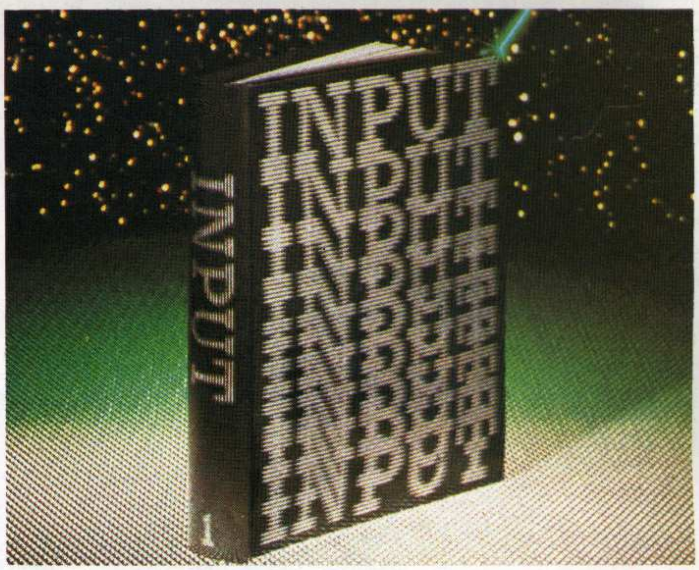
Cz$ 30,00



INPUT

Vol. 4 Nº 48

NESTE NÚMERO



PLANO DA OBRA

"INPUT" é uma obra editada em fascículos semanais, e cada conjunto de 15 fascículos compõe um volume. A capa para encadernação de cada volume estará à venda oportunamente.

COMPLETE SUA COLEÇÃO

Exemplares atrasados, até seis meses após o encerramento da coleção, poderão ser comprados, a preços atualizados, da seguinte forma: 1. PESSOALMENTE — Por meio de seu jornaleiro ou dirigindo-se ao distribuidor local, cujo endereço poderá ser facilmente conseguido junto a qualquer jornaleiro de sua cidade. Em **São Paulo**, os endereços são: rua Brigadeiro Tobias, 773, Centro; av. Industrial, 117, Santo André; e no **Rio de Janeiro**: rua da Passagem, 93, Botafogo. 2. POR CARTA — Poderão ser solicitados exemplares atrasados também por carta, que deve ser enviada para DINAP — Distribuidora Nacional de Publicações — Números Atrasados — Estrada Velha de Osasco, 132, Jardim Teresa — CEP 06000 — Osasco — SP. Não envie pagamento antecipado. O atendimento será feito pelo reembolso postal e o pagamento, incluindo as despesas postais, deverá ser efetuado ao se retirar a encomenda na agência do Correio. 3. POR TELEX — Utilize o nº (011) 33 670 DNAP.
Em **Portugal**, os pedidos devem ser feitos à Distribuidora Jardim de Publicações, Lda. — Qta. Pau Varais, Azinhaga de Fetais — 2 685, Camarate — Lisboa; Apartado 57 — Telex 43 069 JARLIS P.
Atenção: Após seis meses do encerramento da coleção, os pedidos serão atendidos dependendo da disponibilidade do estoque.
Obs.: Quando pedir livros, mencione sempre título e/ou autor da obra, além do número da edição.

COLABORE CONOSCO

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

Editor
VICTOR CIVITA

REDAÇÃO
Diretor Editorial: Carmo Chagas

Editores Executivos: Antonio José Filho, Berta Sztark Amar

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

COLABORADORES

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

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

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

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

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

RISCOS E PRÊMIOS AVALANCHE

Precisamos de incentivos e prêmios — como bolo e limonada — para ir em frente em nossa aventura. Mas não relaxe: as serpentes assassinas estão sempre à espera de um descuido.

Depois de montado, o cenário requer apenas algumas modificações para se adaptar aos diferentes níveis de dificuldade. No segundo nível, o grande problema de Willie consiste nos buracos existentes ao longo da encosta. Para desenhá-los, colocam-se blocos gráficos da cor do céu sobre o perfil da montanha. As cobras surgem apenas a partir do terceiro nível de dificuldade. A rotina que as movimenta será incorporada ao programa mais tarde.

- ROTINAS ENCARREGADAS DE FORNECER OS PRÊMIOS
- NÍVEIS DE DIFICULDADE
- BURACOS E MAIS BURACOS
- DESENHO DAS COBRAS ASSASSINAS
- ROTINA DE IMPRESSÃO DAS FIGURAS
- COMO TESTAR O PROGRAMA

```
 10 REM org 58455
 20 REM elb ld a,(57344)
 30 REM ld hl,57272
 40 REM ld b,a
 50 REM inc b
 60 REM ld de,8
 70 REM ab add hl,de
 80 REM djnz ab
 90 REM push hl
100 REM pop bc
110 REM ld hl,191
120 REM ld a,58
130 REM call print
140 REM ld a,(57344)
150 REM cp 0
160 REM jr z,ed
170 REM push af
180 REM call hls
190 REM pop af
200 REM cp 1
210 REM jr z,ed
220 REM call snp
230 REM ed ret
240 REM hls ld hl,457
250 REM call hlp
260 REM ld hl,401
270 REM call hlp
280 REM  ld hl,314
290 REM call hlp
300 REM ret
310 REM hlp ld b,4
320 REM hlq push bc
330 REM ld bc,15616
340 REM ld a,45
```

```
350 REM call print
360 REM ld de,32
370 REM add hl,de
380 REM pop bc
390 REM djnz hlq
400 REM ret
410 REM snp ld hl,457
420 REM call snq
430 REM ld hl,401
440 REM call snq
450 REM ld hl,314
460 REM call snq
470 REM ret
480 REM snq ld a,4
490 REM ld bc,57232
500 REM snr push af
510 REM ld a,43
520 REM call print
530 REM ld de,32
540 REM add hl,de
550 REM pop af
560 REM dec a
570 REM jr nz,snr
580 REM ret
590 REM print org 58217
```

OS PRÊMIOS

O conteúdo do endereço 57344 é colocado no acumulador. Esta posição de memória será utilizada para armazenar o nível de dificuldade em vigor. O par de registros HL recebe a seguir o endereço inicial dos padrões dos prêmios na tabela de dados. O nível de dificuldade é transferido do acumulador para o registro B, aumentando, então, em uma unidade. O par DE recebe o número 8 e é somado ao par HL.

O laço rotulado com **ad** prossegue adicionando 8 ao apontador HL até que o conteúdo de B se torne zero. Lembre-se de que a instrução **djnz** opera sobre o registro B: subtrai uma unidade e salta se o conteúdo deste não tiver chegado a zero. Tal processo movimenta o apontador do padrão do prêmio — o par de registro HL — pela tabela, até que ele aponte para o início do bloco gráfico adequado ao nível de dificuldade vigente. Cada bloco é um quadrado de oito por oito pontos, ocupando, assim, oito bytes na tabela.

O resultado dessas adições repetidas fica no par de registros HL. Contudo, a sub-rotina **print** precisa dele no par BC para imprimir o bloco correspondente. A maneira mais fácil e rápida de promover a transferência consiste em guardar o valor de HL na pilha, recuperando-o em seguida em BC.

A cor do caractere do prêmio é codificada pelo número 58, colocado em A. Depois disso, o programa chama a sub-rotina **print**, que desenha o bloco na cor apropriada. Essa rotina foi fornecida no primeiro artigo da série *Avalanche* (página 748).

O NÍVEL DE DIFICULDADE

O conteúdo da posição de memória usada para guardar o nível de dificuldade volta para o acumulador. O computador verifica inicialmente se ele é igual a zero. Em caso afirmativo, a instrução **jr z,ed** salta para o final da rotina, e a instrução **ret** provoca o retorno da mesma.

Se o nível vigente for diferente de zero, o processador preserva seu número, guardando-o na pilha. A sub-rotina **hls**, que desenha os buracos ao longo da encosta, é chamada a seguir. O nível de dificuldade é então recuperado da pilha e comparado a 1. Nesse nível, temos vários buracos mas não existe nenhuma serpente dentro deles. Assim, se A contiver 1, a instrução **cp 1** resulta em zero, o que ativa o indicador de zeros. A instrução **jr z,ed** salta, então, para o final da rotina, onde **ret** provoca o retorno.

Se o nível for maior que 1, a sub-rotina **snp**, que desenha as serpentes assassinas, é chamada. Depois disso, o programa termina.

Quando a rotina principal de controle estiver na memória, o processador retornará a ela. Por enquanto, ele retorna ao BASIC.

OS BURACOS

A sub-rotina seguinte tem a função de cavar buracos ao longo da encosta. Ela começa com o rótulo **hls**, que é chamado pela rotina principal quando é preciso acrescentar os buracos ao cenário. Três pequenos módulos, com duas instruções cada um, compõem essa sub-rotina. Eles começam com comandos do tipo **ld hl**, que colocam no par HL o endereço da tela correspondente à posição de impressão do topo de cada buraco. Após cada instrução desse tipo, a rotina **hlp** é chamada para colocar os blocos que formam o buraco sobre o desenho da encosta, feito pelos programas dados no último artigo.

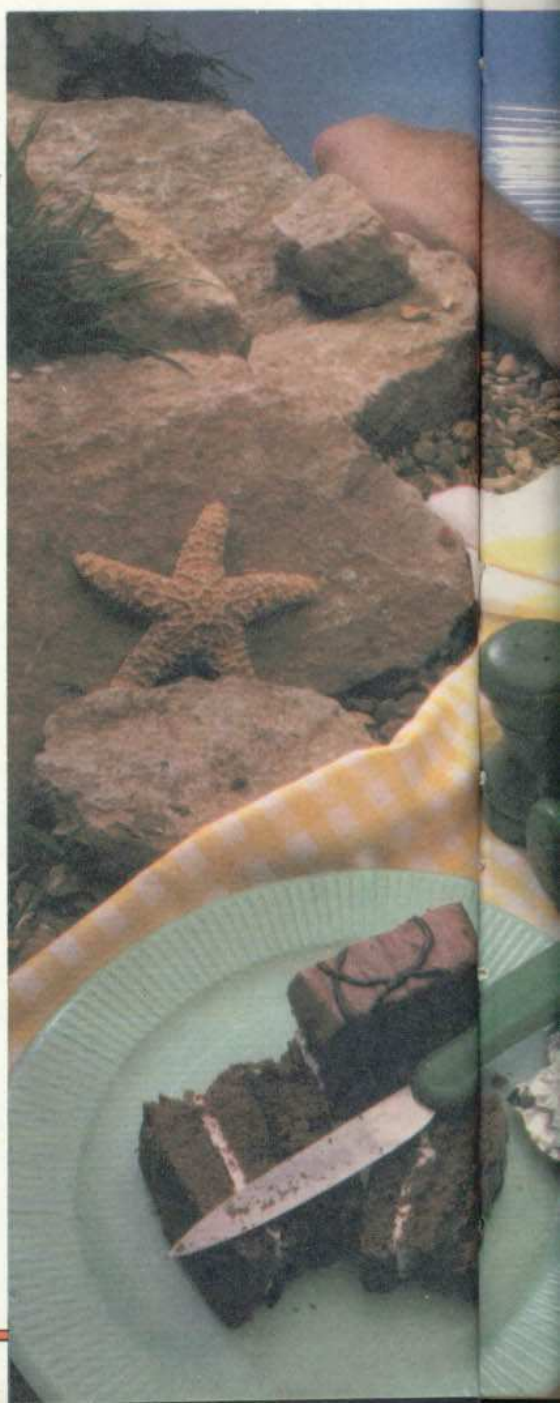
A sub-rotina contém três módulos porque deve desenhar três buracos, cada um em uma posição diferente colocada em HL. Todos eles são feitos da mesma maneira, pela sub-rotina **hlp**.

MAIS BURACOS

Como cada buraco tem quatro blocos de profundidade, o número 4 é colocado no registro B e guardado temporariamente na pilha. O par BC recebe, então, o endereço de um espaço em branco. Na realidade, este é um byte da tabela de códigos para o alto da tela.

O código da cor do céu — 45 — vai para o acumulador e a sub-rotina **print** é novamente chamada. Assim, para colocar o primeiro bloco cor do céu sobre o perfil da encosta, o computador tira a primeira "pá de terra" do local onde ficará o buraco.

Colocando 32 em DE e somando esse valor a HL, fazemos com que este par aponte para o endereço da posição imediatamente inferior. O contador é recuperado da pilha e a instrução **djnz** subtrai uma unidade dele, saltando em direção ao rótulo **hlq**, se B ainda não for



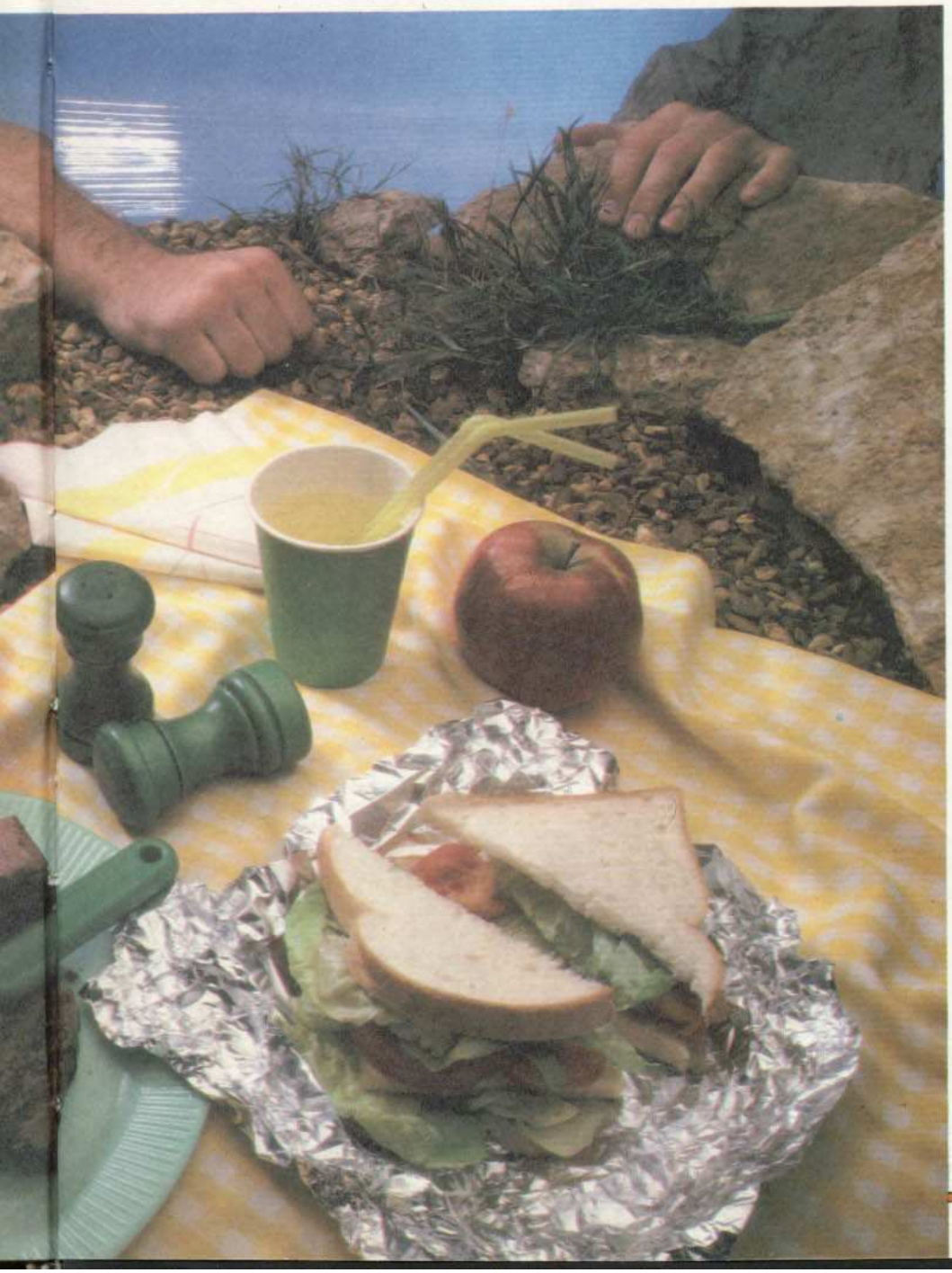
zero. O processador executa esse laço quatro vezes, imprimindo mais um bloco cor do céu — ou tirando mais uma pá de terra — a cada volta.

AS COBRAS

O resto do programa desenha as cobras. As sete primeiras instruções são parecidas com as que iniciam a rotina dos buracos. Elas colocam no par HL a posição de impressão de cada cobra, chamando em seguida a sub-rotina que desenha a serpente.

Essa rotina também se assemelha àquela que desenha os buracos — afinal, as cobras devem caber dentro deles.

Desta vez, o contador ficará no acumulador, pois precisaremos controlar o apontador da tabela de padrões. No caso dos buracos, esse apontador foi direcionado para o endereço de um espaço em branco. Aqui ele percorrerá a tabela que criamos na memória RAM. A sub-rotina **print** atualiza automaticamente o apontador, movendo-o para o próximo byte. Como o apontador fica no par de registros BC, teríamos que ficar guardando seu valor na pilha, se quiséssemos recorrer à instrução **djnz** para controlar o laço. Com o contador em A, essa instrução é substituída por **dec a** e **jr nz, snr**.



TESTE DA ROTINA

Para testar a rotina, coloque níveis de dificuldade de 0 a 3 no endereço 57344, usando o comando **POKE 57344**. Chame a rotina com **RANDOM 58455** e verifique se os buracos estão sendo desenhados no nível 1; as cobras, nos níveis 2 e 3, e se prêmios diferentes aparecem a cada nível.

T

O programa listado a seguir desenha os buracos e as serpentes assassinas ao longo da encosta, conforme o nível de dificuldade.

```
10   ORG 19289
20   ELB LDA 18238
30   LDB #16
40   MUL
50   ADDD #18142
60   TFR D,U
70   LDX #2782
80   JSR CHARPR
90   LDA 18238
100   BEQ ED
110   PSHS A
120   JSR HOLES
130   PULS A
140   CMPA #1
150   BEQ ED
160   JSR SNAKE
170   ED RTS
180   HOLES LDX #5095
190   LDU #3071
200   JSR HOLPR
210   LDX #4591
220   LDU #3071
230   JSR HOLPR
240   LDX #3833
250   LDU #3071
260   JSR HOLPR
270   RTS
280   SNAKE LDX #5095
290   LDU #18078
300   JSR HOLPR
310   LDX #4591
320   LDU #18078
330   JSR HOLPR
340   LDX #3833
350   LDU #18078
360   JSR HOLPR
370   RTS
380   HOLPR LDB #4
390   HOLPRI PSHS U,X,B
400   JSR CHARPR
410   PULS B,X,U
420   LEAX 256,X
430   LEAU 16,U
440   DECB
450   BNE HOLPRI
460   RTS
470   CHARPR LDB #2
480   CHARI PSHS B,X
490   LDB #8
500   CHARZ PULU A
510   STA ,X
```

```
520  LEAX 32,X
530  DECB
540  BNE CHARZ
550  PULS X,B
560  LEAX 1,X
570  DECB
580  BNE CHARI
590  RTS
```

COMO TESTAR O PROGRAMA

Esse programa acrescenta blocos gráficos ao cenário já montado. Portanto, para testá-lo, é preciso que as rotinas e tabelas criadas anteriormente estejam na memória do micro. Depois de carregar os códigos e montar o programa, digite as linhas BASIC que seguem. Elas permitirão que você execute a parte do jogo já programada.

```
5 PCLEAR4:CLEAR 200,16999
10 EXEC 19000:EXEC 19109
20 POKE 18238,0
30 EXEC 19289
40 GOTO 40
```

A linha 10 executa parte do jogo montada nos artigos anteriores; a 20 acerta o valor do nível de dificuldade e a 30 executa o programa deste artigo. A 40 serve apenas para manter a tela de alta resolução ligada.

Para termos certeza de que nossa rotina funciona perfeitamente, devemos interromper o programa com **BREAK** e editar a linha 20 para mudar o nível de dificuldade. O nível 0 deixa o cenário intacto, apenas com a montanha — os perigos desse nível são as pedras, que faremos rolar morro abaixo em outra parte do programa.

O nível 1 desenha os buracos na encosta e o 2 acrescenta as serpentes assassinas. O nível 3 soma a tudo isso a avalanche, que, como dissemos, será promovida por outra rotina.

Teste cada um dos níveis. Verifique se as guloseimas roubadas estão sendo desenhadas no topo da montanha.

NÍVEIS DE DIFICULDADE

O endereço 18328 será usado para guardar o nível de dificuldade em vigor. Seu conteúdo transfere-se para o acumulador no início do programa e, em seguida, o número 16 é colocado no registro B. A instrução **MUL** multiplica o conteúdo desses dois registros, colocando o resultado em D.

Como o bloco gráfico de cada prêmio tem dezesseis bytes de comprimento, precisamos multiplicar o nível vigente por dezesseis, para obtermos na tabela o padrão do bloco a ele adequado.

O número obtido pela soma de 18142 ao resultado da multiplicação indica a posição inicial do padrão do bloco desejado, na tabela de padrões. Esse apontador é então transferido para o registro U, que funciona como apontador da pilha do usuário.

O registro X recebe o valor 2814, posição de impressão do prêmio na tela gráfica. Em seguida, o processador salta para o rótulo **CHARPR**, que imprime o bloco gráfico correspondente.

A instrução **LDA 18238** repõe o nível de dificuldade dentro do acumulador. Se o nível for zero, a instrução **BEQ** provoca um desvio para o rótulo **ED**. Como neste nível já nada mais precisa ser acrescentado ao cenário — com exceção do prêmio —, a instrução **RTS** termina a sub-rotina.

Se o nível for maior que zero, o programa guarda seu número na pilha da máquina. A sub-rotina que cava os buracos na encosta é, então, chamada.

Depois que a escavação termina, o nível é recuperado da pilha e comparado a 1. Se este for o número do nível atual, a instrução que desenha as cobras, **JSR SNAKE**, é ignorada e o processador retorna da sub-rotina.

BURACOS E COBRAS

As rotinas que desenham os buracos e as cobras compõem-se de três séries de três instruções, e funcionam da mesma maneira. Em cada série, o registro X recebe a posição da tela onde deve ser impresso o topo do bloco gráfico, enquanto o registro U recebe o endereço inicial do padrão do bloco na tabela. Em seguida, o processador salta para a rotina de impressão, **HOLPR**.

Como todos os buracos são iguais, o apontador da tabela tem sempre o valor inicial 3071. Esse endereço não fica na tabela, e sim na tela, pois o buraco nada mais é que um bloco cor do céu. O mesmo acontece com as cobras, cujo bloco gráfico tem endereço inicial 18078 na tabela de padrões.

As cobras são impressas, evidentemente, na mesma posição que os buracos que ocupam. O topo do primeiro buraco fica em 5095; o segundo, em 4591 e o terceiro, em 3833. Tanto as cobras quanto os buracos são desenhados sobre o perfil da montanha original. O restante do cenário permanece intacto.

A ROTINA DE IMPRESSÃO

As rotinas **HOLES** e **SNAKES** chamam a sub-rotina **HOLPR**, que é responsável tanto pelo desenho dos buracos quanto pelo das cobras.

O registro B recebe inicialmente o número 4 — os buracos têm quatro blocos de profundidade. Os registros U, X e B são guardados na pilha antes que a rotina **CHARPR** seja chamada. Esta é responsável pela impressão do bloco indicado pelo apontador de tabela de padrões na posição determinada por X.

Assim, se estivermos no nível 1 e formos desenhar um buraco, o apontador da tabela, U, indicará seu padrão e a rotina se encarregará de colocá-lo na posição dada por X. Se estivermos no nível 2 ou 3, U apontará para o padrão do bloco da cobra, que será impresso na posição X da tela.

Uma vez feita a impressão B, X e U são recuperados da pilha. X recebe então o valor X+256, que corresponde à posição de impressão imediatamente inferior. U recebe o valor U+16, que aponta para o padrão do próximo bloco gráfico na tabela.

Em seguida, B é subtraído em uma unidade e, se ainda não foi reduzido a zero, o processador volta ao rótulo **HOLPRI**. Quando o conteúdo desse registro chegar a zero, todos os quatro

blocos terão sido desenhados, a instrução **BNE HOLPRI** será ignorada e o processador retornará da sub-rotina.

A impressão do bloco é feita pela rotina **CHARPR**, que explicamos a seguir.

Cada bloco gráfico impresso na tela tem dezesseis bits — dois bytes, portanto — de largura e oito bits de profundidade. O registro B recebe, assim, o valor 2, para contar a largura do buraco. Esse contador é preservado na pilha, juntamente com a posição de impressão contida no registro X. B recebe então o valor 8, para contar a profundidade do bloco.

O byte correspondente ao padrão da linha do bloco é recuperado da pilha do usuário, colocado no acumulador e, em seguida, transferido para a posição de impressão apontada pelo registro X. Por meio desse processo, o bloco é impresso linha por linha na tela.

A instrução **LEAX 32,X** coloca o valor X+32 no registro X, movendo o apontador de posições de tela uma linha de pixels para baixo. O contador B é diminuído em uma unidade e o processador retorna para imprimir o próximo byte, até que B tenha sido reduzido a zero — **BNE CHARZ**.

Em seguida, X e B são recuperados da pilha e X recebe o valor X+1 — **LEAX 1,X** —, o que faz o apontador mover-se um byte para a direita na tela. O contador B é então subtraído de uma unidade. Se ele ainda não tiver sido reduzido a zero, o processador retorna ao rótulo **CHARI** para imprimir mais uma coluna de oito bytes. Quando B já for igual a zero, as duas colunas estarão desenhadas e o processador retornará da sub-rotina **CHARPR**.

```
10    org 53682
20    ld a,(-5228)
30    rla
40    rla
50    ld b,56
60    add a,b
70    ld hl,(62407)
80    ld de,254
90    add hl,de
100    call 77
110    ld a,(-5228)
120    cp 0
130    jr z,ed
140    push af
150    call ho
160    pop af
170    cp 1
180    jr z,ed
190    call sn
200    ed ret
210    ho ld de,457
220    push de
230    ld de,401
240    push de
250    ld de,314
260    push de
270    ld c,3
280    po ld hl,(62407)
290    pop de
300    ld b,4
310    pp add hl,de
320    ld a,254
330    push hl
340    push bc
350    call 77
360    pop bc
370    pop hl
380    ld de,32
390    djnz pp
400    dec c
410    jr nz,po
420    ret
430    sn ld a,37
440    ld (-5227),a
450    add a,3
460    ld (-5226),a
470    add a,1
480    ld (-5225),a
490    add a,3
500    ld (-5224),a
510    ld de,457
520    push de
530    ld de,401
540    push de
550    ld de,314
560    push de
570    ld c,3
580    po ld hl,(62407)
590    ld de,-5227
600    ld (-5220),de
610    pop de
620    ld b,4
630    pp add hl,de
640    ld de,(-5220)
650    ld a,(de)
660    inc de
670    ld (-5220),de
680    push hl
690    push bc
700    call 77
710    pop bc
720    pop hl
730    ld de,32
740    djnz pp
750    dec c
760    jr nz,po
770    ret
780    end
```

ROTINA DE PRÊMIOS

O conteúdo do endereço -5228 é colocado no acumulador. O nível de dificuldade do jogo ficará armazenado nessa posição de memória.

O valor de A (0, 1, 2 ou 3) será utilizado para gerar o código da tabela de padrões que corresponde ao prêmio no nível vigente, ou seja:

nível	valor de A	prêmio	código
1	0	sanduíche	56
2	1	maçã	60
3	2	garrafa	64
4	3	sorvete	68

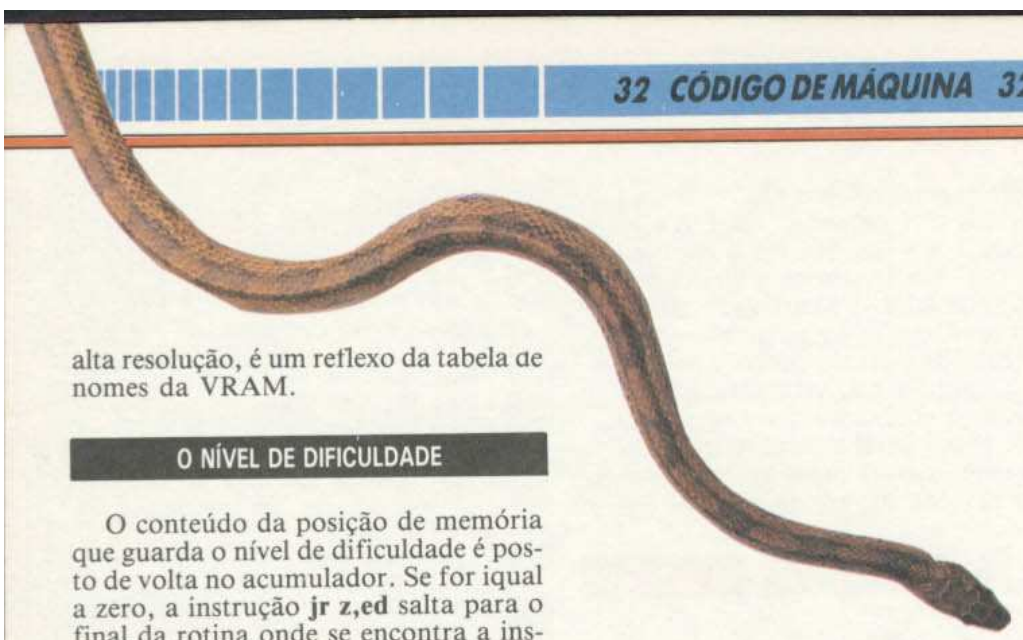
Armazenamos o código de padrão do sanduíche, 56, no registro B. Este será, portanto, o código-base. Em seguida, multiplicamos o conteúdo de A por quatro — a operação equivale a deslocar o conteúdo de A dois bits para a esquerda, o que é feito pela instrução **rla** executada duas vezes.

Somando o código-base contido em B ao valor de A, obtemos no acumulador o código do prêmio certo.

O endereço inicial da tabela de nomes de VRAM, que está nas posições de memória 62407 e 62408, é colocado no par de registros HL. Adicionamos a esse endereço o deslocamento correspondente à posição na qual desejamos imprimir o prêmio — 256, que equivale ao topo da montanha.

Para a impressão do bloco correspondente ao prêmio, utiliza-se a sub-rotina 77 da memória ROM. Esta coloca o valor do acumulador A na posição da VRAM apontada pelo par HL. No nosso caso; ela coloca o código do bloco do prêmio na tabela de nomes.

Lembre-se de que a tela, no modo de



alta resolução, é um reflexo da tabela de nomes da VRAM.

O NÍVEL DE DIFICULDADE

O conteúdo da posição de memória que guarda o nível de dificuldade é posto de volta no acumulador. Se for iqual a zero, a instrução **jr z,ed** salta para o final da rotina onde se encontra a instrução **ret**. Caso contrário, o processador continua preservando o número do nível na pilha. A sub-rotina **hls**, que desenha os buracos ao longo da encosta, é então chamada.

O nível de dificuldade é recuperado da pilha e comparado a 1. Nesse nível temos vários buracos, mas nenhuma serpente dentro deles. Logo, se A contiver 1, a instrução **cp 1** resulta em zero, o que ativa o indicador de zeros. A instrução **jr z,ed** salta novamente para o final da rotina e retorna.

Se o nível for superior a 1, a sub-rotina **snp**, que desenha as serpentes assassinas, é chamada e, depois, o programa termina. Quando a rotina principal de controle estiver na memória, o programa retornará a ela. Por enquanto, retorna ao BASIC.

OS BURACOS

A próxima sub-rotina é totalmente dedicada a cavar buracos ao longo da encosta. Ela começa com o rótulo **ho**, chamado pela rotina principal quando surge a necessidade de acrescentar os buracos ao cenário. Inicialmente, **ho** armazena na pilha o endereço da tela correspondente à posição de impressão do topo de cada buraco. Como são três buracos, isso é feito três vezes.

Todos são desenhados da mesma maneira, por intermédio do laço **ho** que coloca no par de registros HL o endereço inicial da tabela de nomes da VRAM, adicionando a ele o endereço da tela correspondente a cada buraco.

O registro C é usado como contador para os três buracos, e a instrução **jr nz, ho** testa o fim da rotina.

Como cada buraco tem quatro blocos de profundidade, esse valor é colocado no registro B, que contém o número de execuções do laço **hp**.

0 código da tabela de padrões do bloco cor de céu é armazenado no acumulador A. Em seguida, a rotina 77 da ROM, cujo funcionamento explicamos anteriormente, é chamada.

Para evitar possíveis alterações de correntes da rotina 77, as instruções **pusch** e **pop** são usadas para proteger os contadores no par BC e os endereços no par HL. O contador é recuperado da pilha e a instrução **djnz** subtrai uma unidade dele, saltando em direção ao rótulo **hp** se o registro B ainda não tiver sido reduzido a zero.

O laço, executado quatro vezes, tira uma "pá de terra" a cada volta.

AS COBRAS ASSASSINAS

Cabe à rotina **sn** desenhar as cobras. O processo é basicamente igual ao da rotina **ho** e as cobras são desenhadas nas mesmas posições que os buracos. Há uma diferença fundamental: para cavar um buraco, precisamos colocar o mesmo padrão do bloco cor de céu quatro vezes; já para desenhar uma cobra temos que mudar o código do padrão para cada parte do animal.

Essa tarefa seria muito simples se os códigos dos padrões que formam a cobra estivessem em seqüência. Porém, como você pode verificar na tabela de padrões, isso não ocorre. Armazenamos, assim, no início da rotina, os códigos dos padrões que formam a cobra — 37, 40, 41, 44 — nos endereços de -5227 até -5224. Depois, basta assegurar que esses endereços sejam chamados seqüencialmente para cada cobra a ser desenhada. Isso é feito por intermédio do par DE, que serve como ponte, armazenando o endereço inicial da tabela (-5227) nas posições -5220 e -5219. Esses endereços contêm o endereço do código do padrão correspondente a cada uma das partes da cobra.

No mais, esta rotina é idêntica à que cava buracos.

TESTANDO

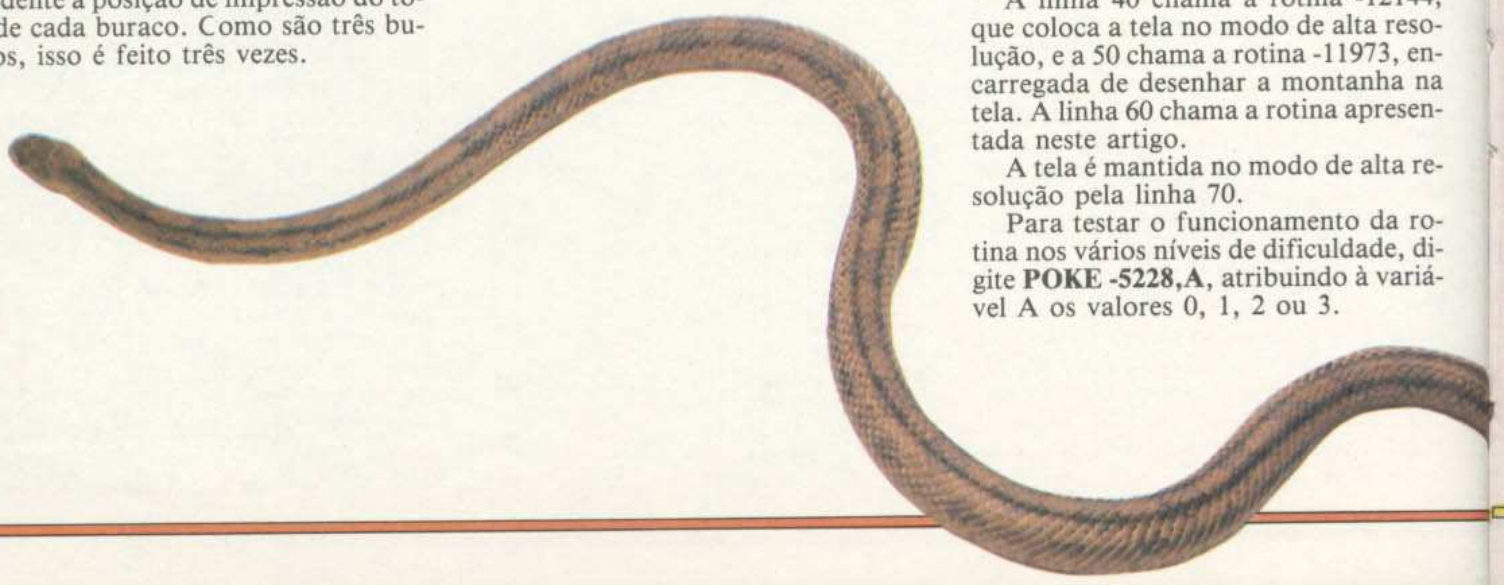
Para testar a rotina, digite o seguinte programa BASIC:

```
10 DEFUSR1=-12144
20 DEFUSR2=-11973
30 DEFUSR3=-11854
40 A=USR1(0)
50 A=USR2(0)
60 A=USR3(0)
70 GOTO 70
80 END
```

A linha 40 chama a rotina -12144, que coloca a tela no modo de alta resolução, e a 50 chama a rotina -11973, encarregada de desenhar a montanha na tela. A linha 60 chama a rotina apresentada neste artigo.

A tela é mantida no modo de alta resolução pela linha 70.

Para testar o funcionamento da rotina nos vários níveis de dificuldade, digite **POKE -5228,A**, atribuindo à variável A os valores 0, 1, 2 ou 3.



OS SEGREDOS DO TRS-80 (2)

- CÓPIA DA TELA
- FUNÇÃO **VARPTR** E VÍDEO
- TRANSFERÊNCIA
- APLICAÇÕES
- IMPRIMINDO A TELA

Pode-se consultar, com o BASIC, as posições da memória que armazenam o vídeo e as cadeias de caracteres. Usando esse recurso, você tornará seus programas bem mais rápidos.

No artigo anterior desta série, vimos como manipular o conteúdo da memória de vídeo do TRS-80 e compatíveis com o auxílio dos comandos **PEEK** e **POKE**. Podemos fazer isso porque o vídeo do TRS-80 é mapeado diretamente em um segmento fixo da RAM, que começa na locação 15360 e se estende até a locação 16376. Cada caractere na tela é armazenado em uma posição correspondente dessa memória de 1024 bytes.

Porém, os comandos **PEEK** e **POKE** são muito lentos quando usados dentro de laços **FOR...NEXT** para a transferência, consulta ou escrita de grande quantidade de caracteres na tela. Há uma técnica bem mais poderosa e rápida, que aprenderemos neste artigo: é o uso da função **VARPTR** como apontador de cadeias de caracteres para a tela.

Essa técnica permite armazenar instantaneamente 255 caracteres em exibição na tela, em uma cadeia alfanumérica (*string*). O limite de 255 caracteres (que corresponde a quatro linhas de vídeo) é imposto pelo tamanho máximo de um *string* alfanumérico. Se quisermos deslocar a tela inteira, portanto, basta efetuar a transferência em "partes" de 255 bytes de cada vez.

Para entender como a técnica funciona, precisamos conhecer o processo de armazenagem de cadeias alfanuméricas na memória do TRS-80. Existe um espaço na memória RAM, criado toda vez que o BASIC é inicializado. O argumento do comando **CLEAR** define o tamanho desse espaço (o tamanho predefinido é de cinqüenta bytes).

Sempre que uma variável alfanumérica é definida (ou seja, sempre que não é uma constante dentro de um programa), o interpretador BASIC cria um par de bytes *apontadores*, que contém o endereço absoluto inicial da cadeia, na memória de *strings*. Nessa locação inicial, está armazenado um byte, que contém o comprimento da cadeia em bytes (daí a limitação intrínseca de 255 bytes por cadeia). Nas duas locações seguintes, encontram-se os bytes (o menos significativo e o mais significativo) do endereço inicial onde está armazenado o conteúdo da cadeia alfanumérica.

Seu programa pode ter acesso direto a toda essa informação através da função **VARPTR**. Para uma variável **V$**, por exemplo, teríamos:

VARPTR(V$) = locação do apontador de **V$**
PEEK(VARPTR(V$)) = comprimento de **V$**
PEEK(VARPTR(V$)+1) = endereço de **V$**
PEEK(VARPTR(V$)+2) = endereço de **V$**

Nada mais fácil, portanto, do que transferir 255 bytes da tela para uma variável alfanumérica: basta alterar o endereço armazenado nas locações **VARPTR(V$)+1** e **VARPTR(V$)+2** de modo que passe a apontar para a locação inicial da memória de vídeo que queremos transferir. Nenhum byte do conteúdo da tela precisa ser movido!

A sub-rotina seguinte coloca o comprimento desejado e o endereço inicial de vídeo no bloco de apontadores de uma variável **V$**:

```
1000 'SUBROTINA VTRANSF
1010 V$=" ":POKE VARPTR(V$),C%
1020 POKE VARPTR(V$)+1, V%-INT
(V%/256)*256
1030 POKE VARPTR(V$)+2, INT(V%
/256)+60
1040 RETURN
```

A linha 1010 define **V$** e armazena no primeiro endereço do bloco de apontadores o número de bytes de comprimento (a variável inteira **C%**).

As linhas 1020 e 1030 tratam de armazenar no bloco de apontadores, respectivamente, o byte mais significativo e o byte menos significativo do endereço desejado de início no vídeo (a variável inteira **V%**).

Observe que as variáveis **C%** e **V%** estão definidas como inteiras. Para usar a sub-rotina, deve-se colocar em **V%** o endereço da posição de início na tela (0 a 1023), e, em **C%**, o número de bytes a se transferir (0 a 255). Ao retornar da sub-rotina, **V$** conterá os caracteres que estavam armazenados na tela, da posição **V%** a **V%+C%**.

APLICAÇÕES

Essa rotina, que chamaremos de **VTRANSF**, possui numerosas aplicações. Eis aqui algumas delas:

- copiar o conteúdo de uma tela de vídeo na impressora;
- armazenar o conteúdo de várias telas de vídeo na memória;
- armazenar as telas em disco;
- realizar animações gráficas, pela superposição rápida de porções de telas de vídeo previamente armazenadas em memória;
- montar vídeos com várias "janelas" superpostas, independentes.

Nos próximos artigos desta série, trataremos em detalhe das aplicações da rotina **VTRANSF**.

COMO IMPRIMIR A TELA

O programa que se segue é um primeiro exemplo de utilização da sub-rotina **VTRANSF**. Ele imprime o conteúdo corrente da tela em uma impressora, linha por linha:

```
10 CLEAR 500
20 C%=64
30 FOR V%=0 TO 960 STEP 64
40 GOSUB 1000 : LPRINT V$
50 NEXT V%
60 END
```

O programa funciona da seguinte maneira: a linha 10 reserva quinhentos bytes de memória para *strings*. A linha 20 define o comprimento **C%** da variável alfa que receberá cópia de uma linha de vídeo. A linha 30 inicia um laço, que vai até a linha 50, percorrendo os endereços de vídeo (**V%**) desde a primeira até a última linha, de 64 em 64 posições. Finalmente, a linha 40 chama nossa sub-rotina **VTRANSF** (que deverá ser colocada logo após a linha 60) e imprime a variável **V$**, que recebeu a cópia do vídeo.

Se sua impressora não for do tipo gráfico, use esse programa apenas para copiar telas de texto, pois ele não converte os caracteres gráficos (129 a 191) que encontra na tela.

O JOGO A RAPOSA E OS GANSOS (3)

As rotinas deste artigo permitem ao computador não só jogar pela raposa e pelos gansos como também estudar os próximos movimentos para melhorar o nível da partida.

Terceira e última parte do jogo *A Raposa e os Gansos*, este artigo apresenta rotinas que possibilitam ao computador jogar tanto pela raposa quanto pelos gansos. Na verdade, ele pode ser colocado a jogar sozinho ou simplesmente pode não jogar, se você tiver um parceiro humano; neste caso, o computador servirá apenas como tabuleiro e fornecedor de peças.

```
210 LET P=B(G(1))+B(G(2))+B(G(
3))+B(G(4))
220 LET X=F: IF B<B(32) THEN
LET P=P-BX: LET X=33-F
230 LET P=P*B(X): RETURN
250 LET F=FN A(ABS P)-30: LET
B=P/B(F): IF B<0 THEN  LET B=B
+BX: LET F=33-F
260 FOR A=1 TO 4: LET G(A)=FN
A(B)+1: LET B=B-B(G(A)): NEXT
A: RETURN
```

```
210 P=B(G(1))+B(G(2))+B(G(3))+B
(G(4))
220 X=F:IF P<B(31) THEN P=P-BX:
X=31-F
230 P=P*B(X):RETURN
250 F=FNA(ABS(P))-31:B=P/B(F):I
F B<0 THEN B=B+BX:F=31-F
260 FOR A=1 TO 4:G(A)=FNA(B):B=
B-B(G(A)):NEXT:RETURN
```

```
210 P=B(G(1))+B(G(2))+B(G(3))+B
(G(4))
220 X=F:IF P<B(31) THEN P=P-BX:
X=31-F
230 P=P*B(X):RETURN
250 F=FNA(ABS(P))-31:B=P/B(F):I
F B<0 THEN B=B+BX:F=31-F
260 FOR A=1 TO 4:G(A)=FNA(B):B=
B-B(G(A)):NEXT:RETURN
```

```
210 P = B(G(1)) + B(G(2)) + B(G
(3)) + B(G(4))
220 X = F: IF P < B(31) THEN P
= P - BX:X = 31 - F
230 P = P * B(X): RETURN
250 F =  FN A( ABS (P)) - 31:B
= P / B(F): IF B < 0 THEN B = B
 + BX:F = 31 - F
260  FOR A = 1 TO 4:G(A) =  FN
A(B):B = B - B(G(A)): NEXT : RE
TURN
```

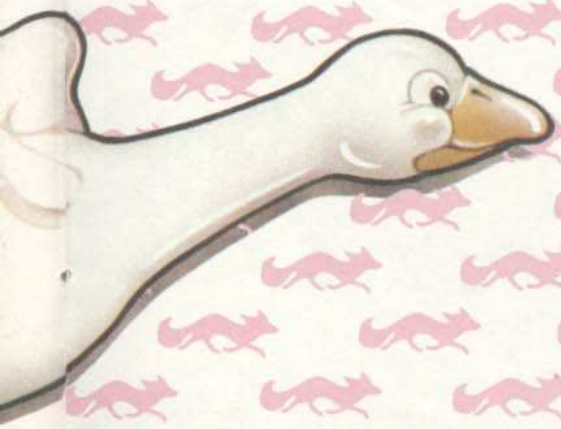
Essas são duas das mais importantes rotinas do programa. A rotina da linha 210 à linha 230 avalia a posição em jogo e a transforma em um número. De modo inverso, quando o computador tiver escolhido a melhor jogada, tem que converter um número em um movimento. As linhas 250 a 260 transformam o valor numérico em um conjunto de valores necessários para posicionar as peças. Essas sub-rotinas são chamadas com muita freqüência e, por isso, foram colocadas logo no início do programa.

O MOVIMENTO DA RAPOSA

```
1010 GOSUB 210
1020 GOSUB 310: GOSUB 250
1030 IF F>28 THEN  PRINT AT 21,
0;"A RAPOSA VENCEU
    ": GOTO 1410
1032 GOSUB 410: IF V=H THEN  PR
INT "OS GANSOS VENCERAM
    ": GOTO 1410
1040 IF PF THEN  GOTO 1110
1050 INPUT "MOVER RAPOSA PARA "
;B: IF B=-1 THEN  GOSUB 2710: G
OTO 1030
1055 GOSUB 4000
1060 FOR A=1 TO X(F): LET X=M(A
,F): IF X=B THEN  IF NOT (FN X(
X)) THEN  LET F=B: GOSUB 210: G
OTO 1200
1070 NEXT A: PRINT AT 21,0;"ISS
O NAO E LEGAL                ":
 GOTO 1050
1110 LET L=SF: LET M=SF: LET V(
M)=E*M: IF M>4 THEN  DIM R(HF+3
): DIM S(HF+3)
1112 GOSUB 1120: GOTO 1200
```

- COMO JOGAR NOS NÍVEIS MAIS BAIXOS
- A ROTINA DE MOVIMENTAÇÃO DA RAPOSA
- COMO MOVER OS GANSOS
- COMO JOGAR NOS NÍVEIS MAIS ALTOS
- O ALGORITMO ALFA-BETA
- TABELAS DE POSIÇÕES PARA UM JOGO MAIS RÁPIDO

```
1120 IF L=1 THEN  GOTO 410
1122 IF L<M-2 THEN  GOSUB 1610:
 IF V<>0 THEN  RETURN
1130 LET L=L-1: LET V(L)=H*L: L
ET A(L)=X(F): LET F(L)=F
1140 LET F=M(A(L),F(L))
1150 IF FN X(F)=0 THEN  GOSUB 1
320: IF V>V(L) THEN  LET V(L)=V
: LET P(L)=F: IF V>V(L+1) THEN
 LET F=F(L): LET L=L+1: RETURN
1160 LET A(L)=A(L)-1: IF A(L)>0
 THEN  GOTO 1140
1170 LET V=V(L): LET F=F(L): LE
T L=L+1: IF L=M THEN  LET F=P(M
-1): GOSUB 210: RETURN
1172 IF L<M-2 THEN  GOSUB 1510:
 RETURN
1180 RETURN
```



```
1010 SCREEN 1,0:GOSUB 210:GOTO
1030
1020 XX=FNXX(G):YY=FNYY(G):PUT(
XX,YY)-(XX+19,YY+19),SQ,PSET:XX
=FNXX(G(C)):YY=FNYY(G(C))+5:PUT
(XX,YY)-(XX+19,YY+9),GS,PSET
1030 CLS:IF F>27 THEN PLAY V$:P
RINT @7,"A RAPOSA VENCEU":GOTO
1410
1032 GOSUB 410:IF V=H THEN PLAY
 V$:PRINT @6,"OS GANSOS VENCERA
M":GOTO 1410
1040 LINE(180,0)-(255,191),PRES
ET,BF:IF PF THEN DRAW"BM180,50C
4"+TH$:GOTO 1110
1050 DRAW"BM180,80C4"+MW$:XX=FN
XX(F):YY=FNYY(F):GOSUB 1810:B=4
*INT(YY/20):B=FNCN(B)
1055 IF B=-1 THEN PLAY V$:PRINT
@6,"OS GANSOS VENCERAM":GOTO 14
10
1060 FOR A=0 TO X(F):X=M(A,F):I
F X=B AND NOT FNX(X) THEN A=X(F
):XX=FNXX(F):YY=FNYY(F):PUT(XX,
YY)-(XX+19,YY+19),SQ,PSET:F=B:G
OSUB 210:NEXT:GOTO 1200
1070 NEXT:GOSUB 5000:GOTO 1040
1110 L=SF:M=SF:V(M)=E*M:IF M>4
THEN FOR A=0 TO HF:R(A)=0:NEXT
1112 LF=F:GOSUB 1120:XX=FNXX(LF
):YY=FNYY(LF):PUT(XX,YY)-(XX+19
,YY+19),SQ,PSET:GOTO 1200
1120 IF L=1 THEN 410
1122 IF L<M-2 GOSUB 1610:IF V<>
0 THEN RETURN
1130 L=L-1:V(L)=H*L:A(L)=X(F):F
(L)=F
1140 F=M(A(L),F(L))
1150 IF FNX(F)=0 GOSUB 1320:IF
V>V(L) THEN V(L)=V:P(L)=F:IF V>
V(L+1) THEN F=F(L):L=L+1:RETURN
1160 A(L)=A(L)-1:IF A(L)>=0 THE
N 1140
1170 V=V(L):F=F(L):L=L+1:IF L=M
 THEN F=P(M-1):GOSUB 210:RETURN
1172 IF L<M-2 GOSUB 1510
1180 RETURN
```

```
1010 GOSUB 210:GOTO 1030
1020 GS(FNZZ(G),1)=G(C):XX=FNXX
(G(C)):YY=FNYY(G(C)):PUT SPRITE
 GS(FNZZ(G(C)),0),(XX,YY),15
1030 IF F>27 THEN PLAYV$:FORK=1
TO500:NEXT:SCREEN0:PRINT"A RAPO
SA VENCEU!":GOTO 1410
1032 GOSUB 410:IF V=H THEN PLAY
V$:FORK=1TO500:NEXT:SCREEN0:PRI
NT"OS GANSOS VENCERAM!":GOTO 14
10
1040 LINE(188,0)-(255,191),11,B
F:IF PF THEN PRESET (188,50):PR
INT#1,"Pensando":GOTO 1110
1050 PRESET (188,80):PRINT#1,"P
ara ?":XX=FNXX(F):YY=FNYY(F):GO
SUB 1810:B=4*INT((YY-2)/20):B=F
NCN(B)
1055 IF B=-1 THEN PLAYV$:FORK=1
TO500:NEXT:SCREEN0:PRINT"OS GAN
SOS VENCERAM!":GOTO 1410
1060 FOR A=0 TO X(F):X=M(A,F):I
F X=B AND NOT FNX(X) THEN A=X(F
):F=B:GOSUB 210:NEXT:GOTO 1200
1070 NEXT:GOSUB 5000:GOTO 1040
1110 L=SF:M=SF:V(M)=E*M:IF M>4
THEN FOR A=0 TO HF:R(A)=0:NEXT
1112 LF=F:GOSUB 1120:GOTO 1200
1120 IF L=1 THEN 410
1122 IF L<M-2 THEN GOSUB 1610:I
F V<>0 THEN RETURN
1130 L=L-1:V(L)=H*L:A(L)=X(F):F
(L)=F
1140 F=M(A(L),F(L))
1150 IF FNX(F)=0 THEN GOSUB 132
0:IF V>V(L) THEN V(L)=V:P(L)=F:
IF V>V(L+1) THEN F=F(L):L=L+1:R
ETURN
1160 A(L)=A(L)-1:IF A(L)>=0 THE
N 1140
1170 V=V(L):F=F(L):L=L+1:IF L=M
 THEN F=P(M-1):GOSUB 210:RETURN
1172 IF L<M-2 THEN GOSUB 1510
1180 RETURN
```

```
1010  GOSUB 210: GOTO 1030
1020 XX =  FN XX(G):YY =  FN YY
(G): HCOLOR= 0: DRAW GS AT XX +
 7,YY + 5: HCOLOR= 3:XX =  FN X
X(G(C)):YY =  FN YY(G(C)): DRAW
 GS AT XX + 7,YY + 5
1030  HOME : IF F > 27 THEN  PR
INT V$: HOME : VTAB 22: PRINT "
A RAPOSA GANHOU!": GOTO 1410
1032  GOSUB 410: IF V = H THEN
 PRINT V$: HOME : VTAB 22: PRIN
T "OS GANSOS GANHARAM!": GOTO 1
410
1040  HOME : VTAB 22: IF PF THE
N  PRINT "PENSANDO...";: GOTO 1
110
1050  PRINT "MOVE PARA ONDE? ";
:XX =  FN XX(F):YY =  FN YY(F):
 GOSUB 1810:B = 4 *  INT (YY /
20):B =  FN CN(B)
1055  IF B =  - 1 THEN  PRINT V
$: HOME : VTAB 22: PRINT "OS GA
NSOS VENCERAM!": GOTO 1410
1060  FOR A = 0 TO X(F):X = M(A
,F): IF X = B AND  NOT  FN X(X)
 THEN A = X(F):XX =  FN XX(F):Y
Y =  FN YY(F): HCOLOR= 0: DRAW
FX AT XX + 2,YY + 8:F = B: HCOL
OR= 3: GOSUB 210: NEXT : GOTO 1
200
1070  NEXT : GOSUB 5000: GOTO 1
040
1110 L = SF:M = SF:V(M) = E * M
: IF M > 4 THEN  FOR A = 0 TO H
F:R(A) = 0: NEXT
1112 LF = F: GOSUB 1120:XX =  F
N XX(LF):YY =  FN YY(LF): HCOLO
R= 0: DRAW FX AT XX + 2,YY + 8:
 HCOLOR= 3: GOTO 1200
1120  IF L = 1 THEN 410
1122  IF L < M - 2 THEN  GOSUB
1610: IF V <  > 0 THEN  RETURN

1130 L = L - 1:V(L) = H * L:A(L
) = X(F):F(L) = F
1140 F = M(A(L),F(L))
1150  IF  FN X(F) = 0 THEN  GOS
UB 1320: IF V > V(L) THEN V(L)
= V:P(L) = F: IF V > V(L + 1) T
HEN F = F(L):L = L + 1: RETURN
1160 A(L) = A(L) - 1: IF A(L) >
  = 0 THEN 1140
1170 V = V(L):F = F(L):L = L +
1: IF L = M THEN F = P(M - 1):
GOSUB 210: RETURN
1172  IF L < M - 2 THEN  GOSUB
1510
1180  RETURN
```

As linhas 1020 a 1180 manipulam os movimentos da raposa. A sub-rotina que desenha o tabuleiro é usada para mostrar o posicionamento atual das peças. Em seguida, verifica-se na linha 1030 se a raposa ganhou. Feito isso, a rotina comprova na linha 1032 se há pelo menos um movimento legal para a raposa (caso contrário a vitória caberá aos gansos).

Se o jogador estiver controlando a raposa, as linhas 1050 a 1070 receberão a jogada e verificarão sua validade. Se, ao contrário, quem estiver controlando a raposa for o computador, as linhas 1110 a 1112 decidirão pelo movimento. O número de lances que está sendo analisado, **M**, e o lance que é analisado no momento, **L**, são definidos na linha 1110. As linhas 1120 a 1180 avaliam qual a melhor jogada.

Quatro matrizes são usadas nessa rotina: **A** contém o número de movimentos a serem analisados; **V**, o melhor resultado até o momento; **P**, a jogada que provoca tal resultado, e **F** indica a posição prévia da raposa.

COMO MOVER OS GANSOS

```
500 LET V=0: FOR C=1 TO 4: LET
G=G(C): FOR A=1 TO Z(G): LET X
=M(A,G): IF X<>F THEN  IF NOT
FN X(X) THEN  RETURN
502 NEXT A: NEXT C: LET V=1:
RETURN
1200 GOSUB 310: GOSUB 250
1202 IF F>28 THEN  PRINT AT 21,
0;"A RAPOSA VENCEU
    ": GOTO 1410
1204 GOSUB 500: IF V THEN  PRIN
T AT 21,0;"A RAPOSA VENCEU
          ": GOTO 1410
1210 IF PG THEN  GOTO 1310
1220 INPUT "QUE GANSO QUER MOVE
R ?";G: IF G=-1 THEN  GOSUB 271
0: GOTO 1202
1225 GOSUB 4000
1230 LET C=FN Z(G): IF C=0 THEN
  PRINT AT 21,0;"NAO HA GANSO E
M ";G;"": GOTO 1220
1240 INPUT "PARA ONDE ";I: IF I
=-1 THEN  GOSUB 2710: GOTO 1202
1250 IF FN X(I) THEN  PRINT AT
21,0;"AI JA TEM UM GANSO !
      ": GOTO 1220
1260 IF I=F THEN  PRINT AT 21,0
;"AI JA TEM UMA RAPOSA
```

```
   ": GOTO 1220
1270 FOR A=1 TO Z(G): IF M(A,G)
=I THEN  LET G(C)=I: GOTO 1010
1280 NEXT A: PRINT AT 21,0;"ISS
O E ILEGAL                    "
 GOTO 1220
1310 LET L=SG: LET M=SG: LET V(
M)=H*M: IF M>4 THEN  DIM R(HF+3
): DIM S(HF+3)
1312 GOSUB 1320: GOTO 1020
1320 IF L=1 THEN  GOTO 510
1322 IF L<M-2 THEN  GOSUB 1610:
 IF V<>0 THEN  RETURN
1324 LET L=L-1: LET V(L)=E*L: L
ET C=1
1330 LET C(L)=C: LET F(L)=G(C):
 LET A(L)=1: IF A(L)>Z(G(C)) TH
EN  GOTO 1362
1340 LET B=M(A(L),F(L)): LET X=
FN X(B): LET G(C)=B: IF X OR B=
F THEN  GOTO 1360
1350 GOSUB 1120: LET C=C(L): IF
 V<V(L) THEN  LET V(L)=V: LET P
(L)=G(C)+C*32
1355 IF V<V(L) THEN  LET G(C)=F
```

```
(L): LET L=L+1: RETURN
1360 LET A(L)=A(L)+1: IF A(L)<=
Z(F(L)) THEN  GOTO 1340
1362 LET G(C)=F(L): LET C=C+1:
IF C<5 THEN  GOTO 1330
1370 LET V=V(L): LET L=L+1: IF
L=M THEN  LET C=INT (P(L-1)/32)
: LET G(C)=P(L-1)-C*32: GOSUB 2
10: RETURN
1372 IF L<M-2 THEN  GOSUB 1510:
 RETURN
1380 RETURN
```

```
500 V=-1:FOR C=1 TO 4:G=G(C):IF
 Z(G)<0 THEN NEXT:RETURN
502 FOR A=0 TO Z(G):X=M(A,G):V=
V AND(FNX(X) OR X=F):NEXT A,C:R
ETURN
1200 GOSUB 250:XX=FNXX(F):YY=FN
YY(F)+5:PUT(XX,YY)-(XX+19,YY+8)
,FX,PSET
1202 IF F>27 THEN PLAY V$:PRINT
 @6,"A RAPOSA VENCEU":GOTO 1410
1204 GOSUB 500:IF V THEN PLAY V
$:PRINT "A RAPOSA VENCEU":GOTO
1410
1210 LINE(180,0)-(255,191),PRES
ET,BF:IF PG THEN DRAW"BM180,50C
4"+TH$:GOTO 1310
1220 XX=FNXX(G(1)):YY=FNYY(G(1)
):DRAW"BM180,80C2"+WG$:GOSUB 18
10
1225 G=4*INT(YY/20):G=FNCN(G)
1230 C=FNZ(G):IF C=0 GOSUB 5000
:GOTO 1210
1240 DRAW"BM180,110C3"+MW$:GOSU
B 1810:I=4*INT(YY/20):I=FNCN(I)
1245 IF I=1 THEN PLAY V$:PRINT
@7,"A RAPOSA VENCEU":GOTO 1410
1250 IF FNX(I) GOSUB 5000:GOTO
1210
1260 IF I=F GOSUB 5000:GOTO 121
0
1270 FOR A=-1 TO Z(G):IF A>=0 T
HEN IF M(A,G)=I THEN XX=FNXX(G(
C)):YY=FNYY(G(C)):PUT(XX,YY)-(X
```

```
X+19,YY+19),SQ,PSET:G(C)=I:XX=F
NXX(I):YY=FNYY(I):PUT(XX,YY+5)-
(XX+19,YY+14),GS,PSET:A=Z(G):NE
XT:GOTO 1010
1280 NEXT:GOSUB 5000:GOTO 1210
1310 L=SG:M=SG:V(M)=H*M:IF M>4
THEN FOR A=0 TO HF:R(A)=0:NEXT
1312 GOSUB 1320:GOTO 1020
1320 IF L=1 THEN 510
1322 IF L<M-2 GOSUB 1610:IF V<>
0 THEN RETURN
1324 L=L-1:V(L)=E*L:C=1
1330 C(L)=C:F(L)=G(C):A(L)=0:IF
 A(L)>Z(G(C)) THEN 1362
1340 B=M(A(L),F(L)):X=FNX(B):G=
G(C):G(C)=B:IF X OR B=F GOTO 13
60
1350 GOSUB 1120:C=C(L):IF V<V(L
) THEN V(L)=V:P(L)=G(C)+C*32:IF
 V<V(L+1) THEN G=G(C):G(C)=F(L)
:L=L+1:RETURN
1360 A(L)=A(L)+1:IF A(L)<=Z(F(L
)) THEN 1340
1362 G=G(C):G(C)=F(L):C=C+1:IF
C<5 THEN 1330
1370 V=V(L):L=L+1:IF L=M THEN C
=INT(P(L-1)/32):G=G(C):G(C)=P(L
-1) AND 31:GOSUB 210:RETURN
1372 IF L<M-2 GOSUB 1510
1380 RETURN
```



```
500 V=-1:FOR C=1 TO 4:G=G(C):IF
 Z(G)<0 THEN NEXT:RETURN
502 FOR A=0 TO Z(G):X=M(A,G):V=
V AND (FNX(X) OR X=F):NEXT:NEXT
:RETURN
1200 GOSUB 250:XX=FNXX(F):YY=FN
YY(F):PUT SPRITE FX,(XX,YY),6
1202 IF F>27 THEN PLAYV$:FORK=1
TO500:NEXT:SCREEN0:PRINT"A RAPO
SA VENCEU!":GOTO 1410
1204 GOSUB 500:IF V THEN PLAYV$
:FORK=1TO500:NEXT:SCREEN0:PRINT
"A RAPOSA VENCEU!":GOTO 1410
1210 LINE (188,0)-(255,191),11,
BF:IF PG THEN PRESET (188,50):P
RINT#1,"Pensando":GOTO 1310
1220 XX=FNXX(G(1)):YY=FNYY(G(1)
):PRESET (188,80):PRINT#1,"Qual
 ?":GOSUB 1810
1225 G=4*INT((YY-2)/20):G=FNCN(
G)
1230 C=FNZ(G):IF C=0 THEN GOSUB
 5000:GOTO 1210
1240 LINE (188,0)-(255,191),11,
BF:PRESET (188,80):PRINT#1,"Par
a ?":GOSUB 1810:I=4*INT((YY-2)/
20):I=FNCN(I)
1245 IF I=1 THEN PLAYV$:FORK=1T
O500:NEXT:SCREEN0:PRINT"A RAPOS
A VENCEU!":GOTO 1410
1250 IF FNX(I) THEN GOSUB 5000:
GOTO 1210
1260 IF I=F THEN GOSUB 5000:GOT
O 1210
1265 GS(FNZZ(G),1)=I
1270 FOR A=-1 TO Z(G):IF A>=0 T
HEN IF M(A,G)=I THEN XX=FNXX(I)
:YY=FNYY(I):PUT SPRITE GS(FNZZ(
I),0),(XX,YY),15:A=Z(G):G(C)=I:
NEXT:GOTO 1010
1280 NEXT:GOSUB 5000:GOTO 1210
1310 L=SG:M=SG:V(M)=H*M:IF M>4
THEN FOR A=0 TO HF:R(A)=0:NEXT
1312 GOSUB 1320:GOTO 1020
1320 IF L=1 THEN 510
1322 IF L<M-2 THEN GOSUB 1610:I
F V<>0 THEN RETURN
1324 L=L-1:V(L)=E*L:C=1
1330 C(L)=C:F(L)=G(C):A(L)=0:IF
 A(L)>Z(G(C)) THEN 1362
1340 B=M(A(L),F(L)):X=FNX(B):G=
G(C):G(C)=B:IF X OR B=F THEN 13
60
1350 GOSUB 1120:C=C(L):IF V<V(L
) THEN V(L)=V:P(L)=G(C)+C*32:IF
 V<V(L+1) THEN G=G(C):G(C)=F(L)
:L=L+1:RETURN
1360 A(L)=A(L)+1:IF A(L)<=Z(F(L
)) THEN 1340
1362 G=G(C):G(C)=F(L):C=C+1:IF
C<5 THEN 1330
1370 V=V(L):L=L+1:IF L=M THEN C
=INT(P(L-1)/32):G=G(C):G(C)=P(L
-1)AND31:GOSUB 210:RETURN
1372 IF L<M-2 THEN GOSUB 1510
1380 RETURN
```

```
500 V = 1: FOR C = 1 TO 4:G = G
(C): IF G(C) < 0 THEN  NEXT : R
ETURN
502  FOR A = 0 TO Z(G):X = M(A,
G):V = V AND ( FN X(X) OR X = F
): NEXT : NEXT : RETURN


1200  GOSUB 250:XX =  FN XX(F):
YY =  FN YY(F): DRAW FX AT XX +
 2,YY + 8
1202  IF F > 27 THEN  PRINT V$:
 HOME : VTAB 22: PRINT "A RAPOS
A VENCEU!": GOTO 1410
1204  GOSUB 500: IF V THEN  PRI
NT V$: HOME : VTAB 22: PRINT "A
 RAPOSA VENCEU!": GOTO 1410
1210  HOME : VTAB 22: IF PG THE
N  PRINT "PENSANDO...";: GOTO 1
310
1220 XX =  FN XX(G(1)):YY =  FN
 YY(G(1)): PRINT "MOVE QUAL GAN
SO?": GOSUB 1810
1225 G = 4 *  INT (YY / 20):G =
  FN CN(G)
1230 C =  FN Z(G): IF C = 0 THE
N  GOSUB 5000: GOTO 1210
1240  HOME : VTAB 22: PRINT "MO
VE PARA ONDE? ": GOSUB 1810:I =
 4 *  INT (YY / 20):I =  FN CN(
I)
1245  IF I = 1 THEN  PRINT V$:
HOME : VTAB 22: PRINT "A RAPOSA
 VENCEU!": GOTO 1410
1250  IF  FN X(I) THEN  GOSUB 5
```

```
000: GOTO 1210
1270  FOR A =  - 1 TO Z(G): IF
A >  = 0 THEN  IF M(A,G) = I TH
EN XX =  FN XX(G(C)):YY =  FN Y
Y(G(C)): HCOLOR= 0: DRAW GS AT
XX + 7,YY + 5:G(C) = I:XX =  FN
 XX(I):YY =  FN YY(I): HCOLOR=
3: DRAW GS AT XX + 7,YY + 5:A =
 Z(G): NEXT : GOTO 1010
1280  NEXT : GOSUB 5000: GOTO 1
210
1310 L = SG:M = SG:V(M) = H * M
: IF M > 4 THEN  FOR A = 0 TO H
F:R(A) = 0: NEXT
1312  GOSUB 1320: GOTO 1020
1320  IF L = 1 THEN 510
1322  IF L < M - 2 THEN  GOSUB
1610: IF V <  > 0 THEN  RETURN
1324 L = L - 1:V(L) = E * L:C =
 1
1330 C(L) = C:F(L) = G(C):A(L)
= 0: IF A(L) > Z(G(C)) THEN 136
2
1340 B = M(A(L),F(L)):X =  FN X
(B):G = G(C):G(C) = B: IF X OR
B = F THEN 1360
1350  GOSUB 1120:C = C(L): IF V
 < V(L) THEN V(L) = V:P(L) = G(
C) + C * 32: IF V < V(L + 1) TH
EN G = G(C):G(C) = F(L):L = L +
 1: RETURN
1360 A(L) = A(L) + 1: IF A(L) <
  = Z(F(L)) THEN 1340
1362 G = G(C):G(C) = F(L):C = C
 + 1: IF C < 5 THEN 1330
1370 V = V(L):L = L + 1: IF L =
 M THEN C =  INT (P(L - 1) / 32
):G = G(C):MD = 32:G(C) =  FN M
D(P(L - 1)): GOSUB 210: RETURN

1372  IF L < M - 2 THEN  GOSUB
1510
1380  RETURN
```

A rotina que vai da linha 1200 à linha 1380 é responsável pela manipulação dos gansos. Uma parte dela — linhas 1220 a 1290 — é utilizada quando o jogador controla os gansos. Quando esse controle cabe ao microcomputador, as linhas 1320 a 1380 encarregam-se da manipulação.

MELHORES JOGADAS

```
410 LET V=H: FOR A=X(F) TO 1
STEP -1: LET X=M(A,F): IF FN X
(X) THEN  NEXT A: LET L=1:
RETURN
420 LET B=F: LET F=X: GOSUB
210: LET V=P: LET F=B: LET L=1
: RETURN
510 LET V=E: FOR C=1 TO 4: LET
G=G(C): IF -B(G)>V THEN  GOTO
 530
520 FOR A=1 TO Z(G): LET X=M(A
,G): IF FN X(X) OR (X=F) THEN
NEXT A: GOTO 530
528 LET V=B(X)-B(G): LET D=C:
LET B=X
530 NEXT C: LET G=G(D): LET G(
D)=B: GOSUB 210: LET V=P: LET
G(D)=G: RETURN
```

```
410 V=H:FOR A=X(F)TO 0 STEP -1:
X=M(A,F):IF FNX(X)<0 THEN NEXT:
L=1:RETURN
420 B=F:F=X:GOSUB 210:V=P:F=B:L
=1:A=0:NEXT:RETURN
510 V=E:FOR C=1 TO 4:G=G(C):IF
-B(G)>V THEN 530
520 FOR A=0 TO Z(G):X=M(A,G):IF
 FNX(X) OR X=F THEN NEXT:GOTO 5
30
528 V=B(X)-B(G):D=C:B=X
530 NEXT:G=G(D):G(D)=B:GOSUB 21
0:V=P:G(D)=G:IF SG=1 THEN G(D)=
B:C=D
540 RETURN
```

```
410 V=H:FOR A=X(F) TO 0 STEP-1:
X=M(A,F):IF FNX(X)<0 THEN NEXT:
L=1:RETURN
420 B=F:F=X:GOSUB 210:V=P:F=B:L
=1:A=0:NEXT:RETURN
```

```
520 FOR A=0 TO Z(G):X=M(A,G):IF
 FNX(X) OR X=F THEN NEXT:GOTO 5
30
528 V=B(X)-B(G):D=C:B=X
530 NEXT:G=G(D):G(D)=B:GOSUB 21
0:V=P:G(D)=G:IF SG=1 THEN G(D)=
B:C=D
540 RETURN
```

```
410 V = H: FOR A = X(F) TO 0 ST
EP  - 1:X = M(A,F): IF  FN X(X)
 < 0 THEN  NEXT :L = 1: RETURN
```

```
420 B = F:F = X: GOSUB 210:V =
P:F = B:L = 1:A = 0: NEXT : RET
URN


510 V = E: FOR C = 1 TO 4:G = G
(C): IF  - B(G) > V THEN 530
520  FOR A = 0 TO Z(G):X = M(A,
G): IF  FN X(X) OR X = F THEN
NEXT : GOTO 530
528 V = B(X) - B(G):D = C:B = X

530  NEXT :G = G(D):G(D) = B: G
OSUB 210:V = P:G(D) = G: IF SG
= 1 THEN G(D) = B:C = D:
540  RETURN
```

Essas sub-rotinas são usadas apenas quando o computador está nos menores níveis de dificuldade e analisam apenas o próximo movimento.

As linhas 410 e 420 analisam todos os movimentos possíveis para a raposa, usando o mapa da matriz **M**, montada a partir da linha 2110. A sub-rotina devolve um valor P (configuração após a melhor jogada) e um valor V (avaliação do melhor resultado).

As linhas 510 e 530 funcionam de maneira similar às anteriores, só que se encarregam dos gansos. As matrizes P e V são determinadas da forma já descrita. Nos dois casos, a instrução **GOSUB 210** escolhe o melhor movimento entre os que são avaliados.

A TABELA DE POSIÇÕES


```
1510 GOSUB 210: LET C=P
1520 LET C=C-INT ((C/HF+C)-C)*H
F: IF C<0 OR C>=HF THEN  GOTO 1
520
1550 FOR A=C+1 TO C+4: IF R(A)<
>0 AND R(A)<>P THEN  NEXT A: RE
TURN
1560 LET R(A)=P: LET S(A)=V: RE
TURN
1610 GOSUB 210: LET C=P
1620 LET C=C-INT ((C/HF+C)-C)*H
F: IF C<0 OR C>=HF THEN  GOTO 1
620
1650 FOR A=C+1 TO C+4: IF R(A)<
>0 AND R(A)<>P THEN  NEXT A: LE
T V=0: RETURN
1660 LET V=S(A)*(R(A)=P): RETUR
N
```

```
1510 GOSUB 210:C=P
1520 C=C-INT((C/HF+C)-C)*HF:IF
C<0 OR C>=HF THEN 1520
1550 FOR A=C TO C+C:IF R(A)<>0
AND R(A)<>P THEN NEXT:RETURN
1560 R(A)=P:S(A)=V:A=C+C:NEXT:R
ETURN
1610 GOSUB 210:C=P
1620 C=C-INT((C/HF+C)-C)*HF:IF
C<0 OR C>=HF THEN 1620
1650 FOR A=C TO C+C:IF R(A)<>0
AND R(A)<>P THEN NEXT:V=0:RETUR
N
```

```
1660 V=-S(A)*(R(A)=P):A=C+C:NEX
T:RETURN
1810 SCREEN 1,0:PUT(XX,YY)-(XX+
19,YY+19),SQ,NOT:FOR Z=1 TO 100
:NEXT
1820 PUT(XX,YY)-(XX+19,YY+19),S
Q,NOT
1830 K$=INKEY$:IF K$="^" AND YY
>8 AND XX>8 THEN YY=YY-20:XX=XX
-20:GOTO 1810
1840 IF K$=CHR$(10) AND YY<129
AND XX<129 THEN YY=YY+20:XX=XX+
20:GOTO 1810
1850 IF K$=CHR$(8) AND XX>28 TH
EN XX=XX-40:GOTO 1810
1860 IF K$=CHR$(9) AND XX<128 T
HEN XX=XX+40:GOTO 1810
1870 IF K$=CHR$(13) THEN RETURN
1875 IF K$="Q" THEN YY=0:XX=-12
:RETURN
1880 GOTO 1810
```



```
'1510 GOSUB 210:C=P
1520 C=C-INT((C/HF+C)-C)*HF:IF
C<0 OR C>=HF THEN 1520
1550 FOR A=C TO C+C:IF R(A)<>0
AND R(A)<>P THEN NEXT:RETURN
1560 R(A)=P:S(A)=V:A=C+C:NEXT:R
ETURN
1610 GOSUB 210:C=P
1620 C=C-INT((C/HF+C)-C)*HF:IF
C<0 OR C>=HF THEN 1620
1650 FOR A=C TO C+C:IF R(A)<>0
```

```
AND R(A)<>P THEN NEXT:V=0:RETUR
N
1660 V=-S(A)*(R(A)=P):A=C+C:NEX
T:RETURN
1810 LINE (XX,YY-2)-(XX+19,YY+1
7),14,BF:FOR Z=1 TO 100:NEXT
1820 LINE (XX,YY-2)-(XX+19,YY+1
7),3,BF:FOR Z=1 TO 200:NEXT
1830 K$=INKEY$:IF K$=CHR$(30) A
ND YY>18 AND XX>18 THEN YY=YY-2
0:XX=XX-20:GOTO 1810
1840 IF K$=CHR$(31) AND YY<141
AND XX<141 THEN YY=YY+20:XX=XX+
20:GOTO 1810
1850 IF K$=CHR$(29) AND XX>38 T
HEN XX=XX-40:GOTO 1810
1860 IF K$=CHR$(28) AND XX<138
THEN XX=XX+40:GOTO 1810
1870 IF K$=CHR$(13) THEN RETURN
1880 GOTO 1810
```

```
1510  GOSUB 210:C = P
1520 C = C -  INT ((C / HF + C)
 - C) * HF: IF C < 0 OR C >  =
HF THEN 1520
1550  FOR A = C TO C + C: IF R(
A) <  > 0 AND R(A) <  > P THEN
 NEXT : RETURN
1560 R(A) = P:S(A) = V:A = C +
C: NEXT : RETURN
1610  GOSUB 210:C = P
1620 C = C -  INT ((C / HF + C)
 - C) * HF: IF C < 0 OR C >  =
HF THEN 1620
1650  FOR A = C TO C + C: IF R(
A) <  > 0 AND R(A) <  > P THEN
 NEXT :V = 0: RETURN
1660 V = S(A) * (R(A) = P):A =
C + C: NEXT : RETURN
1810 X0 = XX:Y0 = YY
1820  HCOLOR= 0: GOSUB 1890: HC
OLOR= 3:X0 = XX:Y0 = YY: GOSUB
1890
1830  GET K$: IF K$ = "A" AND Y
Y > Y1 + 5 AND XX > X1 + 5 THEN
 YY = YY - 20:XX = XX - 20: GOT
O 1820
1840  IF K$ = "Z" AND YY < Y2 -
 38 AND XX < X2 - 38 THEN YY =
YY + 20:XX = XX + 20: GOTO 1820

1850  IF K$ =  CHR$ (8) AND XX
> X1 + 21 THEN XX = XX - 40: GO
TO 1820
1860  IF K$ =  CHR$ (21) AND XX
 < X2 - 45 THEN XX = XX + 40: G
OTO 1820
1870  IF K$ =  CHR$ (13) THEN
HCOLOR= 0: GOSUB 1890: RETURN
1880  GOTO 1820
1890  HPLOT X0 + 1,Y0 + 1 TO X0
 + 19,Y0 + 1 TO X0 + 19,Y0 + 18
 TO X0 + 1,Y0 + 18 TO X0 + 1,Y0
 + 1: RETURN
```

Nos níveis de maior dificuldade será necessário usar o algoritmo alfa-beta — veja o primeiro artigo da série (página 872). Na verdade, ele já foi digitado como parte das rotinas de movimentação da raposa e dos gansos.

Antes de utilizá-lo, porém, o programa verifica se ele é necessário no momento — será o nível de dificuldade suficientemente alto para justificar o seu emprego?

A rotina das linhas 1110 a 1150 cuida da raposa, enquanto a rotina das linhas 1310 a 1350 dedica-se ao quarteto de gansos.

O algoritmo é aplicado quando se executa o último teste **IF**, no fim das linhas 1150 e 1350, depois de **V(M)** ter sido adequadamente definida, de acordo com o nível de dificuldade.

O algoritmo alfa-beta é mais eficiente quando o computador discrimina quais são os melhores movimentos realizados inicialmente — para os gansos, o quadrado de número mais alto de cada fileira de quatro; para a raposa, o quadrado aberto a ela mais próximo do lado dos gansos.

O algoritmo alfa-beta é usado em conjunto com uma tabela construída em função de movimentos que já foram considerados. Assim, o computador pode se decidir mais rapidamente, quando se trata de uma jogada já estudada. Quanto maior for o tamanho da tabela que é possível montar no computador, mais rápido será o processamento.

Inicializada nas linhas 2500, 2750 e 2800, a tabela tem seu tamanho ideal definido por meio de valores teóricos. Ela foi dimensionada, no entanto, no limite da memória RAM disponível em cada microcomputador.

A tabela é zerada nas linhas 1110 e 1310; o conteúdo é verificado nas linhas 1122 e 1322 e definido nas linhas 1172 e 1372.

O TRS-Color e o MSX usam um cursor para indicar as jogadas a serem executadas com as setas. O cursor do Apple é movido para a esquerda e para a direita com as setas, e para cima e para baixo com as letras A e Z.

A ARANHA MARCIANA (1)

- O "ENREDO"
- DEFINIÇÃO DOS GRÁFICOS
- FREDDY, AS FLECHAS, OS BALÕES E A ARANHA
- INICIALIZAÇÃO DO JOGO

Uma enorme e assustadora aranha está pronta a atacar Freddy. Sem uma boa dose de precisão e sangue-frio, ele não escapará. Mas não se assuste: tudo não passa de um terrível pesadelo.

O jogo *A Aranha Marciana* será montado em dois artigos. Neste, os desenhos são definidos e o programa inicializado. No próximo, o jogo passa a funcionar, com a adição das últimas rotinas.

O JOGO

O ponto inicial para a montagem de um jogo é a criação de algum tipo de enredo, ou mesmo de um personagem em torno do qual a ação se desenvolva.

Neste jogo, nosso personagem é Freddy, um limpador de janelas que tem um medo doentio de aranhas. Ele procurou vários especialistas que tentaram, em vão, curá-lo da fobia. O problema chegou a uma tal gravidade que Freddy passou a ter sempre o mesmo pesadelo. Nele aparecem uma aranha marciana — particularmente grande, faminta e de aparência horrível —, uma coleção de balões, um arco e uma flecha.

Com freqüência, ele acorda banhado em suor, após ter sonhado que estava em sua escada, tentando desesperadamente flechar balões que, se atingissem a gaiola da aranha, acima de sua cabeça, destravariam a porta que aprisiona o monstro. Ajude Freddy a estourar os balões, ou ele terá um trágico fim, como almoço de uma horrenda aranha!

Temos, assim, um enredo. Com relação aos pontos, eles serão dados para cada balão estourado. Mas, mesmo que Freddy consiga interceptar todos os balões, seu problema não estará resolvido: a tortura continuará num nível de dificuldade ainda maior com balões mais rápidos. Se deixar que três deles passem, a aranha marciana sairá da gaiola.

O PROGRAMA

Nosso jogo requer a animação de quatro figuras: a aranha, que se movimenta na vertical e na horizontal; Freddy, que se move só na vertical; o balão, que pode aparecer em qualquer

ponto da parte inferior da tela, mas que também se move na vertical, e a flecha. Esta normalmente se move na horizontal, mas acompanha o movimento vertical de Freddy.

As figuras de maior interesse são a aranha e o balão. Muitas variáveis estarão associadas a elas, e será útil armazená-las em uma matriz unidimensional e usar uma constante como referência.

Além do movimento, precisamos definir o desenho de todas as diferentes figuras — animadas ou não — que aparecerão na tela. Como nos melhores jogos de computador, elas devem ser coloridas. Freddy será montado em um bloco gráfico de três por dois caracteres. Usaremos outros blocos de dois por dois caracteres para definir a aranha, um balão inteiro e um balão estourado, dois caracteres para o desenho da flecha e mais dois para o da escada. Assim, teremos no final cerca de 26 caracteres.

STM

A primeira parte do programa define os gráficos e inicializa o jogo.

DEFINIÇÃO DOS GRÁFICOS

S

```
1000 DIM b(6): DIM s(7)
1010 LET xpos=1: LET ypos=2: LE
T colour=3: LET points=4: LET c
```

```
ount=5: LET maxcount=6
1020 LET xinc=3: LET yinc=4: LE
T picture=7
1030 LET dest=65288
1040 FOR i=0 TO 26*8-1: READ j:
 POKE dest+i,j: NEXT i
1050 DATA 15,63,127,255,255,255
,255,127
1060 DATA 240,252,254,255,255,2
55,255,254
1070 DATA 127,63,63,31,15,7,3,6
1080 DATA 254,252,252,248,240,2
24,192,96
1090 DATA 32,96,255,255,96,32,0
,0
1100 DATA 5,10,252,252,10,5,0,0
1110 DATA 1,64,17,40,16,0,0,161
1120 DATA 128,2,136,20,8,0,0,13
3
1130 DATA 161,0,0,16,50,17,64,1
1140 DATA 133,0,0,8,20,136,2,12
8
1150 DATA 48,48,48,48,111,111,4
8,48
1160 DATA 12,12,12,12,246,246,1
2,12
1170 DATA 7,31,49,57,127,112,23
7,255
1180 DATA 224,248,140,204,254,1
4,183,255
1190 DATA 127,59,51,99,115,35,6
,12
1200 DATA 254,108,102,51,49,25,
24,48
1210 DATA 7,31,49,51,127,112,23
5,255
1220 DATA 224,248,140,156,254,1
4,215,255
1230 DATA 127,51,50,27,25,50,11
2,224
1240 DATA 254,204,108,102,54,22
,3,6
1250 DATA 15,31,19,55,55,63,63,
15
1260 DATA 240,248,248,252,252,2
52,252,240
1270 DATA 251,219,139,219,219,2
51,247,239
1280 DATA 252,254,254,254,254,2
54,254,252
1290 DATA 95,127,31,31,31,63,12
7,127
1300 DATA 188,188,188,188,188,1
24,252,248
1310 LET hiscore=0
1320 RETURN
```

```
1000 DIM B(6),S(7),NU$(9)
1010 XP=1:YP=2:PO=4:CT=5:MC=6
1020 XI=3:YI=4:PI=7
1022 DIM AD(14),EF(4),E(4),GJ(7
),KL(4),MP(7),QT(7),UZ(14),SP(7
),S1(4),S2(7),BL(4)
1023 PMODE 4,1:PCLS1:SG=PEEK(18
8)*256
1024 GOSUB 1500
1025 GET(0,0)-(15,23),AD,G:PCLS
1
1026 GOSUB 1520
1027 GET(0,0)-(15,7),EF,G:GET(0,
,0)-(7,7),E,G:PCLS1
1028 GOSUB 1500
1029 GET(0,0)-(15,15),GJ,G:PCLS
1
1030 GOSUB 1520
1031 GET(0,0)-(15,7),KL,G:PCLS1
1032 GOSUB 1500
1033 GET(0,0)-(15,15),MP,G:PCLS
1
1034 GOSUB 1500
1035 GET(0,0)-(15,15),QT,G:PCLS
1
1036 GOSUB 1500:SG=SG+512:GOSUB
 1520:SG=SG-512
1037 GET(0,0)-(15,23),UZ,G:PCLS
1
1038 GET(0,0)-(15,15),SP,G
1039 GET(0,0)-(7,7),S1,G
1040 GET(0,0)-(15,7),S2,G
1041 PCLS0:GET(0,0)-(7,7),BL,G
1050 DATA 15,63,127,255,255,255
,255,127
1060 DATA 240,252,254,255,255,2
55,255,254
1070 DATA 127,63,63,31,15,7,3,6
1080 DATA 254,252,252,248,240,2
24,192,96
1090 DATA 32,96,255,255,96,32,0
,0
1100 DATA 5,10,252,252,10,5,0,0
1110 DATA 1,64,17,40,16,0,0,161
1120 DATA 128,2,136,20,8,0,0,13
3
1130 DATA 161,0,0,16,40,17,64,1
1140 DATA 133,0,0,8,20,136,2,12
8
1150 DATA 48,48,48,48,111,111,4
8,48
1160 DATA 12,12,12,12,246,246,1
2,12
1170 DATA 7,31,49,57,127,112,23
7,255
1180 DATA 224,248,140,204,254,1
4,183,255
1190 DATA 127,59,51,99,115,35,6
,12
1200 DATA 254,108,102,51,49,25,
24,48
1210 DATA 7,31,49,51,127,112,23
5,255
1220 DATA 224,248,140,156,254,1
4,215,255
1230 DATA 127,51,50,27,25,50,11
2,224
1240 DATA 254,204,108,102,54,22
,3,6
1250 DATA 15,31,19,55,55,63,63,
15
1260 DATA 240,248,248,252,252,2
52,252,240
1270 DATA 251,219,139,219,219,2
51,247,239
1280 DATA 252,254,254,254,254,2
54,254,252
1290 DATA 95,127,31,31,31,63,12
7,127
1300 DATA 188,188,188,188,188,1
24,252,248
1310 HS=0
1320 RETURN
1500 FOR CL=0 TO 1:FOR CH=0 TO
1:FOR L=0 TO 7:READ J:POKE SG+C
L*256+CH+L*32,255-J:NEXT L,CH,C
L
1510 RETURN
1520 FOR CH=0 TO 1:FOR L=0 TO 7
:READ J:POKE SG+CH+L*32,255-J:N
EXT L,CH
1530 RETURN
1600 DATA R6D8L6U8BR8,BR6ND8BR2
,R6D4L6D4R6BR2BU8,R6D4NL3D4NL6B
R2BU8,D4R6D4U8BR2,NR6D4R6D4L6BE
8
1610 DATA D8R6U4L6U4BR8,R6ND8BR
2,R6D8L6U8D4R6U4BR2,D4R6D4U8L6B
R8
1620 FOR I=0 TO 9:READ NU$(I):N
EXT
1625 DRAW"C1;S2"
1630 RETURN
1650 N$=STR$(NU):FOR I2=2 TO LE
N(N$)
1660 DI=ASC(MID$(N$,I2,1))-48:D
RAW NU$(DI)+"BR2":NEXT I2:RETUR
N
1700 COLOR 0:LINE (178,2)-(200,
7),PSET,BF:NU=HS:DRAW"C1;BM178,
2":GOSUB 1650:RETURN
```

```
1000 DIM B(6),S(7):SCREEN2,2
1005 A1=1:A2=2:BA=3:BE=4:FD=5:F
L=6
1010 XP=1:YP=2:PO=4:CT=5:MC=6
1020 XI=3:YI=4:PI=7
1030 FOR I=1 TO 6:A$=""
1040 FOR J=1 TO 32
1050 READ A:A$=A$+CHR$(A)
1060 SPRITE$(I)=A$
1070 NEXT:NEXT
1100 DATA 0,0,0,0,7,13,31,18,15
,10,10,18,10,2,4,0,0,0,0,0,208,
176,248,72,240,80,80,72,80,64,1
28,0
1110 DATA  0,0,0,0,7,13,31,18,1
5,10,10,9,8,8,4,0,0,0,0,0,208,1
76,248,72,240,80,80,144,16,16,8
,0
1120 DATA  0,0,0,3,15,31,31,31,
15,15,7,3,3,1,0,0,0,0,0,192,240
,248,248,248,240,240,224,192,19
2,128,0,0
1130 DATA  0,0,0,0,8,4,0,13,1,0
,4,4,8,0,0,0,0,0,0,16,32,0,0,12
8,152,0,64,32,0,0,0,0
1140 DATA  0,3,7,7,3,1,7,15,15,
15,15,7,3,2,2,1,0,192,224,224,1
92,128,224,240,240,240,240,224,
192,192,192,128
1150 DATA  0,0,0,0,0,0,16,112,2
55,112,16,0,0,0,0,0,0,0,0,0,0,0
,8,16,240,16,8,0,0,0,0,0
1310 HS=0
1320 RETURN
```

Esta parte do programa lê os dados das linhas **DATA** para montar os blocos gráficos (UDG ou sprites) do balão e da aranha nas matrizes **B** (ou **b**) e **S** (ou **s**), que são dimensionadas na linha 1000.

As linhas 1010 e 1020 definem os valores iniciais para os apontadores das

matrizes, antes dos UDG ou sprites serem montados. As linhas 1030 a 1040 (até 1070, no MSX) fazem a leitura dos dados em **DATA** e montam os blocos gráficos. Finalmente, a linha 1310 define o recorde como 0. Essa linha é executada apenas uma vez. O programa para o TRS-Color inclui uma rotina a mais, que começa na linha 1600. Sua função é desenhar números na tela de alta resolução.

INICIALIZAÇÃO DO JOGO

```
3000 LET score=0: LET level=1
3010 LET my=15
3020 LET bl=15+5*level: LET ax=
29: LET ay=16: LET dead=0: LET
props=3
3090 GOSUB 5000
3150 PAPER 0: BORDER 0: CLS
3160 FOR x=0 TO 28: PRINT AT 3,
x; INK 0; PAPER 6;" ";AT 0,x;"
": NEXT x: GOSUB 6000
3170 POKE 23607,60: PRINT AT 0,
0; INK 0; PAPER 6;"N=";level;"
 B=";bl;"  ";"SC=";score;AT 0,2
0;"RE=";hiscore
3180 POKE 23607,252
3190 FOR y=5 TO 21: PRINT AT y,
30; INK 6;"kl": NEXT y
3200 POKE 23607,252
3210 GOSUB 4000
3220 GOSUB 4200
3240 RETURN
```

```
3000 SC=0:LV=1
3010 MY=15
3020 BL=15+5*LV:AX=29:AY=16:DD=
0:PP=3
3090 GOSUB 5000
3150 PMODE 4,1:COLOR 0,1:PCLS:S
CREEN 1,1
3160 FOR X=0 TO 28:PUT(X*8,24)-
(X*8+7,31),BL,PSET:PUT(X*8,0)-(
X*8+7,7),BL,PSET:NEXT X:GOSUB 6
000
3165 DRAW"S4;BM2,2;C1;D4R3BR2BU
1R2BU1L2;BM48,2;D4R4U2L4R3U2L3B
R6BD2R2BD1L2;BM100,2;L4D2R4D2L4
BR6NR4U4R4;BM160,2;D4U2R4D2U4BR
2R4L2D4L2R4;S2"
3170 DRAW"C1;BM14,2;":NU=LV:GOS
UB 1650
3171 DRAW"BM58,2;":NU=BL:GOSUB
1650
3172 DRAW"BM114,2;":NU=SC:GOSUB
 1650
3173 DRAW"BM178,2;":NU=HS:GOSUB
 1650
3190 FOR Y=5 TO 21:PUT(240,Y*8)
-(255,Y*8+7),KL,PSET:NEXT Y
3210 GOSUB 4000
3220 GOSUB 4200
3240 RETURN
```

```
3000 SC=0:LV=1
3010 MY=16
3020 BL=15+5*LV:AX=29:AY=16:DD=
0:PP=3
3090 GOSUB 5000
3160 LINE (0,0)-(231,31),6,BF:L
INE (0,7)-(80,24),14,BF:LINE (8
7,7)-(152,24),14,BF:LINE (159,7
)-(224,24),14,BF
3170 GOSUB 1700
3180 GOSUB 6000
3190 LINE (240,31)-(240,190),1:
LINE (255,31)-(255,190),1:FOR I
=2 TO 15:LINE (240,I*16)-(255,I
*16),1:NEXT
3210 GOSUB 4000
3220 GOSUB 4200
3230 RETURN
```

As linhas 3000, 3010 e 3020 zeram o placar, inicializam o nível de dificuldade em 1 e definem uma série de variáveis auxiliares. A linha 3150 determina as cores da tela (menos no MSX) e as linhas 3160 a 3170 exibem o placar e outras informações necessárias.

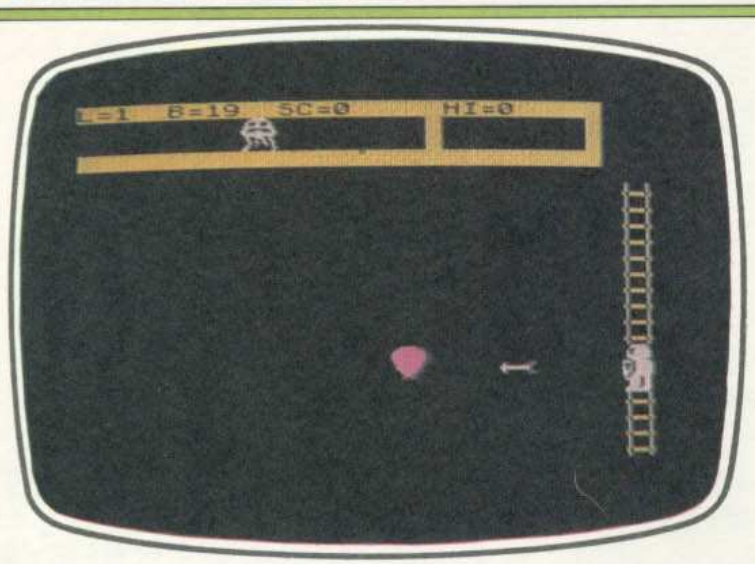
O programa do Spectrum inclui um **POKE** na linha 3170 e outro na 3180. A posição de memória 23607 guarda o apontador do conjunto de caracteres. Normalmente, ela tem o valor 60, que aponta para o conjunto de caracteres da ROM. Nesse programa, entretanto, o valor colocado no endereço 23607 para indicar a posição dos caracteres é 252. Isso permite ao computador usar as letras minúsculas, além das maiúsculas e dos números, como gráficos.

O programa reserva uma área da memória usando **CLEAR** e ali coloca os UDG. Devido ao seu efeito sobre as sub-rotinas, esse comando deve ser utilizado no programa principal, o que faremos na segunda parte do jogo. Portanto, se você quiser executar as linhas aqui apresentadas, digite antes **CLEAR 65287**.

A linha 3190 se incumbe de desenhar a escada, e as sub-rotinas chamadas nas linhas 4000 e 4200 desenham, respectivamente, Freddy e a aranha.

FREDDY E A FLECHA

```
4000 INK 7: PRINT AT my,30;"uv"
;AT my+1,30;"wx";AT my+2,30;"yz
": IF ax=29 THEN  PRINT AT ay,a
x;"e";
4010 RETURN
4110 INK 7: PRINT AT ay,ax;"ef"
: RETURN
```



As figuras montadas pelo programa tal como aparecem na tela dos microcomputadores da linha Spectrum.

```
4000 PUT(240,MY*8)-(255,MY*8+23
),UZ,PSET:IF AX=29 THEN PUT(AX*
8,AY*8)-(AX*8+7,AY*8+7),E,PSET
4010 RETURN
4110 PUT(AX*8,AY*8)-(AX*8+15,AY
*8+7),EF,PSET:RETURN
```

```
4000 PUT SPRITE 2,(240,MY*8),1,
FD:IF AX=29 THEN PUT SPRITE 3,(
AX*8,AY*8),4,FL
4010 RETURN
4110 PUT SPRITE 3,(AX*8,AY*8),4
,FL·RETURN
```

Essa rotina desenha Freddy sobre a escada e sua flecha. A posição do nosso personagem é determinada pelo valor de **MY**. A flecha, por sua vez, é desenhada pela linha 4110 na posição da tela indicada pelos valores de **AX** e **AY**.

A ARANHA MARCIANA

```
4200 IF s(picture)=1 THEN  GOTO
 4250
4210 PRINT AT s(ypos),s(xpos);"
mn";AT s(ypos)+1,s(xpos);"op":
RETURN
4250 PRINT AT s(ypos),s(xpos);"
qr";AT s(ypos)+1,s(xpos);"st":
RETURN
```

```
4200 X2=S(XP)*8:Y2=S(YP)*8:IF S
(PI)=1 THEN 4250
4210 PUT(X2,Y2)-(X2+15,Y2+15),M
P,PSET:RETURN
4250 PUT(X2,Y2)-(X2+15,Y2+15),Q
T,PSET:RETURN
```

```
4200 X2=S(XP)*8:Y2=S(YP)*8:IF S
(PI)=1 THEN 4250
4210 PUT SPRITE 1,(X2,Y2),12,A1
:RETURN
4250 PUT SPRITE 1,(X2,Y2),12,A2
:RETURN
```

A rotina que desenha a aranha se assemelha à anterior, exceto num detalhe: duas imagens são usadas para essa figura. Colocadas alternadamente no mesmo lugar da tela, pelas linhas 4210 e 4250, simulam o movimento das pernas da aranha.

SOBEM OS BALÕES

```
4300 PRINT AT b(ypos),b(xpos);
BRIGHT 1; INK b(colour);"ab";AT
 b(ypos)+1,b(xpos);"cd": RETURN

5000 LET range=INT (RND*6)
5010 LET b(xpos)=(4*range)+INT
(RND*4)
5020 LET b(ypos)=20
5030 LET b(maxcount)=5-level
5040 LET b(count)=1
5050 LET b(colour)=INT (RND*5)+
3
5060 LET b(points)=10-range
5070 RETURN
```

```
4300 X2=B(XP)*8:Y2=B(YP)*8:PUT(
X2,Y2)-(X2+15,Y2+23),AD,PSET:RE
TURN
5000 RG=RND(6)-1
5010 B(XP)=(4*RG)+RND(4)-1
5020 B(YP)=20
5030 B(MC)=5-LV
5040 B(CT)=1
5060 B(PO)=10-RG
5070 RETURN
```

```
4300 X2=B(XP)*8:Y2=B(YP)*8:PUT
SPRITE 4,(X2,Y2),9,BA:RETURN
5000 RG=INT(RND(1)*6)
5010 B(XP)=(4*RG)+INT(RND(1)*5)
5020 B(YP)=20
5030 B(MC)=5-LV
5040 B(CT)=1
5050 B(PO)=10-RG
5070 RETURN
```

A linha 4300 simplesmente desenha os balões na tela. As linhas restantes encarregam-se de inflar um balão toda vez que um outro estourar ou alcançar a gaiola da aranha (os balões podem aparecer em qualquer um dos seis pontos da parte inferior da tela, flutuando em seguida para cima, na direção vertical. A variável **MAXCOUNT** (ou, ainda, **maxcount** ou **MC**, conforme o computador) determina o quanto o balão se moveu, ou seja, o nível que ele atingiu. Em seguida, o programa define a cor do balão e seu valor em pontos. Este dependerá de sua proximidade da escada de Freddy.

AS PORTAS

```
6000 IF level<>1 THEN  POKE 236
07,60: PRINT AT s(ypos),s(xpos)
;"  ";AT s(ypos)+1,s(xpos);"  "
: POKE 23607,252
6010 FOR x=10 TO 30 STEP 9
6020 PRINT  INK 6;AT 1,x;"";AT
2,x;""
6030 NEXT x
6040 LET s(xpos)=1: LET s(ypos)
=1: LET s(xinc)=1: LET s(yinc)=
0: LET s(count)=4: LET s(maxcou
nt)=4: LET s(picture)=1
6050 RETURN
```

```
6000 IF LV<>1 THEN X2=S(XP)*8:Y
2=S(YP)*8:PUT(X2,Y2)-(X2+15,Y2+
15),SP,PSET
6010 FOR X=10 TO 30 STEP 9
6020 PUT(X*8,8)-(X*8+7,15),BL,P
SET:PUT(X*8,16)-(X*8+7,23),BL,P
SET
6030 NEXT X
6040 S(XP)=1:S(YP)=1:S(XI)=1:S(.
```

```
YI)=0:S(CT)=4:S(MC)=4:S(PI)=1
6050 RETURN
```

```
6000 S(XP)=1:S(YP)=1:S(XI)=1:S(
YI)=0:S(CT)=4:S(MC)=4:S(PI)=1
6010 RETURN
```

Três portas fecham a gaiola da aranha, para mantê-la presa. Os usuários do Spectrum notarão que o caractere listado como ? é, de fato, um quadradinho gráfico, obtido com a tecla 8 em modo gráfico.

A primeira seção do programa define os gráficos e inicializa o jogo.

A INICIALIZAÇÃO

```
1000  DIM B(6),S(7),NU$(9): SCA
LE= 1: ROT= 0: HCOLOR= 3
1010 XP = 1:YP = 2:PO = 4:CT =
5:MC = 6
1020 XI = 3:YI = 4:PI = 7
1025 A1 = 1:A2 = 2:BA = 3:BE =
4:FD = 5:FL = 6
1030  DATA   6,0,14,0,75,0,136,
0,1,1,21,1,158,1
1040  DATA   45,45,45,45,77,58,
63,63,63,63,63,63,55,173,240
1050  DATA  43,45,45,44,45,53,4
5,37,45,62,54,63,63,63,63,63,63
,188
1060  DATA  10,62,62,62,54,109,
145,37,37,36,36,100,73,17,54,54

1070  DATA  54,53,109,193,193,1
93,45,36,39,39,36,0
1080  DATA   45,45,45,45,117,57
,63,63,63,63,63,63,55,45,222
1090  DATA  45,45,37,45,45,46,4
5,44,53,55,62,63,63,63,63,63,39
,151
1100  DATA  49,54,62,62,62,79,7
3,72,1,193,193,39,39,108,73
1110  DATA  9,54,55,183,74,73,7
3,39,39,39,36,36,0
1120  DATA   45,45,45,53,63,63,
63,63,55,45,45,45,45,45,44,54
1130  DATA  63,63,63,63,63,63,5
5,45,45,45,45,45,45,45,44,54,63

1140  DATA  63,63,63,63,63,63,5
5,45,45,45,45,45,45,45,53,63
1150  DATA  63,63,63,63,63,63,5
5,45,45,45,45,45,45,45,53,63,63
,63
1160  DATA   63,63,63,63,46,45,
45,45,45,45,45,62,63,63,63,63,6
,3
1170  DATA  55,45,45,45,45,45,5
3,63,63,63,63,63,46,45,45,45,45
,62
1180  DATA  63,63,63,55,45,45,4
5,53,63,63,55,62,87,73,73,8,39,
4,0
1190  DATA  46,110,9,44,172,146
,57,223,219,59,191,146,45,37,77
,9,53,46,45,0
1200  DATA   45,45,45,53,63,63,
63,63,46,45,45,45,45,44,54,63
1210  DATA  63,63,63,63,46,45,4
5,45,45,62,63,63,63,55,45,45
1220  DATA  45,45,222,63,63,62,
63,63,46,45,45,37,45,45,45,45,5
3,63,63
1230  DATA  63,63,62,63,63,63,4
6,45,45,45,37,45,45,45,53,63
1240  DATA   63,63,62,63,63,63
,63,46,45,45,45,45,45,44,45,53
1250  DATA  63,55,62,63,63,63,6
3,60,54,45,45,45,45,45,62,63,63

1255  DATA  63,63,46,45,45,45,5
3,63,63,63,63,46,45,45,45,62,63
,63,55
1260  DATA  45,45,45,62,63,63,4
6,45,45,62,63,63,55,45,45,45,5,
0
1270  DATA   45,44,54,119,33,36
,44,54,54,37,36,172,42,45,45,45
,45,5,0
1280  FOR I = 16384 TO 16816: R
EAD C: POKE I,C: NEXT
1290 P = 233: POKE P,64: POKE P
- 1,0
1300 HS = 0
1310  RETURN
```

Essa parte do programa lê os dados das linhas **DATA** para montar, nas matrizes **B** e **S** (dimensionadas na linha 1000), os blocos gráficos do balão e da aranha.

As linhas 1010 a 1025 determinam os valores iniciais para os apontadores das matrizes. As linhas 1280 a 1290 fazem a leitura dos dados em **DATA** (linhas 1030 a 1270) e a montagem dos blocos gráficos. Finalmente, a linha 1310 define o recorde como 0. Essa linha será executada apenas uma vez.

Adicione agora este programa:

```
3000 SC = 0:LV = 1
3010 MY = 15
3020 BL = 15 + 5 * LV:AX = 28:A
Y = 16:DD = 0:PP = 3
3090  GOSUB 5000
3150  HGR : GOSUB 1700
3160  FOR Y = 0 TO 3: HPLOT 0,Y
 TO 234,Y: HPLOT 0,Y + 23 TO 23
4,Y + 23: NEXT
3170  FOR X = 0 TO 3: HPLOT X +
 77,3 TO X + 77,23: HPLOT X + 1
54,3 TO X + 154,23: HPLOT X + 2
31,3 TO X + 231,23: NEXT
3175  GOSUB 6000
3180  HPLOT 235,25 TO 235,160:
HPLOT 255,25 TO 255,160
3210  GOSUB 4000
3220  GOSUB 4200
3240  RETURN
```

As linhas 3000, 3010 e 3020 zeram o placar, inicializam o nível de dificuldade em 1 e definem uma série de variáveis auxiliares necessárias ao jogo. As linhas 3160 a 3180 mostram o placar e outras informações. A linha 3190 desenha a escada, e as sub-rotinas chamadas nas linhas 4000 e 4200 desenham, respectivamente, Freddy e a aranha.

A rotina seguinte desenha Freddy na escada e sua flecha. A posição do nosso personagem é determinada pelo valor de **MY**. A flecha, por sua vez, é desenhada pela linha 4110 na posição da tela indicada pelos valores de **AX** e **AY**.

```
4000  DRAW FD AT 240,MY * 8: IF
 AX = 28 THEN  DRAW FL AT AX *
8,AY * 8
4010  RETURN
4110  DRAW FL AT AX * 8,AY * 8:
 RETURN
```

A ARANHA MARCIANA

```
4200 X2 = S(XP) * 8:Y2 = S(YP)
* 8: IF S(PI) = 1 THEN 4250
4210  DRAW A1 AT X2,Y2: RETURN

4250  DRAW A2 AT X2,Y2: RETURN
```

A rotina que desenha a aranha é muito parecida com a anterior, exceto por um detalhe: duas imagens são usadas para essa figura (**A1** e **A2**). Elas são colocadas alternadamente no mesmo lugar da tela, pelas linhas 4210 e 4250, simulando que as pernas da aranha se movem.

```
4300 X2 = B(XP) * 8:Y2 = (B(YP)
 + 1) * 8: HCOLOR= 0: DRAW BA A
T X2,Y2: HCOLOR= 3:Y2 = B(YP) *
 8: DRAW BA AT X2,Y2: RETURN
5000 RG =  INT ( RND (1) * 6)
5010 B(XP) = (4 * RG) +  INT (
RND (1) * 5)
5020 B(YP) = 20
5030 B(MC) = 5 - LV
5040 B(CT) = 1
5050 B(PO) = 10 - RG
5070  RETURN
```

A linha 4300 desenha os balões na tela. As linhas restantes encarregam-se de inflar um balão toda vez que um deles estourar ou alcançar a gaiola da aranha.

Os balões aparecem em qualquer um dos seis pontos da parte inferior da tela, flutuando em seguida verticalmente. A variável **MC** determina o nível que o balão atingiu. Por fim, o programa estabelece o valor, em pontos, do balão, segundo sua proximidade da escada.

A rotina final dessa parte do jogo é responsável pela contagem do número de portas (máximo de três) que estão mantendo a aranha dentro da gaiola.

```
6000  IF LV <  > 1 THEN X2 = S(
XP) * 8:Y2 = S(YP) * 8: DRAW AL
GUMACOISA
6040 S(XP) = 1:S(YP) = 1:S(XI)
= 1:S(YI) = 0:S(CT) = 4:S(MC) =
 4:S(PI) = 1
6050  RETURN
```

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 uso de rotinas em código de máquina dentro de programas BASIC. Os comandos **USR** e **DEFUSR**. Armazenagem de rotinas **USR**.

PERIFÉRICOS

Os diferentes tipos de tabletes gráficos. Digitalização e mapeamento. Geração de sinais. Sensores de superfície.

PROGRAMAÇÃO DE JOGOS

Programe uma colônia de bactérias em código de máquina.

CÓDIGO DE MÁQUINA

A rotina principal de *Avalanche*. Acertos iniciais.

PROGRAMAÇÃO DE JOGOS

Complete o jogo *A Aranha Marciana*, listando as rotinas de animação.