

Este livro de Tim Hartnell apresenta 20 jogos dinâmicos para o ZX Spectrum. Nele encontrará uma grande variedade de jogos, desde os de tabuleiro, como Xadrez e Pirandello, a jogos de arcada, como Peões e Corrida da Morte, incluindo um grande programa de aventuras, Vingança no Castelo do Terror.

Cada jogo é apresentado minuciosamente e, na maior parte dos casos, o programa é explicado linha por linha. São realçados os artifícios e técnicas usados pelos programadores e Tim Hartnell sugere como cada um os pode aplicar para realizar os seus próprios programas. Um capítulo final apresenta sugestões para melhorar e desenvolver programas.

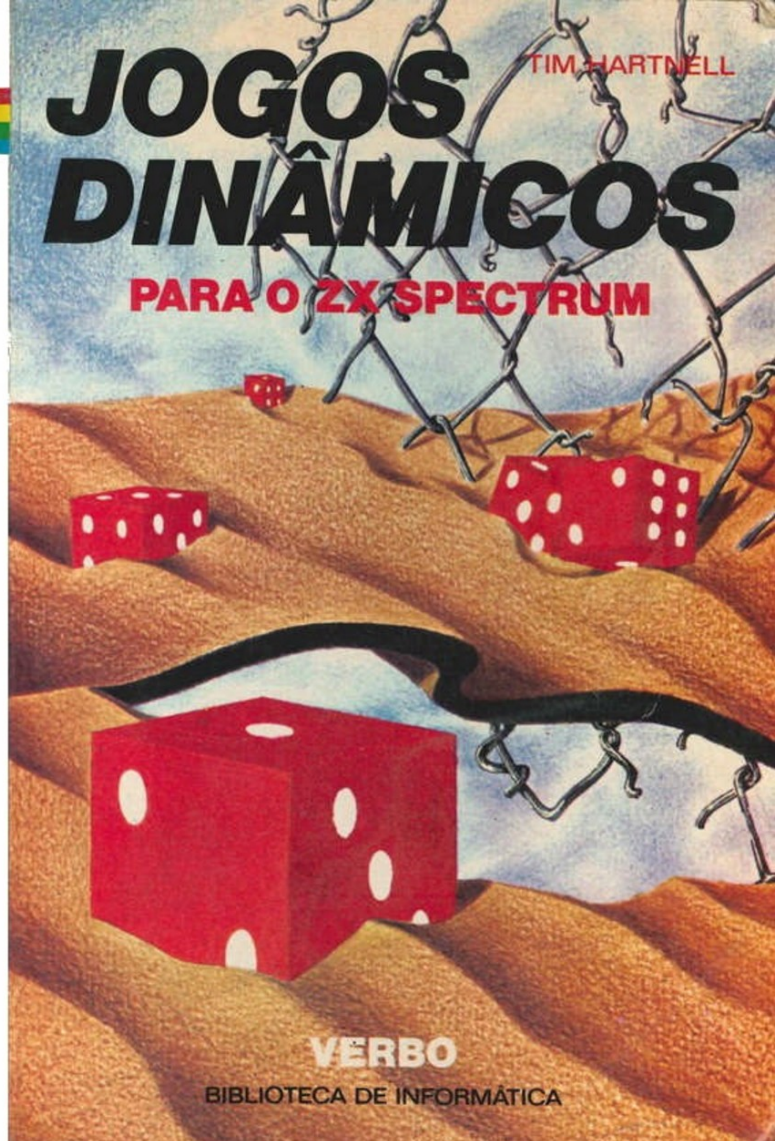
Todos os programas deste livro foram verificados e testados pelo Gabinete Verbo de Informática.



BIBLIOTECA VERBO DE INFORMÁTICA

1
JOGOS DINÂMICOS HARTNELL

VERBO



**Jogos
Dinâmicos
para o
ZX Spectrum**

LITEC

LIVRARIA EDITORA TÉCNICA LTDA

Rua dos Timbiras, 257 - CEP: 01208 - São Paulo
Caixa Postal 30869 - Tel: 222-0477

REF.

1734

PREÇO

49,00 -

TIM HARTNELL

Jogos Dinâmicos para o ZX Spectrum

Verbo

ÍNDICE

Prefácio	7
Jogos de arcada:	9
NÍVEL CINCO	9
CIRCO	16
PEÕES	27
DESPOLETAR A MINA	30
HELICÓPTERO	37
TIRO AO ALVO	41
BATALHA DAS LETRAS	48
DEMOLIÇÃO	53
SIMULADOR DE CONDUÇÃO A 3 DIMENSÕES	57
PÂNICO NA NORUEGA	60
CORRIDA DA MORTE	62
AS RÉPLICAS LOUCAS	66
Jogos de tabuleiro:	75
PIRANDELLO	75
DAMAS	82
JOGO DO GALO	97
XADREZ PARA PRINCIPIANTES	102
Aventura/simulações:	136
A VINGANÇA NO CASTELO DO TERROR	136
QUIOSQUE DE LIMONADAS	168
Aperfeiçoar a mente:	179
SINTAXE	179
O POÇO	183
Sugestões para programação:	188
COMO APERFEIÇOAR OS PROGRAMAS	188
Apêndices:	195
O GERADOR DE LETRA GÓTICA	195
RENUMERAÇÃO COM CÓDIGO MÁQUINA	207
AUTORES DOS PROGRAMAS	215

Título original: *Dynamic Games for the ZX Spectrum*
Tradução de Carlos Jorge Trindade e Silva Pinto
Capa de Luís Anglin
© Copyright by Tim Hartnell. 1984
Direitos reservados para a Língua Portuguesa
Editorial VERBO. Lisboa/São Paulo
N.º Ed. 1576
Composto por Fotocompográfica, Lda.
Impresso por Empresa Litográfica do Sul
— Vila Real de St.º António,
em Janeiro de 1985
Depósito legal n.º 7861/85

Prefácio

Sob vários aspectos, um computador é o adversário ideal no jogo.

Incapaz de se mostrar eufórico na vitória, é amável na derrota, infatigável, paciente e infinitamente acessível.

Embora não consiga fornecer aquele diálogo que faz tanto parte do divertimento como o próprio jogo, o computador ainda é o melhor companheiro para um bom passatempo.

Qualquer que tenha sido o motivo que levou à compra de um *Spectrum* decerto quem o fez já passou pelo menos certo tempo brincando com jogos; e é aqui que este livro entra em acção.

Encontra-se nele uma grande variedade de jogos, desde os jogos de tabuleiro, como damas e xadrez, a jogos de acção, como OS PEÕES ou NÍVEL CINCO, até ao nosso maior programa de aventuras: VINGANÇA NO CASTELO DO TERROR.

Apontámos para jogos que valessem a pena pelo seu interesse duradouro, bem como para aqueles capazes de oferecerem auxílio em ideias e sugestões a aplicar na construção de programas de computador.

Para ajudar o leitor a tirar o maior partido deste livro, apresentam-se instruções minuciosas de cada jogo.

Na maior parte dos casos a introdução conduz através do programa, linha por linha, explicando os truques que os programadores utilizaram e sugerindo como esses truques podem ser aplicados a novos programas e jogos.

Vários programas incluem também sugestões da forma de como se podem moldar às necessidades de cada qual, melhorando-os e desenvolvendo-os.

A maioria dos programas deste livro foram escritos por Tim Rogers, Paul Toland e o autor.

Tim e Paul são ambos estudantes (Tim em Londres e Paul em Belfast), ambos merecedores de felicitações pela qualidade e originalidade do seu trabalho, o que se reflecte nos programas que seleccionámos para este livro.

E pronto! É a altura de começar a introduzir instruções no *Spectrum*.

Vamos divertir-nos a valer com estes jogos, que merecem verdadeiramente o nome de «dinâmicos».

Tim Hartnell
Londres, Março 1983

Jogos de arcada

NÍVEL CINCO

Não vale a pena entrar em pânico, só porque vem aí o NÍVEL CINCO, uma movimentada adaptação de um dos jogos de arcada* mais apreciados.

A área de jogo é constituída por cinco níveis (andares) ligados entre si por um conjunto de escadas.

Quatro monstros partem do andar mais baixo e tomam o caminho que os levará ao topo. A tarefa do jogador é contê-los na prisão abaixo do nível térreo.

O principal problema é que, uma vez que estes monstros se movem sempre para cima, a única maneira de os reter no fundo é abrindo buracos nos vários andares para que eles aí caiam.

Pode-se abrir um buraco pressionando a tecla "Ø", desde que não haja mais nenhum buraco no *écran*.

Cada buraco durará apenas um curto período de tempo.

O seu movimento controla-se por meio das seguintes teclas: "5" (esquerda), "8" (direita), "6" (sobe) e "7" (desce).

O jogo termina depois de dominados todos os monstros.

Nessa altura, anuncia-se quanto tempo o jogo demorou.

O jogo termina prematuramente se o jogador for apanhado por um monstro.

Como em muitos dos programas do *Spectrum*, só nos aperceberemos da qualidade dos grafismos por uma amostra obtida numa impressora.

Este programa é bastante rápido e cativante.

O recorde do seu próprio autor, Paul Toland, é de 60 unidades de tempo.

Quem será capaz de batê-lo?

A linha 1 do programa mostra quais os gráficos e as teclas em que se encontram: tijolos, na tecla "A"; secções de escada, na

* Em vários países, os locais públicos onde se encontram máquinas de jogos são conhecidos por "arcades", que significa arcadas.

tecla "B"; a pessoa que corre (o jogador), na tecla "C"; e os atraentes monstros na tecla "E".

A linha 25 remete o comando para a sub-rotina localizada na linha 9100 após a INK (tinta ou cor) ter sido colocada na cor branca e o PAPER (fundo) e o BORDER (enquadramento) terem sido colocados em azul.

Nesta sub-rotina as instruções são impressas no *écran*. "O balanço" das instruções aparece então juntamente com uma mensagem de "BOA SORTE!", e a instrução de RETURN na linha 9160 manda o comando de volta para a linha 30, onde se chama a sub-rotina da linha 9000. A linha 9000 começa por um RESTORE (restabelecer) do ponteiro de dados a ela própria (o que tem o efeito de colocar o homem que corre no item de DATA que segue imediatamente o número de linha que acompanha a instrução RESTORE).

Assim se assegura que, quando os ciclos I e J correm (linhas 9001 a 9040) os números lidos para serem introduzidos nos gráficos (através de uma instrução POKE), sejam os das linhas 9050, 9060, 9070, 9080 e 9090 e não da linha 1070.

Uma vez definidos os gráficos, o programa é conduzido (através da segunda instrução da linha 30), à sub-rotina da linha 700, que imprime as cinco carreiras de tijolos, uma por cada andar.

A linha 910 faz um RESTORE a ela própria (do mesmo modo que a linha 9000 que vimos há pouco), deslocando o ponteiro de dados para a primeira instrução DATA que se seguir ao RESTORE. Neste caso é a linha 1030 que contém a informação usada nos ciclos J e I (1000 a 1050) para a colocação das escadas nos seus lugares. Em 1055 limpa-se a linha do fundo do *écran* e seguidamente remete-se o comando do programa à linha 40, onde vai começar a verdadeira diversão.

Na linha 40, colocam-se as coordenadas Y e X da figura nas posições 14 e 13, respectivamente. Na linha 50, a variável M, que conta os monstros, é inicializada a 4, e na segunda instrução desta linha cria-se uma matriz para os guardar antes de eles serem colocados na partida através do ciclo na linha 60.

O vector H (que guarda o buraco) é DIMENSIONADO na linha 70; na linha 80 o fundo e os caracteres são colocados a "8" (correspondente à "cor" transparente) até que na linha 90 se suprime a figura representando o homem que corre.

A variável TEMPO é inicializada a 0 na linha 100 (recorde-se o resultado de 60 de Paul Toland) e posteriormente incrementada em 395, onde o seu valor se imprime no *écran*.

O título do programa coloca-se no topo do *écran* (110), deixando um lugar para o TEMPO poder aparecer; finalmente na linha 190 a variável LIVRE é inicializada a 0.

O ciclo I vai verificar a posição e controlar a acção dos monstros.

Este ciclo corre entre as linhas 200 e 390 e tem contidas nele todas as acções importantes do programa. Estas incluem o rastreio das posições verticais dos monstros (linha 205), a sua reimpressão (210 e 220), a verificação do ambiente que os rodeia (linhas 225 e 230), e os saltos condicionados conforme o monstro lhe tenha caído em cima (salto para a linha 430) ou esteja apenas a trepar (salto para 280). A coordenada horizontal do monstro é incrementada na linha 240, e a linha seguinte inverte a direcção do movimento do monstro caso este tenha atingido algum dos extremos da carreira.

"Adeus, monstro", é o que se poderá dizer se o monstro cair num buraco (linha 270) antes de ser reimprimido na sua nova posição pela instrução da linha 280.

Um buraco dura 40 ciclos do programa, após os quais é novamente preenchido pela instrução da linha 295. A linha 300 lê o teclado e saltará sobre uma grande parte da rotina se entretanto não for pressionada alguma tecla.

Se na rotina se lê um zero, é porque se quer abrir um buraco.

Se H(3) é igual a zero, o computador sabe que não há mais buracos no *écran* e, por isso, colocará as coordenadas do buraco que se pretende abrir uma linha abaixo da posição da figura, antes de ele ser impresso e de o programa saltar para a linha 390.

O computador sabe que pressionando zero na linha 315 não se poderá estar a pressionar "5", "6", "7" ou "8" nas linhas mais próximas; por isso, mantém a velocidade máxima de

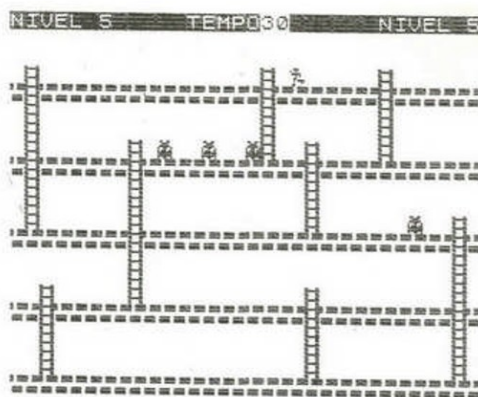
execução do programa saltando as linhas que ele sabe não ser necessário processar desta vez. Na linha 317 verifica-se se está a ser pressionada alguma tecla não compreendida entre "4" e "9"; se estiver, o comando salta para 390.

A próxima linha (320) coloca em posição a figura que corre, e a linha 325 faz o POKE do número 15 no endereço do ATTR atribuído à localização da figura humana, por forma a fazê-la desaparecer da posição actual antes de ser reimpressa na linha 390. Os ATTR (atributos) são armazenados em memória a seguir ao DISPLAY FILE, ou seja a partir da posição 22528 (ver mapas de memória do manual do ZX Spectrum).

Na linha 327 verifica-se a posição da figura. Se esta não estiver numa carreira, transfere-se o comando para a linha 350, onde se verifica se se está a subir ou a descer numa escada.

Na linha 370, a variável N, usada para fazer o POKE do atributo da posição do jogador, é colocada com o valor do seu ATTR presente antes de o «boneco» ser impresso na linha 380. Alcança-se a linha 330 se o homem que corre estiver numa carreira a mover-se de acordo com a vontade do jogador, expressa pelas teclas "5" ou "8", e modificada pela linha 340, que evita que caia dos "limites do mundo" sempre que chega a um dos extremos da carreira. A linha 390 recomeça todo o ciclo.

Depois de a figura que corre ter passado quatro vezes pelo ciclo 1, a variável LIVRE é verificada para averiguar se todos os monstros já foram capturados e, caso o não tenham sido, devolve-se o comando à linha 190, para que o jogo continue. A linha 410 assinala que todos os monstros foram capturados, e a 430 assinala que, pelo contrário, foram eles que apanharam a figura. Na linha 500 oferece-se a oportunidade de começar um novo jogo e, qualquer que seja a tecla pressionada, à excepção de "N" ou "n", permitirá reiniciá-lo. Na linha 1 estão assinalados os gráficos definidos pelo utilizador. Ao introduzir o programa pela primeira vez, sempre que se encontre um destes gráficos escreva-se a letra correspondente em modo "G" (gráfico), reportando-se para tal à referida linha 1. Exemplo: ao encontrar-se a figura que corre, como no fim da linha 90, deve escrever-se, no seu lugar, "C" em modo "G".



```

1 REM UDG "A"=A "B"=B "C"=C "E"=E
2 REM
10 REM NIVEL 5 P.TOLAND
20 INK 7: PAPER 1: BORDER 1: C
L5
25 GO SUB 9100
30 GO SUB 9000: GO SUB 700
40 LET X=14: LET Y=3: LET C=1
50 LET M=4: DIM M(M,3)
60 FOR I=1 TO M: LET M(I,1)=19
: LET M(I,2)=I*3: LET M(I,3)=1-I
NT (RND+.5)*2: PRINT AT 19,I*3;"
": NEXT I
70 DIM H(3): LET H(1)=0: LET H
(2)=0: LET H(3)=0
80 INK 8: PAPER 8
90 PRINT AT Y,X: INK 5;"A"
100 LET N=15: LET TEMPO=0
110 PRINT AT 0,0: INVERSE 1:"NI
VEL 5 TEMPO=
NIVEL 5"
190 LET LIVRE=0
200 FOR I=1 TO 4
205 IF M(I,1)=21 THEN GO TO 295
207 LET LIVRE=1
210 LET D=M(I,1): LET A=M(I,2)
220 PRINT AT D,A: OVER 1:"A"
225 IF ATTR (D-1,A)=13 THEN GO
TO 430
230 IF ATTR (D-1,A)=14 THEN LET

```



```

D=D-1: GO TO 280
240 LET A=A+M(I,3)
250 IF A=0 OR A=31 THEN LET M(I,3)=-M(I,3)
260 IF ATTR (D,A)=13 THEN GO TO 430
270 IF ATTR (D+1,A)=15 THEN LET D=D+4: BEEP .1,30: IF D>21 THEN LET D=21
280 PRINT AT D,A: OVER 1;"M"
290 LET M(I,1)=D: LET M(I,2)=A
295 IF H(3)>0 THEN LET H(3)=H(3)+1: IF H(3)=40 THEN PRINT INK 2; PAPER 7; AT H(1),H(2);"M": LET H(3)=0
300 LET I$=INKEY$
310 IF I$="" THEN GO TO 390
315 IF I$="0" AND H(3)=0 AND ATTR (Y+1,X)=58 THEN LET H(1)=Y+1: LET H(2)=X: LET H(3)=1: PRINT PAPER 1; INK 7; AT H(1),H(2);" ": GO TO 390
317 IF I$<"5" OR I$>"8" THEN GO TO 390
320 PRINT AT Y,X: OVER 1;"X"
325 POKE 22528+Y*32+X,N
327 IF (Y+1)/4<>INT ((Y+1)/4) THEN GO TO 350
330 LET X=X+(I$="8")-(I$="5")
340 LET X=X+(X<0)-(X>31)
350 IF I$="7" AND ATTR (Y-1,X)=14 THEN LET Y=Y-1
360 IF I$="6" AND ATTR (Y+1,X)=14 THEN LET Y=Y+1
370 LET N=ATTR (Y,X)
380 PRINT AT Y,X: OVER 1; INK 5;"X"
390 NEXT I
395 LET TEMPO=TEMPO+1: PRINT AT 0,17,TEMPO
400 IF LIVRE THEN GO TO 190
410 PRINT OVER 1; AT 9,5;"MONSTR OS TODOS CAPTURADOS": BEEP 3,30
420 GO TO 440
430 PRINT AT Y,X;"M": OVER 1; AT 9,5;"OS MONSTROS APANHARAM-NO": BEEP 3,-20
440 PRINT AT 13,10;"DEPOIS DO TEMPO DE ";TEMPO
500 INPUT "OUTRO JOGO ?";A$
510 IF A$<>"N" AND A$<>"n" THEN RUN

```

```

599 STOP
700 FOR I=1 TO 5
800 PRINT INK 2; PAPER 7; AT I*4,0;"*****"
800 NEXT I
910 RESTORE 910
1000 FOR J=1 TO 8
1010 READ X,Y,L
1020 FOR I=0 TO L
1030 PRINT INK 6; AT Y+I,X;"H"
1040 NEXT I
1050 NEXT J
1055 PRINT AT 21,0; PAPER 0,,
1060 RETURN
1070 DATA 1,3,8,25,3,4,8,7,8,20,7,4,30,11,8,2,15,4,17,3,4,20,15,4
8999 STOP
9000 RESTORE 9000
9001 FOR I=0 TO 4
9010 FOR J=0 TO 7
9020 READ N: POKE USR "A"+I*8+J,N
9030 NEXT J
9040 NEXT I
9045 RETURN
9050 DATA 207,207,207,0,0,126,126,126
9060 DATA 195,255,195,195,195,255,195,195
9070 DATA 55,144,124,20,48,47,34,96
9080 DATA 28,9,62,40,12,116,68,6
9090 DATA 102,60,126,153,187,255,255,153
9100 PRINT INVERSE 1;" NIVEL 5"
9110 PRINT "Existem quatro monstros movendo--se ao longo do ecran que tentam chegar ao cimo. Existem cinco andares ligados por escadas e tem de tentar aprisionar os monstros na masmorra abaixo do nivel ter-reo. O unico meio de colocar os monstros abaixo do ecran e abrir um buraco num dos andares pelo qual o monstro vai cair. Porem, apos algum tempo o buraco desapa-rece, nao podendo em caso algum existir mais de um buraco."

```

```

9120 PRINT "Morrera se algum mon
stro o atin-gir; mas a sua taref
a sera a de aprisionar os quatro
monstros no menor tempo possive
l."
"; FLASH 1;"Carregue
em qualquer tecla para continua
r"
9130 PAUSE 0:CLS
9140 PRINT "Use o 5,6,7 e 8 para
mover o seuboneco e 0 (zero) pa
ra abrir um buraco. BOA SORTE..
vai pre-cisar dela!!"
9150 PAUSE 200:CLS
9160 RETURN

```

CIRCO

CIRCO é um jogo de arcade semelhante a um outro conhecido por *Breakout* e por outros nomes, que consiste em destruir vários níveis de tijolos usando uma bola que salta entre eles e uma plataforma movida pelo jogador; mas este envolvendo um cenário bem mais interessante e um maior desafio ao jogador.

A finalidade é rebentar os balões que se encontram no topo do *écran*, atirando para cima, com as teclas "5" e "8", os palhaços que estão no balancé.

Pressionando a tecla "1" provoca-se a troca de lugares no balancé relativamente ao palhaço que lá se encontra. O palhaço que cai tem de aterrar na parte desocupada do balancé para impulsionar o outro para o ar.

Em princípio ele deverá aterrar na extremidade da prancha. Se aterrar perto do centro, o outro palhaço não sobe o suficiente para acertar em qualquer dos balões. O jogo é tão difícil de dominar que modificámos a ideia original, por forma a incluir palhaços suplementares, só para darmos ao jogador uma oportunidade de rebentar um número razoável de balões.

Ao fazer correr o programa, ver-se-ão assim as instruções:

CIRCO: INSTRUÇÕES

O circo chegou à cidade, e a grande atracção é o número dos palhaços num balancé.

Dois palhaços baloçam-se tentando rebentar os balões no ar.

O palhaço que desce não deve cair no solo nem no lado errado do balancé.

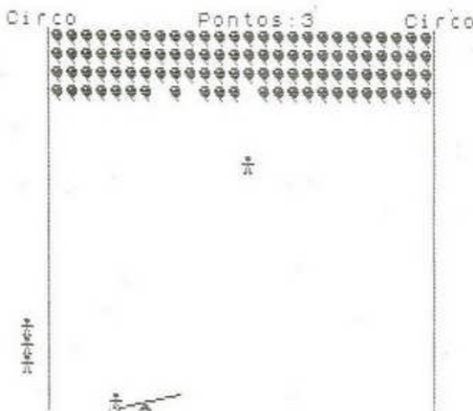
Se ele aterra perto do meio, o outro palhaço não sobe o bastante para acertar em qualquer balão.

O circo só tem seis palhaços, por isso tenha cuidado.

Você controla o balancé com as teclas "5" e "8". Pode fazer o palhaço no balancé trocar de extremidade pressionando "1".

Pressione "N" para parar ou outra tecla para iniciar.

Eis como se apresenta o programa durante o funcionamento:



Na linha 5 a listagem mostra caracteres usados pelos gráficos definidos pelo utilizador, com "A" para o eixo (centro) da prancha, "B" para o palhaço e "C" para o balão.

Depois de se colocar na linha 20 o gerador de números aleatórios, o programa vai para a linha 80, onde a instrução OVER se coloca a 1, os caracteres a preto, o fundo a branco e o

enquadramento a vermelho, isto tudo antes de o *écran* ser limpo (CLS) por forma a estabelecer a cor do fundo.

Depois o programa desloca-se para a sub-rotina da linha 900, onde se chamam mais duas sub-rotinas (as de 1100 e de 1020).

A primeira delas (de 1100) imprime o título e as instruções no *écran*, e a segunda (a da linha 1020) define os gráficos. Ao voltar desta sub-rotina, desenha-se a moldura à volta do *écran* (linhas 902 a 905) e colocam-se os balões no lugar pelo ciclo I (de 910 a 930). O ciclo I também coloca os «palhaços suplentes» no lado esquerdo do *écran*.

Nas próximas duas linhas define-se um certo número de variáveis. A variável SC conta o número de balões que já foram atingidos (variável que é incrementada na linha 180). SS representa a posição horizontal da base do baloioço, usada na linha 50 para o colocar e ao palhaço numa dada posição e que na linha seguinte servirá para desenhar a própria prancha do baloioço. Note-se que as linhas 40 e 60 formam uma sub-rotina repetidamente chamada durante a execução do programa.

A variável M conta os «palhaços que estão em jogo» em cada momento e é incrementada na linha 310 após ter sido para lá remetida pela linha 230. Usa-se o vector B\$ para imprimir 26 espaços em branco na linha 330 depois de cada palhaço já ter cumprido a sua tarefa. As variáveis OD e AN controlam a posição do palhaço. Mudando para as suas simétricas algébricas (como na linha 107), permite-se ao palhaço que mude de extremidade da prancha. Usam-se OD e AN nas linhas 50, 260 e 280.

Com a variável CA detecta-se quando se deve mudar AN (ver linhas 95 e 270), e define-se HL na linha 250, que se usa em 160 para determinar quando a coordenada Y do palhaço que está a saltar (deslocamento vertical) deverá começar a ser incrementada na direcção oposta àquela em que se está a mover (ver 160 e 140). X é a coordenada horizontal de partida do palhaço usada na linha 130 e incrementada pela variável A na linha 140, variável essa que muda na linha 280 para inverter o movimento horizontal do palhaço.

A variável Y (definida na linha 950) é a coordenada vertical de

partida do palhaço. Este valor começa algures entre "5" e "8".

Na linha 100 lê-se o teclado para "5" e "8", mudando a coordenada horizontal da base do balancé.

Verifica-se esse valor na linha seguinte (105), assegurando que o baloioço não se desloca em demasia para a esquerda ou para a direita.

```

5 REM U.D.G.  A=B  C=
7 REM
10 REM CIRCO.  P.TOLAND
15 REM -----
20 RANDOMIZE
40 GO TO 80
50 PRINT AT 21,ss;"A";AT 21,ss
+od;"A"
60 PLOT ss*8-16,7*(an=1): DRAW
38,7*-an: RETURN
80 OVER 1: INK 0: PAPER 7: BOR
DER 2: CLS
85 GO SUB 900: GO SUB 50
90 GO SUB 50
95 IF ca THEN LET an=-an: LET
ca=0
100 LET ss=ss+(INKEY$="8")-(INKE
Y$="5")
105 LET ss=ss+(ss=4)-(ss=27)
107 IF INKEY$="1" THEN LET an=-
an: LET od=-od
110 GO SUB 50
130 PRINT AT y,x;"A"
140 LET x=x+a: LET y=y+d
150 IF x=3 OR x=28 THEN LET a=-
a: BEEP 0.1,1
160 IF y<hl THEN LET d=-d
165 LET no=ATTR(y,x)
167 PRINT AT y,x;"A"
170 IF no=56 THEN GO TO 210
180 LET sc=sc+61-no: BEEP 0.1,1
0
185 PRINT AT y,x: OVER 0;"A";AT
0,20:sc
190 LET d=-d
195 IF sc=260 THEN GO TO 340
200 GO TO 90
210 IF y<21 THEN GO TO 90
220 LET dis=x-ss: BEEP 0.01,5
230 IF dis<-2 OR dis>2 OR SGN d
is<>-an THEN GO TO 310
250 LET hl=11-ABS dis*5
260 LET x=ss+od

```



```

270 LET od=dis: LET ca=1
280 LET a=SGN od: LET d=-1
290 GO TO 90
310 LET m=m+1
320 BEEP .2, -10: BEEP .7, -20
330 IF m<5 THEN PRINT AT 21,3:
OVER 0,b$: GO SUB 950: GO SUB 50
: GO TO 90
335 BEEP 1, -40: RUN
340 PRINT AT 9,2: FLASH 1;"CONS
EGUIU!!!PONTUACAO MAXIMA"
345 BEEP 1,20: BEEP 2,30
350 PAUSE 1: PAUSE 100
360 GO SUB 1100: GO SUB 1020
362 PLOT 23,0: DRAW 0,167
365 PLOT 232,0: DRAW 0,167
370 FOR i=1 TO 4
380 PRINT AT i,3: INK i;"*****
*****"
385 PRINT AT 15+i,1;"X"
390 NEXT i
395 PRINT AT 0,0:"Circo
Pontos:0 Circo"
400 LET sc=0: LET ss=15: LET m=
0: DIM b$(26)
405 LET od=-2: LET ca=0: LET hl
=0: LET x=3: LET y=5+INT (RND*4)
: LET a=1: LET d=1: LET an=-1
415 PRINT AT 15+m,1: OVER 0;" "
420 FOR i=5+m TO y STEP -1
425 PRINT AT i,1;"X"
430 BEEP .05,21-i
435 PRINT AT i,1;"X"
440 NEXT i
445 PRINT AT y,x;"X"
450 RETURN
455 RESTORE 1070
460 FOR i=0 TO 23
465 READ n: POKE USR "a"+i,n
470 NEXT i
475 RETURN
480 DATA 0,0,0,0,16,56,124,254
485 DATA 56,56,16,254,16,40,40,
108
490 DATA 0,60,110,126,62,28,8,4
108
495 PRINT "Circo: INSTRUcoes,"
500 PRINT "O circo chegou a c
idade e a grande atraccao e um
numero de palhaços num balance
, Dois palhaços bala
icam-se, ten-tando rebentar os ba
loes no ar,"

```

```

1115 PRINT " O palhaco que desc
e nao deve cair no solo nem no
lado erra- do do balance.Se ele
aterra per-to do meio, o outro
palhaco nao subira o bastante pa
ra acertar em qualquer balao.O
circo so temseis palhaços, por i
sso tenha cuidado,"
1120 PRINT " Voce controla o ba
lance, com astecias 5 e 8.Pode fa
zer o palha-co no balance trocar
de extremi-dade pressionando "1
",Pressione "N"para parar ou out
ra tecla pa-ra iniciar,"
1130 IF INKEY$="" THEN GO TO 113
0
1140 IF INKEY$="n" OR INKEY$="N"
THEN STOP
1150 CLS
1160 RETURN

```

PEÕES

Neste jogo, ajudamos cinco peões a atravessar uma movimentada auto-estrada com seis faixas de rodagem. Um por um, os peões confiam-nos a vida à medida que os vamos conduzindo através do tráfego, usando para isso as teclas "5" (esquerda), "8" (direita), "6" (cima) e "7" (baixo).

Um só erro será o fim deles.

Aqui trabalhamos não apenas contra o tráfego mas também contra o relógio. Cada vez que conseguirmos que os cinco fatigados peões atravessem, dizem-nos quanto tempo demorou (tempo esse medido na conhecida unidade "pulsações"), e após um curto intervalo para retomar o fôlego lá estarão mais cinco peões em linha do outro lado da auto-estrada esperando pela nossa assistência. Como se observa nas duas listagens, há sempre cinco peões no *écran*, esperando pacientemente nos cantos inferior esquerdo ou superior direito, e por vezes na auto-estrada. Há indicação de quantos são os que ainda precisam de assistência (como se não soubéssemos contar!), bem como quantos já concluíram a viagem até ao topo do *écran*. A informação de tempo da segunda linha sob "Peões salvos" é actualizada sempre que se consegue fazer passar outro peão pelo tráfego.

Dominado este jogo, pode converter-se o programa numa segunda versão.

Nesta variante o peão mexe-se continuamente e por isso é necessária uma grande concentração para o ajudar (ou para a ajudar) a passar através do tráfego.

Sugerimos que se crie grande familiaridade com este jogo na sua versão original antes de se fazerem as modificações que darão o "passeante de movimento perpétuo".

O programa começa na linha 20 com a chamada à sub-rotina da linha 410.

Aqui, colocam-se os caracteres a azul, o fundo a branco, o enquadramento a vermelho e limpa-se o *écran* (CLS).

O ciclo A das linhas 440 a 500 define os seis gráficos.

O primeiro (gráfico A, ou carácter 144) é o peão e os outros são os automóveis.

Vê-se neste ciclo (linha 450) que se define o vector literal E\$ como igual ao carácter 143 mais o valor de A.

O ciclo interior B lê (READ) os dados (DATA) das linhas 560 a 610, e de seguida faz o POKE desses dados na linha 480.

A linha 510 inicializa a variável TEMPO (que dá conta do tempo decorrido por cada cinco peões salvos, medido em "pulsações") e inicializa a variável MELHOR TEMPO com um número extremamente elevado. Pode-se observar a grandeza deste número, 9E7, escrevendo muito simplesmente no computador PRINT 9E7 e carregando de seguida em ENTER.

O melhor tempo do jogador será sempre inferior a esse número, mesmo quando finalmente conseguir levar os cinco peões para o outro lado da estrada, pois só esse facto alterará, pela primeira vez, o valor desta variável.

Os vectores literais A\$, B\$, C\$ e D\$ — que "guardam" os carros — são definidos nas linhas 520 a 550. Pode-se colocar quantos carros se quiser nestes vectores desde que cada um deles não tenha mais do que 32 caracteres de comprimento.

De facto, quando se começa a ganhar uma certa "confiança" quanto à primeira versão dos peões procura-se tornar o jogo mais difícil, antes de se entrar na segunda versão, pelo simples aumento do número de carros por vector.

No entanto, os carros não podem ser colocados nos vectores ao acaso.

Os veículos foram desenhados com realismo para estarem virados para a esquerda ou para a direita (há mesmo um tipo de carro que ficará bem qualquer que seja a direcção em que se coloque).

As três faixas de rodagem superiores movem-se para a direita, e as três faixas inferiores para a esquerda. Pode-se usar o gráfico "B" em qualquer dos vectores, mantendo "C" para A\$ e B\$ e reservando os gráficos "D", "E" e "F" para os vectores C\$ e D\$.

Uma vez inicializadas as variáveis e definidos os gráficos, a instrução RETURN na linha 555 manda o comando do programa de volta para o ciclo J que começa em 30. Este ciclo controla os principais elementos do programa e, como se vê, passará por ele cinco vezes, uma por cada passeante.

Na linha 40 imprime-se "Peões salvos" (sempre um a menos do que o valor corrente de J no ciclo) e o tempo que entretanto decorreu.

Na linha 50 imprime-se "Peões restantes", que é a diferença entre o valor corrente de J e cinco (de forma a que quando J for igual a um, os "Peões restantes" sejam quatro, visto que um deles está a atravessar a auto-estrada).

O pequeno ciclo das linhas 60 a 90 imprime uma pequena figura de um peão que conseguiu atravessar a auto-estrada. A linha 100 limpa quaisquer peões que ainda permaneçam no fundo do *écran*; então o ciclo de 110 a 130 imprime no topo um peão por cada um que ainda esteja à espera.

A variável JA é a posição do peão que se está a mover ao longo do *écran*, e JD é a posição no fundo do *écran*.

O peão começa a sua maratona na posição 16,19, que fica aproximadamente no centro do *écran*, perto do fundo e abaixo da primeira fila de automóveis.

A linha 150 imprime um espaço em branco onde o peão será impresso num dado momento, de forma a que, quando o peão se mover, a sua anterior posição seja apagada. As linhas 160 e 170 lêem o teclado, movendo o peão de acordo com os desejos do

jogador antes de ser impresso na linha 180.

A linha 190 imprime todas as faixas de rodagem. Os PRINT AT (imprimir em) são encadeados, pois assim a instrução corre mais rapidamente. As linhas 200 a 230 comparam a posição actual do peão com o conteúdo do vector nessa posição; se o peão e um carro por acaso coincidem, a atenção desloca-se para a rotina da linha 350, onde se invoca a rotina de atropelamento do peão.

As linhas 250 a 280 controlam o movimento dos carros usando a "divisão de vectores em fatias" (*string slicing*) que o Basic Sinclair nos proporciona para mover elementos ao longo dos vectores.

A variável TEMPO é incrementada na linha 290. A linha 300 verifica o valor de JD, e se esse valor for 2 o computador sabe que outro peão conseguiu atravessar o Rubicão. Ouve-se uma bela série de sons antes de o NEXT J no fim da linha 300 mandar a acção de volta à linha 40, para que o próximo intrépido peão enfrente os carros.

Se um peão não conseguiu chegar ao fim (ou seja, JD é diferente de 2) o computador ignorará o resto da linha 300 e chegará a 310. Se J é menor do que seis, o computador devolve a acção à linha 150, onde se elimina o peão antes de ser reimprimido pela linha 180.

Se todos os peões chegaram ao fim são e salvos, a linha 320 imprime "Desta vez:" e o tempo que isso levou; coloca um "5" no fim da frase "Peões salvos:" e imprime no écran "CONSEGUIU!". A linha 330 diz quanto demorou, e a linha 340 manda o programa para a linha 620.

A linha 620 compara o tempo do jogo que acabou de completar com o MELHOR TEMPO, alterando MELHOR TEMPO para TEMPO, caso TEMPO seja inferior (há um novo recorde).

Se MELHOR TEMPO foi diferente de 10000 (o que pode ocorrer na primeira vez que se tenta jogar e que não se completa o jogo, como se explica adiante), aparece no écran o melhor tempo, a fim de se saber como nos saímos. Um par de trinos assinala o fim do programa (linha 640); a variável TEMPO volta

a zero, e o programa vai para 30 para se começar novo jogo.

Se não se conseguiu levar todos os peões até ao outro lado da auto-estrada, a rotina da linha 350 dispara (como já dissemos, as linhas IF/THEN de 200 a 230 mandam para aqui). O ciclo 350 a 370 imprime um peão que pisca no local onde foi atingido por um carro; soam alguns *beeps* e aparece escrito (o que de resto já sabíamos!) que "FOI ATINGIDO". A variável TEMPO é colocada a 10000 mas não se imprime e não se altera o valor da variável MELHOR TEMPO.

Se a variável TEMPO não fosse aumentada artificialmente neste ponto, poderia obter-se um brilhante MELHOR TEMPO simplesmente correndo contra a primeira fila de carros.

Uma vez que se tenha dominado este jogo, experimente-se com mais carros ou inventem-se novos veículos. Alguns cavalos, uma bicicleta ou duas e um riquexó são bom divertimento. Quando esta versão dos PEÕES já não despertar interesse pode-se substituir a listagem pela segunda que apresentamos. Nesta versão, o peão começa o jogo por se mover e continua a mexer-se. Como esta versão se torna extremamente difícil de jogar é natural que se queira facilitar nas primeiras vezes, reduzindo para isso o número de carros ou adicionando:

195 BEEP .05, RND*50

para abrandar para uma velocidade mais controlável. Boa sorte, Peão, e que atravesse a auto-estrada com segurança.

E esta é a listagem do programa:

```
10 REM OS PEÕES
20 GO SUB 410
30 FOR J=1 TO 5
40 PRINT AT 1,2;"Peões salvos:"
  J=J-1;AT 2,0;"Tempo até agora:"
  tempo;
50 PRINT AT 20,2;"Peões restantes:"
  5-J
60 PRINT AT 1,24;
70 FOR Z=1 TO J-1
80 PRINT INK 2;"X";: REM GRAFI
CO A
90 NEXT Z
```



```

100 PRINT AT 20,24;" ";AT 2
0,24;
110 FOR z=1 TO 5-J
120 PRINT INK 2;"X"; REM GRAFI
CO A
130 NEXT z
140 LET ja=16; LET jd=19
150 PRINT AT jd,ja;" "
160 LET ja=ja+(INKEY$="8" AND J
a<31)-(INKEY$="5" AND ja>0)
170 LET jd=jd+(INKEY$="6" AND J
d<19)-(INKEY$="7")
180 PRINT AT jd,ja; INK 2;"X"
190 PRINT AT 3,0; INK 2;a$;AT 9
,0; INK 0;a$;AT 6,0; INK 4;b$;AT
12,0; INK 3;c$;AT 18,0; INK 1;c
$;AT 15,0; INK 2;d$
200 IF (jd=3 OR jd=9) AND a$(ja
+1)<>" " THEN GO TO 350
210 IF jd=6 AND b$(ja+1)<>" " T
HEN GO TO 350
220 IF (jd=12 OR jd=18) AND c$(
ja+1)<>" " THEN GO TO 350
230 IF jd=15 AND d$(ja+1)<>" "
THEN GO TO 350
240 LET a$a$(31)+a$( TO 31)
250 LET b$b$(31)+b$( TO 31)
260 LET c$c$(31)+c$(1)
270 LET d$d$(31)+d$(1)
280 LET tempo=tempo+1
300 IF jd=2 THEN FOR z=1 TO 20:
BEEP .01,z: NEXT z: NEXT J
310 IF J<6 THEN GO TO 150
320 PRINT INK 2;AT 2,0;"Desta v
ez: ";tempo;"
330 AT 10,10; FLASH 1; PAPER 6;
BRIGHT 1;"CONSEGUIU!"
340 PRINT TAB 4;"DEMOROU ";te
mpo;" PULSACOES"
350 GO TO 620
360 FOR z=1 TO 20
370 PRINT AT jd,ja; FLASH 1; BR
IGHT 1; INK 2;"X"; BEEP .008,z
380 NEXT z
390 PRINT AT 10,10; INK 2; FLAS
H 1;"FOI ATINGIDO"
400 PRINT AT 12,8;"TENDE OUTRA
VEZ"
410 LET tempo=10000: GO TO 620
420 INK 1; PAPER 7; BORDER 2: C
LS
440 FOR a=1 TO 6

```

```

450 LET e$=CHR$(143+a)
460 FOR b=0 TO 7
470 READ c
480 POKE USA e$+b,c
490 NEXT b
500 NEXT a
510 LET tempo=0: LET melhortemp
o=9e7
520 LET a$=""
530 LET b$=""
540 LET c$=""
550 LET d$=""
555 RETURN
560 DATA 28,28,8,62,8,28,34,58
570 DATA 0,60,36,231,255,102,0,
0
580 DATA 240,144,158,255,254,10
2,0,0
590 DATA 0,0,30,99,255,27,0,0
600 DATA 0,0,58,40,126,36,0,0
610 DATA 0,0,30,51,255,34,0,0
620 IF tempo<melhortempo THEN L
ET melhortempo=tempo
630 IF melhortempo<>9e7 THEN PR
INT AT 4,2; FLASH 1;"MELHOR TEMP
O ATE AGORA: ";melhortempo
640 FOR z=50 TO 1 STEP -1: BEEP
.05,z: BEEP .05,-z: NEXT z
650 CLS
660 LET tempo=0
670 GO TO 30

```

Observe-se como o programa funciona:

```

Peoes salvos: 2      XX
Tempo ate agora: 184  XXXX
Peoes restantes: 2   XX
Peoes salvos: 5      XXXX
Desta vez: 674
MELHOR TEMPO ATE AGORA: 674
CONSEGUIU!
DEMOROU 674 PULSAÇÕES
Peoes restantes: 0

```

Eis a segunda listagem:

```

40 PRINT AT 1,2;"Peoes salvos:
";J-1;AT 2,0;"Tempo ate agora:
";tempo;
50 PRINT AT 20,2;"Peoes restan
tes: ";5-J
60 PRINT AT 1,24;

```

```

70 FOR z=1 TO J-1
80 PRINT INK 2;"X"; REM GRAFI
CO A
90 NEXT z
100 PRINT AT 20,24;" ";AT 2
0,24;
110 FOR z=1 TO 5-J
120 PRINT INK 2;"X"; REM GRAFI
CO A
130 NEXT z
140 LET Ja=16; LET Jd=19
150 PRINT AT Jd,Ja;" "
155 LET z$=INKEY$; IF z$="5" OR
z$="8" THEN LET z$=x$
160 LET Ja=Ja+(z$="8" AND Ja<28
)-(z$="5" AND Ja>3)
170 LET Jd=Jd+(z$="6" AND Jd<19
)-(z$="7" AND Jd>21)
175 LET x$=z$
180 PRINT AT Jd,Ja; INK 2;"X"
190 PRINT AT 3,0; INK 2;a$;AT 9
,0; INK 0;a$;AT 8,0; INK 4;b$;AT
12,0; INK 3;c$;AT 18,0; INK 1;c
$;AT 15,0; INK 2;d$
200 IF (Jd=3 OR Jd=9) AND a$(Ja
+1)<>" " THEN GO TO 350
210 IF Jd=5 AND b$(Ja+1)<>" " T
HEN GO TO 350
220 IF (Jd=12 OR Jd=18) AND c$(
Ja+1)<>" " THEN GO TO 350
230 IF Jd=15 AND d$(Ja+1)<>" "
THEN GO TO 350
240 LET a$=a$(31)+a$( TO 31)
250 LET b$=b$(31)+b$( TO 31)
260 LET c$=c$(2 TO )+c$(1)
270 LET d$=d$(2 TO )+d$(1)
280 LET tempo=tempo+1
290 IF Jd=2 THEN FOR z=1 TO 20:
BEEP .01,z; NEXT z; LET x$="8";
NEXT J
310 IF J<5 THEN GO TO 150
320 PRINT INK 2;AT 2,0;"Desta v
ez: ";tempo;
330 PRINT "TAB 4;"DEMOROU ";te
mpo;" PULSAÇÕES"
340 GO TO 620
350 FOR z=1 TO 20
360 PRINT AT Jd,Ja; FLASH 1; BR
IGHT 1; INK 2;"X"; BEEP .008,z
370 NEXT z

```

```

380 PRINT AT 10,10; INK 2; FLASH
H 1; "FOI ATINGIDO"
390 PRINT AT 12,9; "PEDES SALVOS
: J-1
400 LET tempo=10000: GO TO 620
410 INK 1: PAPER 7: BORDER 2: C
LS
420 LET x$="8"
440 FOR a=1 TO 6
450 LET e$=CHR$(143+a)
460 FOR b=0 TO 7
470 READ c
480 POKE USA e$+b,c
490 NEXT b
500 NEXT a
510 LET tempo=0: LET melhortempo
0=997
520 LET a$="      0000 0000 0000
  00 00 00 00
530 LET b$="      000  00  00
  00 00 00 00
540 LET c$="      00000000
  00 00 00
550 LET d$="      0000000000 00 00
  00 00 00
555 RETURN
560 DATA 28,28,8,62,8,28,34,66
570 DATA 0,60,36,231,255,102,0,
0
580 DATA 240,144,156,255,254,10
2,0,0
590 DATA 0,0,30,99,255,27,0,0
600 DATA 0,0,56,40,100,36,0,0
610 DATA 0,0,30,51,205,34,0,0
620 IF tempo<melhortempo THEN L
ET melhortempo=tempo
630 IF melhortempo<>997 THEN PR
INT AT 4,2; FLASH 1; "MELHOR TEMP
O ATE AGORA.": melhortempo
640 FOR z=50 TO 1 STEP -1: BEEP
,05,z: BEEP ,05,-z: NEXT z
650 CLS
660 LET tempo=0: LET x$="8"
670 GO TO 30

```

DESPOLETAR A MINA

Aqui lhe damos um jogo muito difícil de dominar, pois envolve o controle simultâneo de dois objectos no *écran*. Tem por objectivo limpar o *écran* das minas, tarefa esta duplamente difícil porque despoletar uma mina é uma operação em duas fases. Primeiro tem de se desactivar a mina usando o "inversor".

Passando o inversor sobre uma mina armada, torna-a inofensiva, mas, passando-o sobre uma mina desactivada, reactiva-a.

O inversor controla-se com as teclas "5" e "8".

Tal como o inversor, vê-se no *écran* o "colector". Passando-o sobre uma mina previamente desactivada pelo inversor, a mina desaparece do *écran*.

Terá de haver muito cuidado para não acertar com o colector numa mina activa.

O colector também se desloca com as teclas "5" e "8" mas com a tecla SHIFT premida.

Existem ao todo 40 minas e uma escolha entre cinco velocidades possíveis; por isso não é um jogo que canse facilmente. Como se pode ver na listagem, não é nada fácil ter tudo sob controle.

Quando se executa o programa pela primeira vez, a linha 30 manda a acção para a sub-rotina da linha 600, onde se redefinem os gráficos "A", "B" e "C".

A variável Z na última instrução DATA da linha 680 activa o RETURN da sub-rotina da linha 600. De volta à linha 30, a segunda rotina, que aí é chamada, dirige o *Spectrum* para a linha 520, onde se imprimirão as instruções:

O OBJECTIVO CONSISTE EM LIMPAR DE MINAS O *ÉCRAN*. ESTA SERÁ UMA OPERAÇÃO DE DUAS FASES:

1 — A MINA DEVE SER DESPOLETADA PASSANDO O INVERSOR SOBRE ELA. ESTE SERÁ CONTROLADO USANDO AS TECLAS "5" E "8".

2 — A MINA DESPOLETADA TERÁ ENTÃO DE SER RETIRADA PELO COLECTOR. ESTE É CONTROLADO PELAS MESMAS TECLAS QUE O INVERSOR, MAS COM CAPS SHIFT PREMIDA. O COLECTOR NÃO DEVE ATINGIR MINAS ARMADAS.

Assimiladas estas instruções, o computador pergunta qual o grau de dificuldade desejado:

INTRODUZA GRAU DE PERÍCIA 1-5. 5 = FÁCIL
PREMIAR 6 PARA PARAR

A rotina de leitura do teclado das linhas 540, 550 e 560 aceita entradas válidas, rejeitando teclas pressionadas fora da gama "1" a "6" (linha 550) e terminando o programa se a tecla "6" for seleccionada (linha 560). A linha 570 define a variável SP com um valor relacionado com a tecla entre "1" e "5" que foi pressionada. SP controla a dificuldade ou facilidade do jogo. A linha 580 manda então a acção para a linha 40, linha essa colocada após o GO TO que começou a instrução da escolha da dificuldade que se acabou de analisar.

A linha 40 prepara o *écran*. Colocam-se a 5 (azul-claro) os caracteres e o fundo, limpa-se o *écran* (CLS) para estabelecer a cor, e coloca-se então o enquadramento em azul-claro. A cor da tinta muda para preto, e a variável T\$ é igualada a um espaço vazio. As linhas 50 e 60 inicializam variáveis que vão controlar, entre outras coisas, a posição do inversor e do colector, bem como o número de minas desactivadas. A rotina das linhas 70, 80 e 90 desenha a moldura dentro da qual vai decorrer a acção, e a rotina das linhas 100 a 150 coloca as minas no lugar, assegurando (linha 130) que não há minas impressas umas em cima das outras. Se isto sucedesse haveria menos do que 40 minas no *écran* e nunca se assinalaria o fim do jogo porque se puxaria o "gatilho" (disparado no fim da linha 370 caso todas as minas tenham sido desarmadas).

Uma vez que as minas estejam em posição impressas em vermelho (veja-se a mudança de cor no fim da linha 90), a cor da tinta muda para azul-claro na linha 155. I\$ é uma variável usada para interpretar os desejos do jogador. A primeira secção (de 180 a 210) interpreta informação respeitante às teclas de "5" a "8", que controlam o inversor. A parte a seguir (linhas 220 a 260) foca as versões dessas teclas que sofrem a acção da tecla CAPS SHIFT. (Note-se que o CODE — código — de I\$ é tomado na linha 220 e usado para interpretar essas teclas; esta é uma técnica útil se desejarmos ler teclas que sofreram o carregamento de um SHIFT, em qualquer posição no *écran*. Para descobrir o código necessário, basta escrever uma pequena rotina para imprimir o código da tecla que está a ser premida.)

Os resultados destas leituras são confiados às variáveis A, D,

CA e CD, que se adicionam às posições correntes do inversor e do colector (A e D ao inversor, CA e CD ao colector) após as posições anteriores terem sido tapadas com espaços em branco (linha 270).

Dá-se à variável N um valor igual ao atributo (ATTR) da nova posição do inversor, e a CN o valor dos atributos da posição do colector.

A linha 320 reimprime os dois objectos e a linha 325 acrescenta um curto *beep*. Como se vê, a duração desse som está relacionada com o valor introduzido ao princípio para o factor de dificuldade. Quanto mais elevado for o número introduzido maior duração o *beep* produzido na linha 325.

Como se sabe, todas as acções param no *Spectrum* durante o soar do *beep*, e o uso de uma variável como esta é uma maneira útil de obter simultaneamente um som e um grau de controle sobre a velocidade com a qual o programa corre.

Se o programa parecer demasiado difícil, mesmo no nível cinco, muda-se o nove na linha 570 para outro valor mais pequeno. Se N ou CN forem iguais a 40 (linha 330), o computador sabe que o jogador embateu na parede e o programa vai para a linha 400 com a triste notícia: "VOCÊ BATEU NA BARREIRA ELECTRIFICADA".

As linhas 340 e 350 fazem as necessárias mudanças na mina (se o inversor se encontra sobre uma delas), tornando as minas armadas em inofensivas e vice-versa.

Se o valor de CN for 42, o *Spectrum* sabe que o jogador acertou numa mina por desarmar e assim termina o jogo indo para a linha 440, onde se informa, em letras vermelhas intermitentes: "O SEU COLECTOR FOI ATINGIDO POR UMA MINA".

Se se conseguiu retirar uma mina (se o atributo CN é 41, linha 370), o resultado do número de minas retiradas é incrementado em uma unidade. Se S é igual a 40 (fim da linha 370), o computador sabe que todas as minas foram retiradas e activa a linha 470 para o cumprimentar: "VOCÊ CONSEGUIU!!!"

Se nada disto sucedeu (ou seja, não bateu numa mina armada nem retirou ainda todas as minas), a linha 380 manda o programa

de volta à linha 160 para continuar o processo.

O resto do programa, de 400 a 510, contém o fim das mensagens de jogo e algumas "sinfonias" mais ou menos intempestivas interpretadas pelo *Spectrum*.

Quando o jogador já estiver um verdadeiro mestre na arte de despoletar minas, limpe a linha 325 e veja que tal se sai.

Este será o teste final.

```

10 REM UDG A=B=C
20 REM CAMPO MINADO
30 GO SUB 600: GO TO 520
40 INK 5: PAPER 6: CLS: BORDE
R 5: INK 0: LET t$=""
50 LET s=0: LET x=16: LET y=1
60 LET cx=16: LET cy=20: LET c
d=0
65 LET a=1: LET d=0: LET ca=1
70 PLOT 4,4: DRAW 247,0
80 DRAW 0,167: DRAW -247,0
90 DRAW 0,-167: INK 2
100 FOR i=1 TO 40
110 LET tx=INT (RND*30)+1
120 LET ty=INT (RND*18)+2
130 IF SCREEN$(ty,tx)="" THEN
GO TO 110
140 PRINT AT ty,tx;"#"
150 NEXT i
160 INK 5
165 LET i$=INKEY$
170 IF i$="" THEN GO TO 270
180 IF i$="5" THEN LET a=-1: LE
T d=0: GO TO 270
190 IF i$="8" THEN LET a=1: LET
d=0: GO TO 270
200 IF i$="6" THEN LET a=0: LET
ca=1: GO TO 270
210 IF i$="7" THEN LET a=0: LET
d=-1: GO TO 270
220 LET i=CODE i$
230 IF i=8 THEN LET ca=-1: LET
cd=0: GO TO 270
240 IF i=9 THEN LET ca=1: LET c
d=0: GO TO 270
250 IF i=10 THEN LET ca=0: LET
cd=1: GO TO 270
260 IF i=11 THEN LET ca=0: LET
cd=-1
270 PRINT AT cy,cx;" ";AT y,x;t
$: LET t$=""

```

```

280 LET x=x+a: LET y=y+d
290 LET cx=cx+ca: LET cy=cy+cd
300 LET n=ATTR (y,x)
310 LET cn=ATTR (cy,cx)
320 PRINT INK 1;AT y,x;"#";AT c
y,cx;"#"
325 BEEP sp,0
330 IF n=40 OR cn=40 THEN GO TO
400
340 IF n=42 THEN LET t$=CHR$ 16
+CHR$ 1+"#"
350 IF n=41 THEN LET t$=CHR$ 16
+CHR$ 2+"#"
360 IF cn=42 THEN GO TO 440
370 IF cn=41 THEN LET s=s+1: IF
s=40 THEN GO TO 470
380 GO TO 160
400 PRINT FLASH 1: INK 0;AT 8,5
;"VOCE BATEU NA BARREIRA";AT 9,9
;"ELECTRIFICADA"
410 FOR i=1 TO 10: BEEP .02,-0
420 BEEP .02,5: NEXT i
430 GO TO 500
440 PRINT FLASH 1: INK 2;AT 8,3
;"O SEU COLECTOR FOI ATINGIDO";A
T 9,10;"POR UMA MINA"
450 BEEP .2,-20: BEEP .5,-5
460 GO TO 500
470 PRINT AT 1,3: INK 0;"VOCE C
ONSEGUIU!!"
480 BEEP 1,9
500 INK 0
510 PRINT AT 12,7;" MINAS RETIR
ADAS: ";s
515 FOR l=1 TO 500: NEXT l: CLS
520 PRINT "-----CAMPO MINA
DO-----"
O objectivo consis
te em limpar de minas o ecran. Es
ta sera uma operacao de duas fas
es:
1-A mina deve ser
despoletada passando o inversor
sobre ela. Este sera controlado
usando as teclas de 5 a 8.
2-A mina despoleta
da tera entao de ser retirada pe
lo colector. Este e controlado pe
las mesmas teclas que o colecto
r, mas com CAPS SHIFT premida. O
colector nao deve atingir min

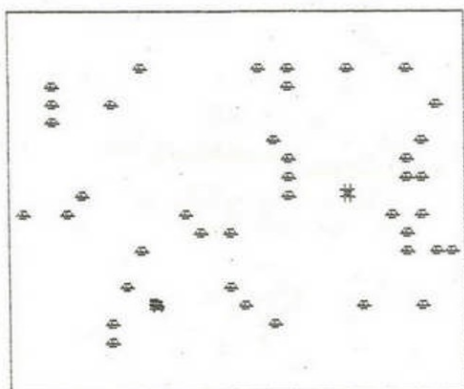
```



```

as armadas,"
530 PRINT "INTRODUZA O GRAU D
E PERICIA 1-5 S=FACIL PREMIR
5 PARA PARAR"
540 LET i$=INKEY$
550 IF i$<"1" OR i$>"5" THEN GO
TO 540
560 IF i$="5" THEN STOP
570 LET sp=VAL i$/9
580 GO TO 40
590 READ a$: IF a$="Z" THEN RET
URN
610 FOR i=0 TO 7
620 READ n: POKE USA a$+i,n
630 NEXT i
640 GO TO 500
50 DATA "A",0,0,0,60,82,255,60
,0
660 DATA "B",36,36,255,60,60,25
,36,36
670 DATA "C",0,0,248,254,254,25
,255,34
680 DATA "Z"

```



UDG A=B B=C

HELICÓPTERO

Aqui está um programa para testar a perícia de voo do jogador. Haverá quatro helicópteros, e a tarefa consiste em limpar um conjunto de linhas vermelhas de ataque. Tem de se aterrar em primeiro lugar na linha mais curta que se encontrar; a seguir na segunda mais curta, e assim por diante até ter aterrado em todas elas na ordem correcta. Um movimento em falso e o jogador perde o seu helicóptero.

Há quatro helicópteros com os quais se pode completar a missão, obtendo-se um bônus em pontos se não for necessário utilizar os quatro aparelhos.

As pequenas máquinas serão conduzidas usando "0" para subir e "5" e "8" para se mover para a esquerda e para a direita, respectivamente.

A gravidade arrastará automaticamente os helicópteros para terra, se não se aplicar potência ao aparelho. A pontuação final está relacionada com a quantidade de combustível que sobrou. Se todos os helicópteros se esmagarem no solo antes de terem destruído as linhas vermelhas será preciso enfrentar a desagradável mensagem "helicópteros abatidos". Existe ainda uma característica especial de pontuação elevada (melhor resultado).

A linha 10 coloca o enquadramento a preto, e a variável H é inicializada a zero na linha 15 (para a pontuação elevada, ver linhas 532 e 535) antes de seleccionar a cor amarela (linha 20) e o fundo negro (linha 30). Depois de tudo isto, o programa dirige-se para a sub-rotina que começa a partir da linha 610, que faz o POKE de dados da linha 900 para o carácter gráfico "A" por forma a criar o pequenino helicóptero que poderá ver na linha 45.

Após a definição do gráfico limpa-se o écran e coloca-se o fundo a negro.

O ciclo das linhas 50 a 70 imprime um bloco amarelo compacto, usando um fundo amarelo e uma fila de espaços dentro das aspas. Dois vectores são DIMensionados para guardar e ordenar as linhas onde os helicópteros deverão aterrar.

O ciclo A corre entre as linhas 100 e 180, usando o gerador de número aleatório da linha 110 e as instruções PLOT (linha 150) e

DRAW (linha 160) para produzir uma base de aterragem mais eficaz, como ficará bem presente ao executar-se o programa. A rotina nas linhas 120 e 130 assegura que se guardam dez linhas de diferentes comprimentos no vector D.

A variável R é inicializada a 1 na linha 185. Esta variável governa a linha em que o helicóptero deve aterrar (quando R é um, o jogador está a apontar para a linha representada por D[R]). A variável de combustível F é colocada a 25000 na linha 190. Além de controlar o combustível, usa-se F para comandar o timbre do gerador de som da linha 240 (F dividido por 1000).

Como se notará, produz-se um som bem adequado à acção do helicóptero. O combustível decresce 21 unidades nas linhas 250 (se o jogador estiver a fornecer potência) e 280. A variável P, a coordenada horizontal, coloca-se a 16 na linha 210, antes de a minúscula nave ser lançada na linha 220. A linha 245 imprime a mensagem "helicópteros que sobram" usando L\$, definido na linha 45 como sendo três helicópteros) no canto inferior direito do écran, e — depois de o combustível ter decrescido na linha 250, e a quantidade que sobrou ter sido impressa na linha 255 no canto inferior esquerdo do écran — a posição do helicóptero é "limpa", antes do seu reaparecimento. A variável U (definida inicialmente como um, na linha 200) controla o movimento vertical do aparelho (quanto mais alto o U mais baixo está o jogador), e as linhas 275 e 290 lêem as intenções expressas através do teclado ("0" para subir, "5" para deslocar para a esquerda, "8" para deslocar para a direita) antes de o programa saltar para a linha 220 a fim de voltar a imprimir o helicóptero, mas agora numa nova posição.

Uma aterragem é detectada na linha 230, o que dirige o programa para a linha 400, onde se verifica o local de aterragem. Se não se tratar da linha mais curta — ou seja P não é igual a D[R] — ouvir-se-á um ruído curto (BEEP .01,-10) e subtraem-se 1000 unidades de combustível antes de o helicóptero ser impresso em modo inverso (INVERSE) durante um momento (linha 412), ouvindo-se outro ruído (linha 413) quando o helicóptero desaparece (linha 414).

Verifica-se o vector L\$. Se for igual ao vector nulo (" "), o

computador sabe que os helicópteros se esgotaram. Recordemos que L\$ era igual a três helicópteros na linha 45. A linha 416 corta um helicóptero do fim deste vector após cada aterragem, de forma que quando L\$=" ", o computador sabe que os quatro helicópteros já foram utilizados (o quarto helicóptero inicialmente não fazia parte de L\$). A linha 418 devolve a acção à linha 200, onde o próximo helicóptero iniciará a sua missão.

A linha 400 verifica se a aterragem foi bem sucedida. No caso afirmativo, salta sobre as más notícias das linhas 405 a 418 e passa directamente para 420, onde se apaga o helicóptero, e o topo das linhas vermelhas é "barbeado" pela linha 425. Um curto ciclo de BEEPs concede ao jogador uma pausa para recuperar o fôlego, e a variável R (que conta o número de linhas) é incrementada uma unidade. Se R for menor do que onze, haverá ainda linhas onde aterrar. Caso contrário, os helicópteros não utilizados são apagados pela linha 490, e o combustível é incrementado em 1000 unidades por helicóptero que tenha sobrado, assegurando, assim, o bónus por aparelhos não utilizados. A nova quantidade de combustível conta-se progressivamente (linha 520) à medida que se renova.

Se necessário, ajusta-se o resultado mais elevado na linha 532, sendo impresso pela linha seguinte (535) antes de ir para a linha 310, para anunciar o final do jogo. Não se tendo conseguido limpar todas as linhas (ou seja, se R é menor do que onze), surgirá a mensagem de insucesso "Helicópteros abatidos". As linhas 330 e 340 esperam que o jogador retire o dedo do teclado (linha 330) no final de um jogo, voltando a colocá-lo (linha 340) para assinalar que está pronto a jogar novamente. Neste caso, o programa move-se para a linha 40, que limpa o écran (mas que não redefine os caracteres nem limpa o valor da variável H que guarda o valor do maior resultado) e determina outra missão com quatro novos helicópteros. Bons voos.

```
5 REM HELICOPTERO
10 BORDER 0
15 LET h=0
20 INK 6
30 PAPER 0
```

```

35 GO SUB 610
40 CLS
45 LET l$="***"
50 FOR a=11 TO 21
60 PRINT AT a,0; PAPER 6;" "

70 NEXT a
80 DIM d(10)
90 DIM e(32)
100 FOR a=1 TO 10
110 LET b=INT (RND*32)
120 IF e(b+1)=1 THEN GO TO 110
130 LET e(b+1)=1
140 LET d(a)=b
150 PLOT b*8+4,87
160 DRAW INK 2;0,-a*8+1
180 NEXT a
185 LET r=1
190 LET f=25000
200 LET u=0
210 LET p=15
220 PRINT INK 4;AT u,p;"*"
230 IF u=9+r THEN GO TO 400
240 BEEP .01,f/1000
245 PRINT INVERSE 1;AT 21,27;l$
" "
250 LET f=f-21
255 PRINT AT 21,0; INVERSE 1;"c
ombustivel ";f;" "
260 PRINT AT u,p;" "
270 LET u=u+1
275 LET i=IN 61438
280 IF INT (i/2)=i/2 THEN LET u
=u-2*(u>1): LET f=f-21
285 LET k=INT (i/4): LET j=INT
((IN 63486)/16)
290 LET p=p+(INT (k/2)=k/2)*(p<
31)-(INT (j/2)=j/2)*(p>0)
300 GO TO 220
310 PRINT AT 7,10; INK 5;"FIM D
O JOGO"
320 IF r<11 THEN PRINT AT 21,0;
PAPER 1; FLASH 1;"helicopteros
abatidos"
330 IF INKEY$<>" " THEN GO TO 33
0
340 IF INKEY$="" THEN GO TO 340
350 GO TO 40
400 IF p=d(r) THEN GO TO 420
405 BEEP .01,-10
410 LET f=f-1000
412 PRINT INVERSE 1;AT u,p;"*"

```

```

413 BEEP .1,-20
414 PRINT AT u,p;" "
415 IF l$="" THEN GO TO 310
416 LET l$=l$(2 TO )
418 GO TO 200
420 PRINT AT u,p;" "
425 PRINT AT 10+r,0;" "

430 FOR a=0 TO 20
440 BEEP .01,a
450 NEXT a
460 LET r=r+1
470 IF r<11 THEN GO TO 200
480 FOR a=1 TO LEN l$
490 PRINT AT 21,26+a;"■"
500 BEEP .5,a*10
510 LET f=f+a*1000
520 PRINT AT 20,0;"combustivel
";f
530 NEXT a
532 IF h<f THEN LET h=f
535 PRINT INVERSE 1;AT 21,0;"me
lhor resultado ";h
540 GO TO 310
510 FOR b=0 TO 7
520 READ c
530 POKE USR "a"+b,c
540 NEXT b
550 RETURN
900 DATA 0,85,8,26,62,62,20,34

```

TIRO AO ALVO

Em TIRO AO ALVO enfrenta-se um terrível torneio de tiro. É um jogo de feira, em que tem de se abater os patos e os vários objectos que passam à frente do jogador.

No entanto, a versão deste jogo para computador apresenta uns perigos adicionais. Várias filas de objectos movem-se acima do jogador em grupos parcialmente bloqueados por quatro barreiras que se movem na direcção oposta. O objectivo é acertar no maior número possível de objectos sem atingir as barreiras.

A arma move-se (gráfico H) usando as teclas "5" e "8"; a direcção é idêntica à direcção das setas que se encontram desenhadas nessas teclas.

A fiel tecla "F" dispara a arma. Nunca poderá haver mais de uma bala no écran. Há um número limitado de balas e o número

de tiros que sobram é assinalado pelo comprimento da linha que se encontra abaixo da sua arma.

Como se vê pela listagem, o *écran* é uma miscelânea selvagem de objectos que se movem, criando um efeito emocionante.

De tempos a tempos, um pato voará através do *écran*. Obtém-se um bónus de seis pontos ao acertar no pato, e perde-se 15 pontos ao falhar. É bastante fácil acertar, porque o pato voa abaixo das barreiras, como se vê na figura apresentada.

A sub-rotina da linha 600 é chamada da linha 20. Esta rotina define os gráficos determinados pelo utilizador, usando "A", "B", "C", "D", "E", "H", "I" e "J". Quando a linha 600 (READ c\$) passa pelo "Z" na linha 750, este dispara a instrução do RETURN.

De volta ao início do programa (linha 30), põe-se o fundo e o enquadramento em amarelo (cor número seis) e limpa-se o *écran* para estabelecer a cor do fundo.

A linha 35 coloca INK a preto (cor número zero) e o programa vai para a linha 500, onde se imprimem as instruções:

O OBJECTIVO DO JOGO É ACERTAR NOS OBJECTOS QUE SE MOVEM ACIMA DE SI E SIMULTANEAMENTE EVITAR AS LINHAS PRETAS. QUANTO MAIS ALTA A FILA A QUE PERTENCE O OBJECTO ATINGIDO, MAIOR A PONTUAÇÃO ALCANÇADA. MOVA A ARMA COM AS TECLAS "5" E "8" E DISPARE COM A TECLA "F". EM CADA MOMENTO SÓ PODE HAVER UMA BALA NO *ÉCRAN* E O NÚMERO DE TIROS QUE LHE SOBAM EQUIVALE AO COMPRIMENTO DA LINHA ABAIXO DA ARMA. ABATEM-SE 15 TIROS AO NÚMERO TOTAL CADA VEZ QUE UM PATO CONSEGUE CHEGAR AO FIM DO *ÉCRAN*.

Depois destas instruções, pede-se-lhe para premir, a fim de começar o jogo. Estas instruções aparecem no final de cada jogo; por isso pode-se sair do programa nesta altura introduzindo um "N" ou um "n". Qualquer outra tecla, excepto "N" ou "n", dará início a novo jogo.

Os jogos seguintes não disparam a sub-rotina da linha 600, que inicializa os gráficos. De volta à linha 40, as variáveis são inicializadas e dimensiona-se o vector literal que guarda os alvos móveis (A\$). Os ciclos de 60 a 100 designam aleatoriamente alvos para os vectores. São impressos pela linha 95 e cria-se na linha 97 um som musical em crescendo, para assinalar que o programa está a ser preparado. A variável B no fim da linha 100 controla o comprimento da linha que aparece abaixo da arma, ou seja, aquela cujo comprimento está relacionado com o número de tiros que sobram. B decresce na linha 210, e a instrução PLOT OVER, que se segue, limpa o fim da linha.

A\$ (5) é definido como sendo igual às barreiras que se movem, por isso teremos A\$ (1) a A\$ (4) como objectos para abater e A\$ (5) representando as barreiras, que farão o seu melhor para impedir que os tiros atinjam o alvo. A linha 107 assinala com um BEEP que o jogo está prestes a começar.

O ciclo I, das linhas 110 a 350, constitui a parte principal do programa. Controla o movimento dos objectos no ciclo (definindo T\$ como o primeiro elemento do vector literal da linha 120 e adicionando-o na linha 130 ao fim do vector literal, que é privado do seu primeiro carácter gráfico), e nessa altura imprime-os (linha 140), escolhendo diferentes cores por cada linha (a linha superior é azul, a segunda vermelha, a terceira magenta e a quarta linha impressa em verde). Depois disto, a linha 170 limpa a posição onde o certo atirador estava impresso, e a linha 180 altera o valor da sua posição horizontal de acordo com a tecla pressionada, detectada nessa linha (a linha 190 ajusta a posição, se está muito para a esquerda ou para a direita). A figura humana é reimpressa na linha 200. Se B é zero, todos os tiros foram usados, por isso a acção passa da linha 205 para a linha 430 onde se informa que as munições se esgotaram. A linha 210 é aquela que verifica se se pode disparar (se F é igual a um é porque não se encontra nenhuma bala no ar, por isso pode disparar-se à vontade). A linha 210 também torna a coordenada horizontal da bala (FX) igual à posição humana horizontal (G). A variável F é a coordenada vertical da bala, por isso (F, FX) é a localização onde a bala necessita estar impressa e mais tarde apagada. A

linha 240 diminui a coordenada F de uma unidade (o que tem por efeito, quando se imprime de novo a bala, movê-la uma linha acima do *écran*).

A mestria do tiro é testada na linha 250. Se se determinar que SCREEN\$ (F, FX) está vazia, nenhum objecto se apresenta directamente acima da bala. Se se encontra um objecto, aplica-se um teste para verificar se se trata da barreira; as linhas 260 e 265 executam esse teste; se P é igual a cinco é porque se acertou na barreira. Neste último caso, o jogo terminou e o computador vai para a linha 400 para dar as "alegres novidades". Se não atingiu a barreira mas acertou em alguma coisa, essa coisa deve ser algum dos objectos a que estava apontando. A linha 270 dá a pontuação relacionada com a altura relativa do objecto (quanto mais pequeno for o valor de P, mais alto se situará no *écran*). Uma vez atingido um objecto, a bala deve parar; por isso, repõe-se F a um (veja-se o fim da linha 270). Caso se tenha acertado no pato, P será igual a seis; por isso obtém-se um bónus de seis pontos, e repõe-se a zero a variável D. O programa salta daqui para 300 se o pato foi atingido. Se o não foi, imprime-se o resultado, piscando, pela linha 275, e o elemento do vector literal que guarda o objecto atingido é substituído por um espaço, pelo que o objecto desaparece.

Não se tendo acertado em alguma coisa (por isso não se alcança GO TO 300 no fim da linha 280), a linha 290 imprime a bala. Cerca de uma em cada 50 passagens, a linha 300 gerará um novo pato, caso não haja outro no *écran*.

A linha 320 move o pato e a 330 verifica se ele já chegou ao lado direito do *écran*. Se chegou, o pato é apagado por esta linha, repõe-se D a zero (a condição "não pato") e subtraem-se 15 pontos à pontuação antes que seja apagada a frase que assinala a pontuação no *écran*. A última parte da linha 330 (GO TO 350) salta sobre a instrução de impressão do pato da linha 340.

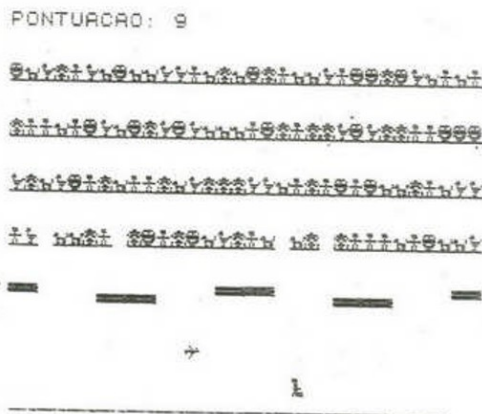
A linha 350 termina este ciclo principal.

Recordemos que A\$ (5) guarda as barreiras que se movem. Em 360 T\$ é igualado ao elemento final em A\$ (5), e este desloca-se para o princípio do vector literal na linha 370. É o oposto do que sucedeu a todos os outros elementos do vector

literal A\$, e por isso as barreiras se movem na direcção oposta (embora à mesma velocidade) em relação aos objectos-alvos.

A linha 380 imprime a barreira e a 390 manda a acção de volta à linha 110, onde o ciclo mestre recomeça. As linhas 400 e 420 são disparadas se acertar na barreira (P é igual a cinco, como atrás se viu) e depois de recordar este facto ao jogador o programa vai para a linha 450, para imprimir o resultado. Se se esgotarem as balas (quer dizer, se a variável B for igual a zero), as linhas 430 e 440 são disparadas: "NÃO HÁ MAIS BALAS". Após as mensagens de final do programa, a página do título reaparece e a introdução de qualquer tecla exceptuando "N" ou "n" fará começar novo jogo.

Eis o programa em acção:



E esta é a listagem do programa:

```
10 REM TIRO AO ALVO
20 GO SUB 800
30 PAPER 6: BORDER 6: CLS
35 INK 0: GO TO 500
40 LET 9=16: LET 7=1: LET 8=0
50 LET 1X=0: DIM A$(5,32): CLS
```

```

60 FOR i=1 TO 4
70 FOR j=1 TO 32
80 LET a$(i,j)=CHR$(INT (RND*
5)+144)
90 NEXT j
95 PRINT AT i*3,0;a$(i)
97 BEEP 4,i*2
100 NEXT i: LET b=255
102 LET d=0: DRAW 255,0
105 LET a$(5)=" "
107 BEEP 1,9
110 FOR i=1 TO 4
120 LET t=a$(i,1)
130 LET a$(i)=a$(i,2 TO 3)+t$
140 PRINT INK i;AT i*3,0;a$(i)
170 PRINT AT 20,g;" "
180 LET g=g+(INKEY$="S")-(INKEY
$="S")
190 LET g=g+(g=-1)-(g=32)
200 PRINT AT 20,g;"H"
205 IF b=0 THEN GO TO 430
210 IF f=1 AND INKEY$="f" THEN
PRINT AT 1,fx;" ": LET f=20: LET
fx=g: BEEP .02,-15: LET b=b-1:
PLOT OVER 1,b,0
220 PRINT AT f,fx;" "
230 IF f=1 THEN GO TO 300
240 LET f=f-1
250 IF SCREEN$(f,fx)=" " THEN
GO TO 290
260 LET p=f/3: BEEP .02,5
265 IF p=5 THEN GO TO 400
270 LET s=s+5-p: LET f=1
272 IF p=6 THEN LET s=s+6: PRIN
T AT 18,fx;" ": LET d=0: GO TO 3
00
275 PRINT FLASH 1;AT 0,0;"PONTU
ACAO: ";s
280 LET a$(p,fx+1)=" ": GO TO 3
00
290 PRINT AT f,fx;"I"
300 IF d=0 AND RND>.98 THEN LET
d=1
310 IF d=0 THEN GO TO 350
320 LET d=d+1
330 IF d=31 THEN PRINT AT 18,31
;" ": LET d=0: LET b=b-15: LET b
=b*(b>=0): DRAW OVER 1;-PEEK 236
77+b,0: BEEP .02,9: GO TO 350
340 PRINT AT 18,d;"J"
350 NEXT i

```

```

360 LET t=a$(5,32)
370 LET a$(5)=t+a$(5, TO 31)
380 PRINT AT 15,0;a$(5)
390 GO TO 110
400 BEEP 2,-9
410 PRINT FLASH 1;AT 0,0;"ATING
IU A BARREIRA"
420 GO TO 450
430 BEEP 1,1: BEEP 1,9
440 PRINT FLASH 1;AT 0,0;"NAO T
EM MAIS BALAS"
450 PRINT FLASH 1; INVERSE 1;"P
ONTUACAO: ";s
455 FOR p=1 TO 500: NEXT p: CLS

```

```

500 PRINT "!!!!!!!!!!!!TIRO AO AL
VO!!!!!!!!!!!!"

```

O objectivo do Jog
o e acertar nos objectos que se
movem acima de si e simultaneame
nte evitar as linhas pretas. Qua
nto mais al-ta a fila a que pert
ence o obje-cto atingido maior a
pontuacao alcançada. Mova a arm
a com as teclas com as teclas S
e 8 e dispare com tecla "F". E
m cada momento pode haver uma
bala no ecran e o numero de tiro
s que lhe so- bram equivale ao c
omprimento da linha abaixo de ar
ma. Abatem-se 15 tiros ao numero
total cada vez que um pato co
nsegue chegar ao fim do ecran"

```

510 PRINT "PRIMA S PARA INICI
AR, N PARA PARAR"
520 IF INKEY$="" THEN GO TO 520
530 IF INKEY$="n" THEN STOP
540 GO TO 40
599 STOP
600 READ c$: IF c$="Z" THEN RET
URN
610 FOR i=0 TO 7
620 READ n: POKE USR c$+i,n
630 NEXT i
640 GO TO 600
650 DATA "A",BIN 00110000,BIN 0
1110000,BIN 00100001,BIN 0011111
0,BIN 00011100,BIN 00001000,BIN
00011000,255
660 DATA "B",0,0,BIN 01100001,B
IN 11100001,BIN 00111110,BIN 000
100110,BIN 00110110,255

```



```

670 DATA "C",BIN 00011000,BIN 0
1111110,BIN 10111101,BIN 0110011
0,BIN 00011000,BIN 10111101,BIN
10011001,255
680 DATA "D",BIN 01111100,BIN 1
0010010,254,254,BIN 11000110,BIN
01111100,BIN 00111000,255
685 DATA "E",BIN 00111000,BIN 0
0111000,16,254,16,40,40,255
690 DATA "H",BIN 00111000,BIN 0
0111000,BIN 00011000,BIN 0001100
0,BIN 00011110,BIN 00011000,BIN
00011111,BIN 00101111
700 DATA "I",0,0,8,28,28,62,
0
710 DATA "J",0,0,32,18,255,BIN
10011000,16,32
750 DATA "Z"

```

UDG- L=H A=I +=J

BATALHA DAS LETRAS

Neste jogo fora do vulgar o jogador controla um helicóptero verde enquanto o *Spectrum* toma conta de um caça a jacto, lutando pela posse do alfabeto.

As letras que pertencem ao jogador encontram-se no canto inferior esquerdo, e as do computador no canto inferior direito.

Usando as teclas "5" e "8" para se mover horizontalmente e "W" e "Z" para se mover na vertical, o jogador deve colocar-se em frente de uma das letras do inimigo que no momento não se encontra a piscar. Pressionando então "O", imediatamente a letra correspondente no seu lado começa a piscar. A letra do inimigo desaparece, o resultado do jogador aumenta e o helicóptero volta para o lado direito do *écran*, pronto para mais uma batalha. Note-se que pode carregar-se em mais do que uma tecla simultaneamente (uma vez que se utiliza a instrução IN em vez de INKEY\$ para se ler o teclado; vejamos as linhas 410 e 420), por isso é possível haver movimento diagonal. Enquanto o jogador faz tudo isto, o computador está igualmente empenhado em lhe roubar as letras.

Pairar em frente às letras e pressionar "O" é uma forma de as obter, mas há outro método. Se o caça azul do computador está

na mesma linha de alguma das letras adversárias que não piscam, o helicóptero pode colocar-se nessa mesma linha e disparar contra o avião usando a letra "O". O computador tentará também disparar contra o helicóptero e por isso a cena pode ficar bastante animada.

O jogo termina quando todas as letras forem capturadas por um lado ou pelo outro. O computador exibirá então o resultado do vencedor. (A propósito, o jogador é designado como "Humanoide" na sequência dos resultados.)

Nas linhas 100 a 150 são definidos os gráficos (as letras usadas são "A", "B", "C" e "D") lidos a partir dos dados da linha 900. Dimensionam-se dois vectores, P e T, e distribuem-se aleatoriamente por cada um deles (e imprimem-se) as 22 letras do alfabeto.

A posição do helicóptero é definida pelos valores de U e P. A coordenada vertical P começa em 29 (ver linha 290), e a coordenada horizontal começa acima da posição onde a última letra do seu lado foi imprimida. A posição do caça começa num local semelhante relativamente à sua última letra usando as variáveis A e T (que começa em 1, ver linha 310). As linhas 410 e 420 lêem o teclado, usando IN, como já se mencionou, e a figura do helicóptero é impressa na linha 430. Depois de decidir capturar uma letra ou atirar ao aparelho do computador, a linha 440 manda a acção para a linha 580, que mais à frente dirige o programa para a sub-rotina da linha 1050, que faz disparar a sua arma laser em caso de necessidade. Se na linha 440 "A" for igual a "U", o programa vai para a linha 700, com o fim de capturar a letra e aumentar a pontuação.

Se estiver muito deslocado para a esquerda, P será igual a um, por isso o computador irá para 700, para executar essas duas tarefas.

A linha 460 limpa a posição do caça do *Spectrum* e usa a rotina seguinte para determinar a sua nova posição. A variável A, que controla a posição vertical do jacto, é alterada na linha 480, e T muda, na linha seguinte, por forma a aproximar o jacto do helicóptero. As linhas 610 e 620 adicionam aos movimentos do jacto uma componente aleatória.

A linha 650 reimprime o caça na sua nova posição, e o princípio da linha 660 determina se esse caça a jacto irá atacar o helicóptero, mandando o programa para a sub-rotina da linha 1000 no caso afirmativo. Se T for igual a 29, o computador sabe que já chegou ao seu lado do *écran*, mandando a acção do programa para a sub-rotina da linha 1100, que captura a letra que interessa. Então o computador apaga a letra do lado esquerdo que se encontra na mesma altura, indo para a sub-rotina de 800, onde faz soar o "trinado" da vitória antes de voltar a colocar a letra no lugar. Se o resultado do jogador (S) somado ao do *Spectrum* (X) der o total de 22, o jogo terminará e a linha 735 volta a dirigir o programa para 1200, que limpa o *écran* antes de ser anunciado o resultado da BATALHA DAS LETRAS.

A linha 745 apaga o jacto da sua antiga posição e redefine P e U antes de regressar à linha 400, para toda esta "dança" recomençar.

```

10 REM BATALHA DAS LETRAS
100 FOR a=1 TO 4
110 FOR b=0 TO 7
120 READ c
130 POKE USR CHR$ (143+a)+b,c
140 NEXT b
150 NEXT a
200 DIM p(2,22)
205 DIM t(2,22)
210 FOR p=1 TO 22
215 FOR t=1 TO 22
220 LET e=INT (RND*22)
230 IF p(f,e+1)>0 THEN GO TO 22
240 LET p(f,e+1)=p
250 PRINT INK 2;AT e,(f=2)*31;C
255 LET t(f,p)=e+1
260 NEXT f
270 NEXT p
280 LET u=t(2,22)-1
290 LET p=29
300 LET a=t(1,22)-1
310 LET t=1
320 DIM h(2,22)
330 LET s=0
340 LET x=0
400 PRINT AT u,p;" "

```

```

410 OUT 254,255: LET p=p+(IN 61
438<>255)*(p<29)-(IN 63486<>255)
440 LET u=u+(IN 65278<>255)*(u<
21)-(IN 64510<>255)*(u>0)
430 PRINT AT u,p; INK 4;" "
440 IF IN 57342<>255 AND p>t TH
EN GO SUB 580: IF a=u THEN GO SU
B 700
450 IF p=1 THEN GO TO 700
460 PRINT AT a,t;
470 IF p(1,u+1)<11 THEN GO TO 5
50
480 LET a=a+(a<u)-(a>u)
490 LET t=t+(ABS (a-u)<ABS (t-p
))-ABS (a-u)>ABS (t-p))*(t>1)
500 GO TO 650
510 IF h(1,p(2,a+1))=1 OR h(2,p
(2,a+1))=1 THEN GO TO 600
520 LET t=t+1
530 GO TO 650
540 GO SUB 1050
550 BEEP .005,20: GO SUB 1050
560 RETURN
570 LET f=RND
580 IF AND<.5 OR u=21 THEN LET
a=a-(a>0)
590 IF AND<.5 OR a=0 THEN LET a
=a+(a<20)
600 PRINT AT a,t; INK 5;" "
610 IF ABS (a-u)<3 AND p>t AND
h(1,p(1,u+1))=0 AND h(2,p(1,u+1
))=0 THEN GO SUB 1000: BEEP .005,
0: GO SUB 1000: IF a=u THEN GO T
O 850
620 IF t=29 THEN GO TO 1100
630 GO TO 400
640 IF h(1,p(1,u+1))=1 OR h(2,p
(1,u+1))=1 THEN GO TO 745
650 PRINT AT u,0;" "
660 GO SUB 800
670 PRINT FLASH 1;AT t(2,p(1,u+
1))-1,31;CHR$ (64+p(1,u+1))
680 LET s=s+((h(2,p(1,u+1)))=0)
690 IF s+x=22 THEN GO TO 1200
700 LET h(2,p(1,u+1))=1
710 PRINT AT u,p;" "
720 LET p=29
730 IF u=a THEN LET u=(a=0)
740 PRINT #1;AT 1,0;"Pontuacao
";(x),"Pontuacao ";(s)
750 GO TO 400
800 BEEP .005,10

```

```

810 BEEP .01,20
820 BEEP .005,10
830 RETURN
850 PRINT AT a,t;" "
860 LET a=u
870 GO TO 1100
900 DATA 0,64,48,62,31,31,31,32
905 DATA 0,0,0,0,192,252,0,0
910 DATA 0,3,7,15,31,8,127,0
920 DATA 0,230,230,246,254,16,2
54,0
1000 PLOT OVER 1;(t+2)*8,(21-a)*
8+3
1010 DRAW OVER 1; INK 3;(p-t)*8-
1,0
1030 RETURN
1050 PLOT OVER 1;p*8,(21-u)*8
1060 DRAW OVER 1; INK 5;(t-p)*8,
0
1070 RETURN
1100 IF h(2,p(2,a+1))=1 OR h(1,p
(2,a+1))=1 THEN GO TO 1160
1105 PRINT AT a,31;" "
1110 GO SUB 800
1120 PRINT FLASH 1;AT t(1,p(2,a+
1))-1,0;CHR$(64+p(2,a+1))
1130 LET x=x+((h(1,p(2,a+1)))=0)
1140 IF x+s=22 THEN GO TO 1200
1150 LET h(1,p(2,a+1))=1
1160 PRINT AT a,t;" "
1170 LET t=1
1180 IF u=a THEN LET a=(u=0)
1190 GO TO 765
1200 CLS
1205 PRINT AT 11,10;"FIM DO JOGO "
1210 PRINT AT 19,0;
1220 PRINT "spectrum ";
1230 FOR a=1 TO x
1240 PRINT "/";
1250 NEXT a
1260 PRINT TAB 25;x;"humanoide "
1270 FOR a=1 TO s
1280 PRINT "/";
1290 NEXT a
1300 PRINT TAB 25;s
1310 IF INKEY$<>" " THEN GO TO 13
10
1320 IF INKEY$="" THEN GO TO 132
0
1330 RUN

```

DEMOLIÇÃO

Nesta versão bastante colorida de *Breakout*, de que já se falou, propõe-se a tentativa de furar através dum muro de tijolos usando um projectil em forma de diamante com arestas aguçadas. No decorrer do jogo vê-se que o gráfico "A" foi redefinido para se assemelhar a tijolos. A parede em que se tenta penetrar tem na realidade um aspecto bastante sólido. Dispõe-se de 12 projecteis para atirar à parede, e a base, que se move lateralmente no fundo do écran, é controlada pelas teclas "5" (para a esquerda) e "8" (para a direita). O resultado mostra-se constantemente no écran e actualiza-se quando necessário.

A linha 15 coloca o enquadramento e o fundo a vermelho e as letras a branco. A moldura dentro da qual o jogo deve ter lugar é impressa a verde pela linha 20 (o topo da moldura) e pelo ciclo da linha 30. (Como é evidente, não se torna necessário seguir à risca o nosso esquema de cores em qualquer dos programas; diversos tipos de televisores reagem de modo diferente à cor do *Spectrum*, e combinações que resultam bem num televisor podem não ser muito adequadas noutro: por isso deve-se experimentar a melhor combinação). A variável S (para a pontuação) é inicializada a zero na linha 35, e A e B (que controlam o curso do projectil) são ambas inicializadas a um nessa mesma linha. Nas linhas 40 e 42 encontram-se os ciclos que definem os tijolos e o projectil, respectivamente, desenhados a partir dos dados das linhas 210 e 215.

O ciclo seguinte, das linhas 45 a 55, imprime os tijolos. Note-se que a linha 55 consta de 28 gráficos "A". A linha 50 escolhe um número aleatório para determinar a cor de cada fila de tijolos. A segunda parte da linha verifica se a cor escolhida não é o vermelho (ou seja a cor número 2) para evitar que o tijolo fique com a cor da cena. Se verificar que o vermelho foi seleccionado (Z é igual a dois) então regressa ao princípio da linha para escolher nova cor.

A linha seguinte, a 60, determina a posição de partida do projectil. Este é impresso em X,Y e as variáveis usadas para "apagar" o projectil (DX e DY) são definidas iguais a X e Y nesta linha. Se Y é maior do que 28 ou menor do que 3, é porque

o projectil bateu num dos lados, e por isso a linha 65 chama a atenção à linha 155 para lhe alterar a direcção (ou seja, fá-lo ricochetear na parede lateral).

Se X for menor do que dois, é porque o projectil atingiu a parte de cima da moldura e a linha 70 manda-o de volta para baixo. A linha 75 verifica se a posição que o projectil ocupa contém um tijolo; se na realidade isso sucede vai-se para a sub-rotina da linha 125, onde se produz o som de esmagamento do tijolo (!), e altera-se o valor da variável B, usada para incrementar Y.

Se X é menor do que 20, o projectil encontra-se no fundo da moldura. A linha 80, alertada para a posição dele através do valor de X, verifica se a base móvel está na posição correcta para devolver a bola. Se isso acontecer, produz-se um som de batida do projectil (BEEP .008,10) e multiplica-se por menos um a variável "A", que controla a mudança na posição vertical da mesma.

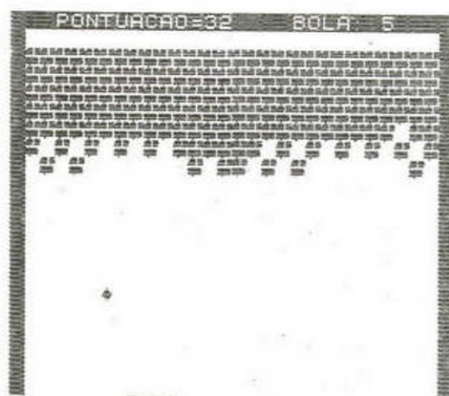
A primeira parte da linha 85 usa as variáveis DX e DY para limpar a antiga posição do projectil, sendo este, depois, novamente impresso usando X e Y.

A linha 90 imprime a base móvel. Repare-se que a base tem um espaço em cada extremidade, o que significa que ao movê-la se limpa a sua anterior posição automaticamente. A rotina na linha 95 lê o teclado e usa a leitura de INKEY\$ e o valor de N para mudar o seu próprio valor, o que determina a posição da base móvel.

A posição do projectil é actualizada na linha 105. Se X é maior do que 21 (linha 110), o computador sabe que o projectil falhou a base móvel e caiu abaixo do fundo da moldura. A sub-rotina da linha 165 imprime o número do projectil, incrementa uma unidade à variável do projectil (Q) e cria um ruído na linha 175 para assinalar a entrada de um novo projectil. Na linha 108 verifica-se o número de projecteis usados e se for descoberto que esse número é 12 ou inferior, o programa passa para a linha seguinte. Se os 12 projecteis foram usados, a mensagem FIM DO JOGO*** é assinalada intermitentemente no écran numa variedade de cores, à medida que um ciclo de som gera um fundo sonoro para essa piscadela colorida. Após esta apresentação, de

tirar o fôlego a qualquer um, começará um novo jogo.

Se não se ultrapassou o valor máximo de Q, é gerada uma nova posição inicial para o projectil, imprime-se a linha inferior da moldura toda de novo para eliminar a antiga posição da base móvel, e finalmente a direcção do projectil (quer seja para a direita e para cima ou para a esquerda e para cima) é determinada pelo valor dado a M na linha 185 e relacionado com B na linha 200. Agora dirige-se a atenção para a linha 60, onde a dança recomeça. As duas linhas finais do programa — 210 e 215 — contêm os dados para os tijolos e para o projectil, respectivamente.



```

10 REM DEMOLICAO
15 BORDER 2: PAPER 2: INK 7: C
L8
20 PRINT INK 4: "
25 LET a=1
30 FOR n=1 TO 20: PRINT AT n,1
: INK 4: "

```



```

35 LET s=0: LET a=1: LET b=1
40 FOR n=USR "A" TO USR "A"+5:
READ K: POKE n,K: NEXT n
42 FOR n=USR "B" TO USR "B"+5:
READ K: POKE n,K: NEXT n
45 FOR n=2 TO 8
50 LET z=INT (RND*8): IF z=2 THEN GO TO 50
55 PRINT AT n,2: INK z: BRIGHT
1: "
": NEXT n
58 PRINT AT 0,20: INVERSE 1;"B
OLA: ";1
60 LET n=15: LET x=20: LET y=I
NT (RND*10)+5: LET dx=x: LET dy=
y
65 IF y>28 OR y<3 THEN GO SUB
150
70 IF x<2 THEN LET a=-a
75 IF SCREEN$(x,y)<>" " THEN
GO SUB 125
80 IF x>20 AND (y=n+2 OR y=n+1
OR y=n+3) THEN LET a=-a: BEEP .
008,10: LET y=y+1
85 PRINT AT dx,dy;" ";AT x,y;"
♦": LET dx=x: LET dy=y
90 PRINT AT 21,n;"
95 LET n=n+(INKEY$="8" AND n<=
28)-(INKEY$="5" AND n>0)
105 LET x=x-a: LET y=y+b
110 IF x>21 THEN GO TO 165
115 PRINT AT 0,4: INVERSE 1;"PO
NTUACAO="s
120 GO TO 65
125 BEEP .008,20
130 PRINT AT dx,dy;" ";AT x,y;"
♦"
135 LET a=-1
140 LET s=s+1
145 RETURN
150 BEEP .008,30
155 LET b=1-2*(y>28 OR y<2)
160 RETURN
165 LET q=q+1
170 BEEP 1,0: BEEP .02,20
175 IF q>12 THEN FOR z=1 TO 10:
PRINT AT 5,8: PAPER RND*8: INK
8: FLASH 1:"***FIM DO JOGO***":
FOR d=0 TO 30 STEP 3: BEEP .008,
d: BEEP .008,-(d): NEXT d: NEXT
z: RUN

```

```

180 PRINT AT 0,20: INVERSE 1;"B
OLA: ";q
185 LET m=RND
190 PRINT AT 21,0;"
195 LET a=1
200 LET b=-b*(m<.5)+b*(m>=.5)
205 GO TO 60
210 DATA BIN 1110111,BIN 111101
11,BIN 11110111,0,BIN 11111110,B
IN 11111110,BIN 11111110,0
215 DATA BIN 1000,BIN 11100,BIN
111110,BIN 111110,BIN 11100,BIN
1000,0
1700 PRINT AT 0,20: INVERSE 1;"B
OLA: ";q: INVERSE 0

```

UDG- ■=A ♦=B

SIMULADOR DE CONDUÇÃO A 3 DIMENSÕES

Este pequeno programa é de grande efeito e uma vez apanhado o jeito verifica-se que é bastante competitivo. O jogador está a guiar por uma estrada criada por duas linhas (desenhadas usando a instrução DRAW) que convergem no horizonte. Usando as teclas "5" (para mover para a direita) e "8" (para mover para a esquerda) tenta-se manter o veículo dentro da estrada. A tecla "I" usa-se para alterar a velocidade.

O programa começa por definir um certo número de variáveis (O, P, IS, Q, D e R) e colocar o fundo e o enquadramento a preto e a cor dos caracteres a branco.

A linha 35 lê o teclado para "5" e "8" multiplicando o resultado da leitura pela velocidade (variável I). Na linha 40, P adiciona-se a D (definido inicialmente como sendo zero na linha 25) e usa-se para preparar as coordenadas para as linhas desenhadas na sub-rotina 100 e que representam as bermas da estrada. A variável O corresponde à posição horizontal do condutor no écran (ver linha 70) e calcula-se a partir de P na linha 45. A velocidade é alterada aleatoriamente na linha 50, e se P está situada entre 120 e 140, limpa-se o écran na linha 55,

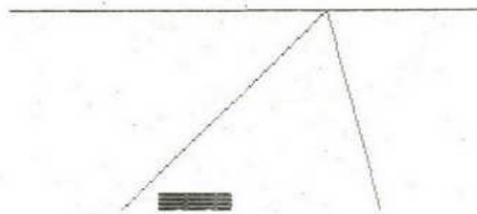
antes de se reimprimir a estrada pela sub-rotina da linha 100. Embora o *écran* seja limpo uma e outra vez para que a estrada seja reimpressa, a instrução DRAW trabalha tão depressa que a "nova estrada" aparece quase instantaneamente, e o "piscar" que se observa mantém-se a um nível mínimo.

A linha 65 verifica se houve toques em algum dos lados da estrada; se isso sucedeu, manda o programa para a rotina da linha 150. O veículo reimprime-se na linha 70 e a "distância percorrida" (variável S) é incrementada por um valor relacionado com a sua velocidade, antes, porém, da reimpressão no *écran*.

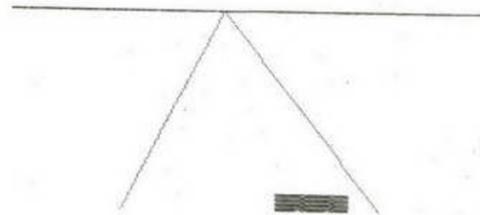
A linha 77 permite a mudança de velocidade (se D for igual a zero) e a linha 80 imprime um traço através do topo do *écran* que funciona como uma espécie de velocímetro. A linha 90 manda a acção de volta à linha 35 para os próximos segundos da condução. A sub-rotina da linha 100 a 140 usa instruções PLOT e DRAW para colocar a estrada no *écran*.

Querendo melhorar o programa, pode mudar-se a linha 70 para cinco gráficos definidos pelo utilizador para representar um pára-brisas e um volante.

Velocidade 70 km/h
0,39 kilometros



Velocidade 100 km/h
2,01 kilometros



```

1>REM CONDUCAO EM 3D
3 INK 7: BORDER 0: PAPER 0: C
LS
5 LET o=15: LET s=0
10 LET p=120
12 LET i=0
15 LET d=4
20 LET a=50
25 LET d=0
30 LET r=200
35 LET p=p-(i*2)*((INKEY$="8")
-(INKEY$="5"))
40 LET p=p+d
45 LET o=(255-p)/8
50 IF p>120 AND p<140 THEN LET
i=i+1: LET d=i*(INT (RND*2)-INT
(RND*2))
55 CLS
60 GO SUB 100
65 IF ATTR (21,0+4)=5 OR ATTR
(21,0+1)=5 THEN GO TO 150
70 PRINT AT 21,0," "
75 LET s=s+i/100: PRINT AT 3,0
;s:" kilometros "
77 IF INKEY$="1" AND d=0 THEN

```



```

90 FOR b=1 TO 12
100 PRINT AT a+b,t;a$(INT ((b/1
3)+.5))
105 BEEP 1/s;b
107 LET s=s+1: PRINT AT 0,0;"Po
ntuacao: ";s
110 PRINT AT a+b,0;"
"
120 LET t=t-((INKEY$="8")*(t>0)
-(INKEY$="5")*(t<21))*(INT (b/4)
)
130 NEXT b
140 IF t>14 OR t<9 THEN GO TO 2
00
150 LET i=-i
170 IF i=-1 THEN PAPER 0: BORDE
R 0: INK 8
180 IF i=1 THEN PAPER INT (RND*
7)+1: BORDER 4: INK 0
190 CLS
195 GO TO 80
200 FOR a=1 TO 20
205 LET n=INT (RND*8): IF n=2 T
HEN GO TO 205
210 PAPER n: CLS
220 NEXT a
225 PAPER 6: INK 0: CLS
230 PRINT AT 11,9;"Pontuacao: "
;s
235 PRINT AT 18,8;"Prima uma te
cla "
240 PAUSE 0
250 RUN

```

CORRIDA DA MORTE

Será o jogador capaz de iludir um rastejante e horrendo monstro verde e passar a CORRIDA DA MORTE antes de ser apanhado? Este programa ajuda a responder à pergunta.

O programa cria no *écran* um labirinto rectangular, estando o jogador colocado na esquerda e o mencionado monstro algures no labirinto. Usando as habituais teclas "5" e "8" para se mover para a esquerda e para a direita, e "W" e "Z" para se mover para cima e para baixo, terá de se manipular o homenzinho (a figura no fim da linha 240) do lado esquerdo para a extrema direita do labirinto. Na linha 5 encontra-se o gerador de números aleatórios, e nas linhas 10 a 30 são colocados o

enquadramento e o papel a preto os caracteres a amarelo. A linha 40 manda a acção para a sub-rotina da inicialização da linha 500, onde ficam os dados para os dois gráficos definidos pelo utilizador (o homenzinho e o monstro rastejante). Definidos os gráficos, o programa volta à linha 50 onde se define uma função que gera números aleatórios. Na linha 55 limpa-se o *écran*, desencadeando a definição da cor do papel na linha 30, e inicializando duas variáveis X (o número a usar na função A para produzir intervalos aleatórios nas paredes do labirinto nas linhas 160 e 170) e S (a pontuação inicial, diminuída em 16 unidades na linha 245 cada vez que se passa pelo ciclo principal).

A linha 100 imprime uma barra amarela ao longo do topo do *écran*, e depois o ciclo das linhas 110 a 130 imprime barras no tom azul-pálido na parte inferior (linha 120), e o lado direito da moldura, que limita o labirinto, em amarelo (linha 125). A linha 140 completa o material em bruto para o labirinto com uma barra ao longo do fundo do *écran*.

A função A aparece agora sozinha a imprimir espaços em branco aleatoriamente nas sólidas paredes do labirinto. Estes são os espaços que têm de usar-se (e que o monstro usa) na arripante corrida em direcção ao lado direito. As linhas 190 a 220 atribuem valores a U (a coordenada horizontal da posição inicial do homenzinho), a T (a coordenada vertical inicial do monstro rastejante), a A (a coordenada horizontal inicial do monstro) e P (a que se dá valor 1, uma vez que o homenzinho deve partir do lado esquerdo da CORRIDA DA MORTE).

A linha 235 verifica se o monstro verde e o homenzinho estão a ocupar o mesmo local. Em caso afirmativo, o monstro atacou, e o programa muda para a linha 400, onde, após um curto excerto do "Hino da Vitória dos Monstrozinhos Verdes", se recebe a informação de que se perdeu a Corrida da Morte. O resto do ciclo principal de 230 a 315 trata de ler o teclado.

"Ganhou a Corrida da Morte" são as alegres notícias da linha 360. Na linha 365 verifica-se se atingiu ou bateu a pontuação mais elevada, igualando a variável correspondente (H) à da pontuação presente (S) se necessário, e as linhas 370 e 375 imprimem, respectivamente, o resultado actual e o resultado

mais elevado. Note-se que a linha 370 usa o comprimento da versão do vector literal da pontuação (instrução LEN STR\$ s) para posicionar a impressão.

As linhas 380 e 390 esperam que retire as mãos do teclado (380), aguardando até que pressione uma tecla (390) para começar nova corrida (a partir da linha 55). Quando se estiver realmente confiante na perícia para manobrar na CORRIDA DA MORTE, pode modificar-se o programa por forma a conter mais dois ou três monstrosinhos.

```

1>REM CORRIDA DA MORTE
5 RANDOMIZE
7 LET h=0
10 BORDER 0
20 INK 5
30 PAPER 0
40 GO SUB 500
50 DEF FN a(x)=INT (RND*x)+1
55 CLS
60 LET x=20
70 LET s=10000
80 REM W move para cima
90 REM Z move para baixo
100 PRINT INVERSE 1;"
110 FOR a=0 TO 20
120 PRINT INK 5;" ■■■■■■■■
125 PRINT AT a,30; INVERSE 1;"
130 NEXT a
140 PRINT AT 21,0; INVERSE 1;"
150 FOR a=2 TO 28 STEP 2
160 PRINT AT FN a(x),a;" "
170 PRINT AT FN a(x),a;" "
180 NEXT a
190 LET u=FN a(x)
200 LET a=FN a(x)
210 LET p=1
220 LET t=FN a(15)*2-1
230 PRINT AT INT a,t; INK 4;"X"
235 IF ATTR (u,p)=4 THEN GO TO 400
240 PRINT AT u,p;"^"
245 LET s=s-16
250 BEEP .03,s/200
255 PRINT AT 0,1; INVERSE 1;"PO

```

```

NTUACAO ";s;" "
260 PRINT AT u,p;" "
265 IF INKEY$="8" AND ATTR (u,p
+1)=5 THEN LET p=p+2: BEEP .1,p:
BEEP .1,p-12:
270 IF INKEY$="5" AND ATTR (u,p
-1)=5 THEN LET p=p-2: LET s=s-10
0: BEEP .1,0
275 IF p=29 THEN GO TO 320
280 LET u=u+(INKEY$="Z")*(u<20)
-(INKEY$="W")*(u>1)
285 PRINT AT INT a,t;" "
290 LET a=a+RND*(INT a<u)-RND*(
INT a>u)
295 IF ATTR (INT a,t+1)=5 AND t
<p THEN LET t=t+2
300 IF ATTR (INT a,t-1)=5 AND t
>p THEN LET t=t-2
315 GO TO 230
320 FOR a=30 TO 60
330 BEEP .01,a
340 BEEP .01,a-12
350 NEXT a
360 PRINT AT u,5; OVER 1;"Ganho
u a Corrida da Morte"
365 IF h<s THEN LET h=s
370 PRINT AT u+1-2*(u=20),30-LE
N STR$ s;s
375 PRINT AT 0,1; FLASH (h=s);
INVERSE 1;"MELHOR PONTUACAO ";h
380 IF INKEY$="" THEN GO TO 385
385 IF INKEY$="" THEN GO TO 385
390 GO TO 55
400 FOR f=60 TO 30 STEP -1
410 BEEP .01,f
420 BEEP .01,f-12
430 NEXT f
445 PRINT AT u+1-2*(u=20),5; OV
ER 1;"Perdeu a Corrida da Morte"
450 PRINT AT INT a,t; FLASH 1;
INK 4;"X"
460 GO TO 375
500 DATA 24,24,48,94,16,104,76,
0
510 DATA 0,34,84,12,12,84,34,0
520 FOR a=0 TO 1
530 FOR b=0 TO 7
540 READ c
550 POKE USR CHR$ (144+a)+b,c
560 NEXT b
570 NEXT a

```


AS RÉPLICAS LOUCAS

Um único zombie ("morto vivo" ou "alma do outro mundo") multiplicou-se 19 vezes e daí resultaram 20 zombizinhos igualmente malévolos e que querem desesperadamente apanhar o jogador. Este encontra-se encurralado numa área rectangular da Zombielândia, onde as principais características topográficas (para além dos zombies e da pobre e indefesa pessoa do jogador) são evidenciadas por uma quantidade de profundos buracos. O zombie original pode ser identificado por duas características bem definidas: um apuradíssimo dispositivo de rastreio de humanos nele existente, e um ódio pela raça humana que chega a roçar a fronteira da paranóia.

Como é óbvio, as RÉPLICAS são idênticas ao seu antecessor sob todos os aspectos, por isso os 20 pequenos loucos da Terra dos Zombies querem o sangue do intruso. Além de herdarem as qualidades do pai, os filhos-réplica também herdaram as suas fraquezas. A principal fraqueza do zombie número um é uma visão extremamente deficiente, o que poderá ser a chave para a sobrevivência do jogador.

Recordemos que há um certo número de buracos na zona da Terra dos Zombies.

Os dispositivos de rastreio humano dos zombies enquanto bons para apanhar as emanções do *Homo sapiens* são piores do que inúteis quando se trata de detectar a presença dos buracos. Atrair um zombie para um buraco é o mesmo que ter menos um na perseguição.

A chave para a sobrevivência está em tomar vantagem da deficiente visão das criaturas, atraindo-as para os buracos por forma a haver sempre um buraco entre o homem e o zombie. No entanto, tem de lutar-se contra um certo número de horrorzinhos, e enquanto se está a tentar atrair um deles para um buraco, talvez outro ou outros lhe saltem em cima.

Há movimentos para cima, para baixo, para a direita ou para a

esquerda, ou — caso o homenzinho se sinta particularmente corajoso — pode ficar parado de tempos a tempos.

Os zombies, por seu lado, movem-se nos sentidos vertical, horizontal e diagonal. Inicia-se cada jogo de RÉPLICAS LOUCAS com quatro vidas e um sortido completo de 20 zombies. Há um prémio de um ponto por cada zombie que seja atraído para um buraco, e seis pontos de bónus se se conseguir livrar todo o perímetro da terrível ameaça.

Quando se sobrevive, a Terra das Réplicas será novamente desenhada mas com menor número de buracos. Caindo-se num buraco, sendo comido por um zombie ou ultrapassando os limites do terreno, perde-se uma vida. Quatro erros tolos como estes e haverá um homem morto.

O programa assinala a pontuação mais elevada; talvez se passem cinco anos antes de conseguir melhorar os resultados. O homenzinho move-se no *écran* usando as teclas "A" (cima), "S" (baixo), "K" (esquerda) e "L" (direita). Qualquer outra tecla pára-lhe o movimento.

Após a colocação do gerador de números aleatórios na linha 1, dimensiona-se uma matriz para guardar as réplicas. A variável RAND é igualada a 0.1 (linha 15) e então a acção transfere-se para a sub-rotina que começa na linha 9000. Preparam-se os gráficos dos buracos, dos zombies e da figura humana e, na linha 9100, coloca-se a moldura a azul, o fundo a amarelo e a cor dos caracteres a preto.

Aparece um "título de página" seguido de um lembrete da função de cada tecla. A mensagem "Prima uma tecla para começar" (linha 9200) antecede a mensagem de PAUSE 0, que segura o programa até se premir uma tecla.

A mensagem RETURN da linha 9900 devolve a atenção à linha 20, onde as vidas disponíveis são inicializadas a quatro unidades (o nome da variável é *men*, uma unidade inferior às vidas que sobrem; por isso quando *men* for igual a zero ainda há uma vida. O resultado disponível é inicializado a zero, e as outras variáveis que em breve serão necessárias inicializam-se na linha 30.

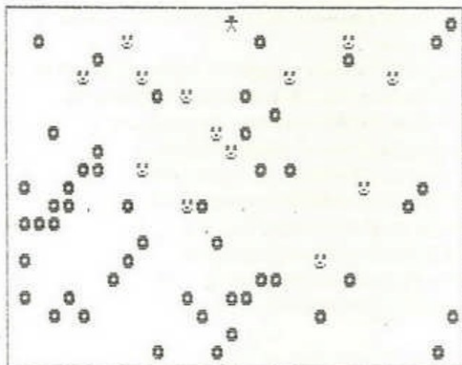
O endereço 23560 sofre um POKE, sendo lá colocado zero.

Este endereço armazena o valor da tecla recentemente pressionada, e fazer o POKE com o valor zero tem o efeito de limpar o valor da tecla que lá estava guardado. A linha 100 manda o programa para a sub-rotina 1000, onde a Terra dos Zombies é desenhada.

A moldura é desenhada na linha 101, e então o ciclo duplo ('I' e 'J') da linha 102 coloca os buracos nos lugares. O número de buracos depende do valor da variável RAND na linha 104, e também do número gerado pela instrução RND da mesma linha.

Após o regresso desta sub-rotina, a linha 110 dirige a atenção para a sub-rotina da linha 3000 que coloca os zombies no écran, contando-os (princípio da linha 3050) e seguindo-lhes o rasto no âmbito da matriz Z (duas últimas partes da linha 3050). Da sub-rotina 3000 iremos, através da linha 120, para a sub-rotina 2000, que coloca o homem no terreno. A elaborada série de instruções do tipo "IF SCREEN\$..." verifica se a área que circunda a posição de partida está de facto livre. Se não estiver, poderá ser assaltado por uma réplica ou "tropeçar" num buraco, mesmo antes de o jogo começar.

Pontuação: 0 M.P.: 0 六六六



Examinando este "retrato" do jogo, vê-se que o número de vidas que sobram é indicado pelos homenzinhos que se encontram no canto superior direito acima do terreno. As linhas 130 a 150 colocam estes homenzinhos no lugar, a linha 160 imprime o número de pontos no canto superior esquerdo do écran, e o melhor resultado (MR) é impresso, pela linha 170, no meio do topo do écran.

Recordemos que na linha 40 foi armazenado um zero no endereço 23560 (através duma instrução POKE) por forma a limpar a posição de memória correspondente. Agora, o computador ficará à espera que toque numa das quatro teclas ('A', 'S', 'K' ou 'L') e, caso o faça, uma "espreditada" (instrução PEEK) na posição de memória correspondente ao endereço 23560 (como acontece na linha 200) indicará qual das teclas está a ser carregada. O computador, antes que actue nessa informação, através das linhas 210 a 240, define ainda duas variáveis (TX e TY) idênticas à sua presente posição. Usam-se estas variáveis, se necessário, para "apagar" a figura na linha 450. Se não se premiu qualquer das quatro teclas, a linha 250 leva a cabo a rotina que verifica a sua posição, e que a reimprime. A linha 260 leva a cabo um rápido exame para averiguar se, por distração, não se saiu dos limites do mundo Zombie, e a linha 270 vê se o local para onde o jogador deseja ir não estará ocupado. Caso encontre alguma coisa nesse local, é o fim dele, e a rotina que começa na linha 4000 exibe uma dramática apresentação de "som e luz" para assinalar o seu desacerto. Mas passando no teste da linha 270, a figura humana será reimpressa na linha 280.

Atravessa-se a rotina das linhas 300 a 490 até que o valor da variável NUM (incrementada na linha 400) se torne maior do que o valor de GO (que é originalmente igual a quatro na linha 30, e depois alterado na linha 485).

Toda esta rotina contém a inteligência malévola dos membros da família Réplica. Uma rotina associada, linhas 500 a 530, é chamada através da linha 410, caso o teste executado nessa linha pela função SCREEN\$ encontre um espaço em branco. Se o número de zombies que sobra (a variável left guarda esta

informação) baixar a zero (ou seja, se o jogador destruiu todas as réplicas — bem como o original —, o que é confirmado pela linha 470) obtém-se um bônus (vejam-se as linhas 8000 a 8050) e diminui o valor da variável RAND (o que significa que vão encontrar-se menos buracos, objectivo esse determinado pela linha 1040).

Uma vez terminada a rotina zombie, a linha 305 manda a acção de volta à linha 200 e o processo recomeça. O ciclo da linha 7000 faz a impressão dos zombies, completada com um pequeno acompanhamento musical (linhas 7020 e 7040).

Reduzido a zero o número de homens (veja-se a linha 7060), o computador usa a rotina da linha 7070 para, em caso de necessidade, alterar o melhor resultado (MR), e ainda para oferecer a oportunidade de se iniciar novo jogo.

Se se introduzir um "N" para indicar que não se quer começar novo jogo, a linha 7090 apronta a cor, a orla e o fundo para facilitar a listagem do programa ou para a introdução de um novo jogo. Querendo um novo jogo (e a tecla "S" é detectada na linha 7080), a acção vai para a linha 25, que assegura que o melhor resultado obtido (variável "HI") não é repostado a zero.

Entre as modificações que podem introduzir-se no programa inclui-se a alteração das teclas que fazem mover a figura humana. Para o fazer mudam-se as linhas 210 a 240, onde se usa o valor numérico da tecla introduzida (recordemos o endereço 23560). Ao fazê-lo ter-se-á igualmente de modificar as instruções nas linhas 9150 a 9180. Para produzir mais ou menos buracos, muda-se o valor inicial da variável RAND nas linhas 15 e 7063. Para operar a mudança no número de buracos no fim de cada fase do mesmo jogo, altera-se a linha 8006. Os gráficos definidos pelo utilizador são "A" (para o homem), "B" (para o zombie e sua réplica) e "C" (para os buracos.).

```

1>REM AS REPLICAS LOUCAS
2>RANDOMIZE
10 DIM z(30,2)
11 LET hi=0
15 LET rand=0.1
20 GO SUB 9000

```

```

25 LET men=0: LET score=0
30 LET la=0: LET max=20: LET
zombie=0: LET go=4
40 DOOK=0: LET go=0
100 GOOK=0: LET go=0
110 GOOK=0: LET go=0
120 GOOK=0: LET go=0
130 IF men=0 THEN PRINT AT 0,29
;
140 IF men>1 THEN PRINT AT 0,30
;
150 IF men>0 THEN PRINT AT 0,31
;
160 PRINT AT 0,0;"Pontuacao: ";
score
170 PRINT AT 0,15;"M.P.: ";hi
200 LET a=PEEK 23550
205 LET tx=px: LET ty=py
210 IF a=9 THEN PRINT AT py,px
;
220 IF a=115 THEN GO TO 260
X: LET py=py+1: GO TO 260
230 IF a=107 THEN PRINT AT py,px
X: LET px=px-1: GO TO 260
240 IF a=108 THEN PRINT AT py,px
X: LET px=px+1: GO TO 260
250 GO TO 300
260 IF px<1 OR px>30 OR py<2 OR
py>20 THEN GO TO 4000
270 IF SCREEN$(py,px)<>" " THEN
N GO TO 5000
280 PRINT AT py,px;"A"
300 LET num=0
305 IF num>90 THEN GO TO 200
310 LET zombie=zombie+1: IF zombie>max THEN LET zombie=1
320 IF z(zombie,1)=0 THEN GO TO
330 LET zx=z(zombie,1): LET zy=
z(zombie,2): LET tx=zx: LET ty=zy
340 LET dx=0: LET dy=0
350 IF tx<px THEN LET dx=1
360 IF tx>px THEN LET dx=-1
370 IF ty<py THEN LET dy=1
380 IF ty>py THEN LET dy=-1
390 LET zx=zx+dx: LET zy=zy+dy
400 LET num=num+1
410 IF SCREEN$(zy,zx)=" " THEN
GO TO 5000
420 IF SCREEN$(zy,zx)="B" THEN
GO TO 305

```



```

430 IF ZX=PX AND ZY=PY THEN GO
TO 7000
440 LET Z(Zombie,1)=0
450 PRINT AT ty,tX;" "
460 LET score=score+1: PRINT AT
0,0;"Pontuacao: ";score
470 LET left=left-1: IF left=0
THEN GO TO 3000
480 FOR n=1 TO 10: BEEP 0.01,n:
NEXT n
485 LET go=1+INT (left/5)
490 GO TO 305
500 PRINT AT ty,tX;" "
510 PRINT AT zy,zx; FLASH 1; IN
K 2;" "
520 LET z(zombie,1)=zx: LET z(z
ombie,2)=zy
525 BEEP 0.02,30-left
530 GO TO 305
1000 PAPER 6: CLS
1010 PLOT 4,4: DRAW 248,0: DRAW
0,159: DRAW -248,0: DRAW 0,-159
1020 FOR i=2 TO 20
1030 FOR j=1 TO 30
1040 IF RND>rand THEN GO TO 1050
1050 PRINT AT i,j; INK 1;"0"
1060 NEXT j: NEXT i
1100 RETURN
2000 LET PX=INT (RND*13+5)
2010 LET PY=INT (RND*17+5)
2020 IF SCREEN$(PY,PX)<>" " THE
N GO TO 2000
2021 IF SCREEN$(PY-1,PX-1)<>" "
THEN GO TO 2000
2022 IF SCREEN$(PY,PX-1)<>" " T
HEN GO TO 2000
2023 IF SCREEN$(PY+1,PX-1)<>" "
THEN GO TO 2000
2024 IF SCREEN$(PY-1,PX)<>" " T
HEN GO TO 2000
2025 IF SCREEN$(PY+1,PX)<>" " T
HEN GO TO 2000
2026 IF SCREEN$(PY-1,PX+1)<>" "
THEN GO TO 2000
2027 IF SCREEN$(PY,PX+1)<>" " T
HEN GO TO 2000
2028 IF SCREEN$(PY+1,PX+1)<>" "
THEN GO TO 2000
2030 PRINT AT PY,PX;"*"
2100 RETURN
3000 FOR j=2 TO 20
3020 LET i=INT (RND*30+1)

```

```

3030 IF SCREEN$(j,i)<>" " THEN
GO TO 3020
3040 PRINT AT j,i; FLASH 1; INK
2;"0"
3050 LET left=left+1: LET z(left
,1)=i: LET z(left,2)=j
3060 NEXT j
3100 RETURN
4000 FOR i=1 TO 20
4010 PRINT AT PY,PX;" ": PRINT A
T ty,tX;"*"
4020 BEEP 0.02,i
4030 PRINT AT PY,PX;"*": PRINT A
T ty,tX;" "
4040 BEEP 0.02,20-i
4050 NEXT i
4060 GO TO 7050
5000 FOR i=1 TO 20
5020 PRINT AT PY,PX;" ": PRINT A
T ty,tX;"*"
5030 BEEP 0.02,20+i
5040 PRINT AT PY,PX;"*": PRINT A
T ty,tX;" "
5050 BEEP 0.02,20-i
5060 NEXT i
5070 GO TO 7050
7000 FOR i=1 TO 20
7010 PRINT AT PY,PX; INK 3;"3":
PRINT AT ty,tX;" "
7020 BEEP 0.02,20+i
7030 PRINT AT PY,PX;"*";AT ty,tX
;"3"
7040 BEEP 0.02,20-i
7050 NEXT i
7060 LET men=men-1: IF men<0 THE
N GO TO 7070
7062 FOR i=1 TO 100: NEXT i
7063 LET rand=0.1
7065 GO TO 30
7070 IF score>hi THEN LET hi=sco
re: PRINT AT 0,15;"HI: ";hi
7075 PRINT AT 1,8;"Outro Jogo(s/
n)?"
7080 IF INKEY$="s" THEN GO TO 25
7090 IF INKEY$="n" THEN BORDER 7
: PAPER 7: INK 0: 3TOP
7100 GO TO 7080
8000 PRINT AT 0,10; FLASH 1;"***
BONUS ***"
8005 LET score=score+6
8006 LET rand=rand*0.6
8010 FOR i=1 TO 3

```


quadrados a contar do canto são provavelmente os mais fracos.

Outros pontos fortes incluem as redondezas dos quatro quadrados em que se começa o jogo.

O *Spectrum* tem um conhecimento muito bom do valor relativo das várias posições do tabuleiro. Não é a melhor das estratégias a de depender apenas da posição. No entanto, o conhecimento das principais posições tem muito maior peso do que qualquer outra consideração e o computador pode rapidamente escolher a melhor jogada possível do rol das jogadas legais disponíveis. Portanto, este programa conta apenas com o conhecimento das posições fortes e, por isso, oferece uma rápida e, na maior parte das vezes, inteligente réplica a qualquer jogada que se faça, dando origem a um jogo rápido e competitivo.

A linha 30 manda a acção para a sub-rotina de 4000, que imprime o nome do programa, esperando que uma tecla seja premida para o jogo ter início. Nessa linha, coloca-se o enquadramento a 6 (amarelo), o fundo a 2 (vermelho) e a cor dos caracteres a 9 (uma cor de tinta colocada a 9 selecciona automaticamente a cor mais contrastante possível com a cor do fundo). A palavra "PIRANDELLO" é impressa numa variedade de cores ao longo do *écran*, de cima até ao fundo, completada por um acompanhamento musical. A palavra "AGUARDE" aparece, piscando, perto do topo do *écran*, enquanto os vectores são DIMensionados e as variáveis inicializadas.

A rotina de inicialização começa na linha 5000. DIMensionam-se três vectores. O vector A (linha 5010) guarda as posições no tabuleiro, distribuídas por ordem de importância. O vector D será usado, durante o correr do jogo, para guardar as localizações das peças que devem ser mudadas de um jogador para o outro.

Os ciclos de 5030 a 5070 preenchem o vector A com caracteres 146 (ou seja, com o código — CODE — dum quadrado definido pelo utilizador; ver apêndice A do manual do *Spectrum*). Estes serão usados quando da impressão do tabuleiro, para indicar as posições desocupadas. O próximo ciclo (5080 a 5110) lê a informação das instruções DATA (dados) situadas em 5320 e 5330 para a introduzir no vector E. Reparando nas duas instruções DATA, vê-se que contêm números de 12 a 89 numa

ordem aparentemente aleatória. Esta ordem não é de facto aleatória, apesar das aparências. Em vez disso, representa a ideia do programador quanto ao valor relativo das posições no tabuleiro. O quadrado 19, por exemplo, é um dos mais valorizados, por isso o computador irá procurar sempre uma jogada possível para esse quadrado antes de ir para qualquer outro lado. Os quadrados de menor valor são os que se encontram perto do fim da instrução DATA de 5330; são eles 28, 78 e 23.

O próximo ciclo (5120 a 5150) preenche o vector D com os números na instrução DATA da linha 5340 (1, 9, 10, 11, -1, -9, -10, -11). Estes representam os "deslocamentos" de uma dada posição no tabuleiro e usam-se quando, após uma jogada, o computador verifica se a posição das peças necessita de ser alterada.

As linhas 5160 e 5170 colocam em posição as peças de abertura. Nessa altura um marcador de jogadas, que indica qual a vez de cada jogador, é inicializado a 145 na linha 5180 por forma a mostrar que o computador joga primeiro.

Feito isto, o *Spectrum* começa o trabalho de definir os gráficos.

São usados três tipos: um quadrado para a posição vazia, um losango (impresso a vermelho) para uma peça do computador, e uma linha diagonal (impressa a branco) para uma peça do jogador. Os dados necessários para estes gráficos são guardados nas linhas 5350 a 5370, com o Z no fim da linha 5370 a "disparar" a mudança de cor da moldura, a limpeza do *écran* (CLS) e o regresso à linha 400, isto tudo na linha 5270.

Após a impressão do tabuleiro (linhas 400 a 490) verifica-se o marcador de jogadas.

Se tiver o valor 145, o computador sabe que está na altura de ser ele a jogar. Caso contrário, o computador muda Y para 144 (para que ele saiba que na próxima verificação de Y é a vez de jogar), e a linha 530 aceita o lance do jogador como sendo dois algarismos, o da linha (horizontal), seguido pelo da coluna (vertical). A jogada é introduzida como um único número de dois dígitos (como 25 ou 81). Serão rejeitadas as jogadas abaixo de 11 e acima de 88.

Verificando-se que não se pode mexer no tabuleiro, introduz-se um 0 e o computador será conduzido à linha 1000, onde fará a jogada. Caso seja o computador a não poder mexer-se (veja linha 350) ele irá verificar (linha 380) se o jogador indicou previamente a sua impossibilidade de jogar e, em caso afirmativo, manda a acção para o fim da rotina de jogo, a partir da linha 1000, onde se contam as peças e se anuncia o vencedor. A linha 560 assegura que a jogada está dentro dos limites numéricos (11 a 88) e que o quadrado para o qual se quer deslocar está em branco (ou seja, se contém o carácter 146).

Uma vez aceite a jogada como legal, o computador transfere a acção para a parte do programa que começa na linha 160, que usa os valores dos deslocamentos guardados no vector E para verificar se qualquer peça foi alterada (isto é, para verificar se houve trocas por peças do adversário). No fim da rotina, que decorre entre as linhas 160 e 320, verifica-se novamente o marcador de jogadas. Se se descobrir que marca 144 ou que a variável J de impressão do quadro (que foi inicializada na linha 160 e alterada quando necessário na linha 290) é igual a um, o computador vai para a linha 400, onde o tabuleiro é novamente impresso. Se nem todos os quadrados foram verificados (ou seja, K na linha 340 não é igual a 60), o computador vai para a linha 130 para continuar a sua pesquisa. Esta linha, obviamente, nunca será alcançada quando o jogador acabou de se mover. Se a linha 850 é alcançada, aparece a mensagem "Passo".

A linha 500 verifica o valor de Y, e se ele for 144 sabe-se que é a vez de o computador jogar. Portanto, o computador regressa à linha 100, onde procura uma jogada possível (linhas 100 a 150, mais 340). Note-se aqui que o computador usa a informação reunida das instruções DATA para determinar a ordem segundo a qual os quadrados devem ser verificados.

O PIRANDELLO é um jogo competitivo e o *Spectrum* um valoroso adversário. Uma vez que se tenha o jogo sob controle, podem alterar-se os gráficos definidos pelo utilizador e/ou experimentar mudar a ordem dos números contidos nas instruções DATA, na tentativa de conseguir melhorar a estratégia do computador.



```

10 REM PIRANDELLO
30 GO TO 4000
100 LET Y=145
110 LET Z=144
120 LET K=0
130 LET K=K+1
140 LET b=e(k)
150 IF a(b)<>146 THEN GO TO 340
160 LET J=0
170 FOR X=1 TO 8
180 LET n=d(x)
190 LET e=0
200 LET f=b
210 IF a(f+n)<>Y THEN GO TO 250
220 LET e=1
230 LET f=f+n
240 GO TO 210
250 IF a(f+n)<>Z THEN GO TO 310
260 IF e=0 THEN GO TO 310
270 FOR a=b TO f STEP n
280 LET a(a)=Z
290 LET J=1
300 NEXT a
310 NEXT X
320 IF Y=144 THEN GO TO 400
330 IF J=1 THEN GO TO 400
340 IF K<>60 THEN GO TO 130
350 IF a<>0 THEN PRINT AT 0,0;
FLASH 1; INK 2; PAPER 7; "Passo.";
BEEP 1,1; BEEP 1,2; BEEP 1,1
PRINT AT 0,0;
380 IF a=0 THEN GO TO 1000
400 PRINT AT 6,11; BRIGHT 1;"12
345678"
410 BEEP 0.05,32; BEEP 0.05,12;
BEEP 0.05,32
420 FOR a=1 TO 8
430 PRINT TAB 10; BRIGHT 1; INK
6;a;
440 FOR b=2 TO 9
445 LET w=a(a+10+b); BEEP 0.008

```



```

, (w=144)+50*(w=145)-50*(w=146)
450 LET r=2*(w=144)+7*(w=145)+4
*(w=146)
455 PRINT BRIGHT 1; INK R; CHR$
(w);
460 NEXT b
470 PRINT BRIGHT 1; INK 6;a
480 NEXT a
490 PRINT TAB 11; INK 6; BRIGHT
1;"12345678"
500 IF y=144 THEN GO TO 100
510 LET y=144
520 LET z=145
530 INPUT INK 2; PAPER 6; FLASH
1; BRIGHT 1;" Introduza a sua
Jogada "; g
540 IF g=0 THEN GO TO 100
550 LET b=g+1
560 IF b<12 OR b>89 OR a(b)<>14
6 THEN GO TO 530
590 GO TO 160
1000 LET c=0
1010 LET h=0
1020 FOR a=1 TO 100
1030 IF a(a)=144 THEN LET c=c+1
1040 IF a(a)=145 THEN LET h=h+1
1050 NEXT a
1055 PRINT AT 18,0; FLASH 1; BRI
GHT 1; INK 2; PAPER 7;"Resultado
s finais:";"SPECTRUM";c;"HUMA
NO";h
1060 IF c>h THEN PRINT AT 0,0; B
RIGHT 1; FLASH 1; INK 4;"Ganhei."
.. Talvez queira uma des-forra?"
: GO TO 1060
1070 IF c<h THEN PRINT AT 0,0; B
RIGHT 1; FLASH 1; INK 4;"Ganhou"
: GO TO 7000
1080 IF c=h THEN PRINT AT 0,0; B
RIGHT 1; FLASH 1; INK 4;"Empate"
os..."; GO TO 1060
1090 STOP
4000 BORDER 5; PAPER 2; INK 9; C
LS
4010 FOR a=1 TO 22; PRINT TAB a;
INK a/5+3;"PIRANDELLO"
4020 BEEP 0.008,a; BEEP 0.008,a*
2; NEXT a
4030 PRINT AT 15,3; FLASH 1; INK
7; PAPER 0;" AGUARDE "
5000 PAPER 0; INK 9
5010 DIM a(100); DIM e(50)
5020 DIM d(8)

```

```

5030 FOR a=1 TO 8
5040 FOR b=2 TO 9
5050 LET a(a*10+b)=146
5060 NEXT b
5070 NEXT a
5080 FOR a=1 TO 60
5090 READ b
5100 LET e(a)=b
5110 NEXT a
5120 FOR a=1 TO 8
5130 READ b
5140 LET d(a)=b
5150 NEXT a
5160 LET a(45)=144; LET a(46)=14
5170 LET a(55)=144; LET a(56)=14
5180 LET y=144
5190 READ a$; IF a$="Z" THEN BOR
DER 0; CLS : GO TO 400
5200 FOR c=0 TO 7
5210 READ b; POKE USA a$+c,b
5220 NEXT c
5230 GO TO 5270
5240 DATA 19,61,82,12,62,17,32,6
7,69,14,39,64,67,37,34,49,16,
52,85,16,62,42,66,66,64,66,44,36
1,67,36,47
5230 DATA 63,48,58,76,24,27,36,2
7,74,77,43,26,65,33,53,75,72,13,
29,16,88,22,83,79,73,28,76,23
5340 DATA 1,9,10,11,-1,-9,-10,-1
1
5350 DATA "A",0,24,60,126,126,60
2,24,0
5360 DATA "B",0,4,14,28,56,112,3
2,0
5370 DATA "C",255,129,129,129,12
9,129,129,255,"Z"
5000 PRINT AT 0,0; line: PAUSE 1;
RETURN
7000 PRINT AT 2,0; FOR i=1 TO 7
8: READ a; PRINT CHR$ a; BEEP .
05*(a<>32),14; NEXT i
7001 GO TO 7001
7005 DATA 32,79,32,71,97,98,105,
110,101,116,101,32,86,101,114,98
,111,32,100,101,32,73,110,102,11
1,114,109,97,116,105,99,97,102,1
01,108,105,99,105,116,97,45,111,
32,112,101,108,97,32,118,106,116
,111,114,106,97,32,97,108,99,97,

```

DAMAS

Desafiemos agora o *Spectrum* para as damas, o mais popular dos jogos de tabuleiro, depois do xadrez. Alguns estudiosos dos jogos acreditam que as damas foram de facto um antecessor do xadrez. Não se duvida que já se jogavam jogos deste tipo no antigo Egipto, na Grécia e em Roma. O jogo evoluiu através dos séculos até atingir a sua forma actual nos meados do século XVII.

Existem muitas variantes deste jogo, consoante os países ou regiões, mas, neste caso, o que o computador vai jogar é muito mais o jogo padrão do que uma variante exótica. No jogo padrão, as damas são jogadas entre dois adversários em casas alternadas do tabuleiro, geralmente nas pretas.

Para facilitar a visão no *écran* (e também em listagens impressas, caso se queira utilizar para tal a possibilidade introduzindo programa de copiar — COPY — o conteúdo do *écran* de tempos a tempos), jogaremos nas casas brancas.

Inicialmente, cada lado tem 12 peças. Nesta versão as peças do jogador são o "x" minúsculo e as do computador o "o" minúsculo (embora nada impeça de modificar o programa, uma vez corrido, usando quaisquer outros gráficos).

O computador começa no topo do *écran* jogando no sentido de cima para baixo, enquanto o adversário começa no fundo, jogando de baixo para cima. Só pode mover-se na diagonal, isto é, de e para casas brancas. Se na diagonal há uma peça inimiga logo à frente da peça, e uma casa vazia se encontra atrás dela, ainda na diagonal, pode capturar a peça saltando sobre ela (retirando-a do tabuleiro) por forma a aterrar na casa que se encontrava vaga. Pode ser capturada mais de uma peça de cada vez se, quando se "aterra" numa casa vaga após uma captura, for possível a partir desse ponto realizar nova captura.

Algumas regras das damas incluem uma disposição que

determina a obrigatoriedade de se efectuar uma captura sempre que esta seja possível. Neste jogo não existe tal obrigatoriedade, mas o computador fará capturas sempre que possa e, por isso, deveremos jogar com lisura e fazer igualmente uma captura quando ela puder ser feita.

O objectivo deste jogo é capturar todas as peças inimigas ou impossibilitar-lhes o movimento. Neste programa o *Spectrum* jogará até final, não deixando contudo de manter a possibilidade de julgar quando foi batido, atribuindo nessa altura a vitória ao adversário. Este poderá desistir em qualquer altura introduzindo 99 em vez da sua jogada. Ao fazê-lo o *Spectrum* aceitará amavelmente o desejo do adversário.

Uma vez que consiga colocar uma peça na última fila do tabuleiro, ela é "coroadada" e torna-se dama. Neste programa a dama apresenta um aspecto diferente, pela mudança dos caracteres que representam peças. Estes caracteres são facilmente alterados noutros do agrado do jogador como teremos oportunidade de explicar. Uma dama tem o dobro da mobilidade de uma peça vulgar. Isto é, tanto se pode movimentar para a frente como para trás, tornando-se, por isso, numa peça terrível. Note-se que, aterrando na última fila após uma captura, não se pode na altura converter a peça em dama e continuar a sequência de capturas para trás. A peça só pode funcionar como dama na jogada seguinte. O computador sabe isso e dá conta do facto esperando uma jogada antes de transformar a sua peça em dama.

Descobrir-se-á que o computador joga as damas bastante depressa e com um grau de inteligência e antevisão que no entanto deixa uma justa oportunidade ao adversário.

Para adicionar ainda mais interesse, o computador imprime cada jogada no *écran* (tal como ele a encara) fazendo soar um *beep* se encontrar um lance que valha a pena. O computador pesquisará através do tabuleiro, casa por casa do canto superior esquerdo ao longo do topo, depois pela fila seguinte da esquerda para a direita, e assim por diante até ao fim. Quando encontra uma das peças dele, o computador verifica primeiro a possibilidade de capturar uma peça inimiga. Em caso afirmativo, faz a captura sem causar muito alarido, reimprime o tabuleiro e

autoriza o adversário a introduzir a resposta à jogada. Quando joga, o computador procura fazer "lances seguros", usando a técnica simples de encontrar jogadas que não o exponham potencialmente a qualquer perigo. Tem também uma certa facilidade em seleccionar lances de protecção a peças que já se encontram em perigo.

Após ter pesquisado o tabuleiro e armazenado cada uma das jogadas seguras num vector verifica se com um lance apenas pode promover alguma das suas peças a dama. Se tal lance for possível, e porque o computador dá a esta situação prioridade mais elevada do que fazer apenas uma jogada segura, faz dama. Se não puder mover uma peça para a última fila para uma promoção, o computador escolhe ao acaso uma das jogadas que tem armazenadas num vector. Ao escolher aleatoriamente jogadas que ele julga serem de igual valor, o computador pode jogar uma série de partidas sem se repetir, mesmo que o adversário tente forçá-lo a uma partida idêntica a outra anterior.

Se a pesquisa do tabuleiro revelar a não existência de uma jogada segura, ele escolhe uma pedra ao acaso para mover. Tais lances ocorrem raramente e apenas na fase inicial e a meio do jogo. Se tiver apenas algumas peças no tabuleiro, pode considerar-se numa posição desesperada e conceder-lhe a vitória.

Como dissemos, este programa joga rapidamente, porque a maior parte das jogadas tem apenas de pesquisar o tabuleiro parcialmente ou, na pior hipótese, pesquisar todo o tabuleiro apenas uma vez. A velocidade de resposta é atrasada pelas linhas que imprimem os lances que o computador está a considerar. No entanto, elas dão um tal interesse ao jogo que, não obstante abrandarem a velocidade de execução, consideramos que vale a pena conservá-las. Caso se prefira um programa que corra à velocidade máxima (e é na verdade muito rápida), apagam-se algumas linhas, que indicaremos à medida que formos analisando o programa. Existem também alguns comentários, tais como: "Jogada segura", ou "Apanhei-te!" que aparecem de tempos a tempos. Estas coisas lentas abrandam ligeiramente a execução e podem muito bem ser apagadas. Muitas pessoas que usaram este programa acharam que afinal ambas as versões são

interessantes e, depois de terem introduzido e guardado em *cassette* o programa completo, apagam as linhas extra a fim de obterem uma segunda versão que trabalha à velocidade máxima. Assim pode escolher-se a versão que se quer jogar.

Após a localização do gerador de números aleatórios na linha 50, a acção vai para a sub-rotina que começa na linha 1070, onde as variáveis são inicializadas. Ao analisar esta secção ver-se-á que ela começa por colocar o fundo e a moldura a azul e a cor a amarelo antes de o *écran* ser limpo e poder aparecer a cor do fundo. Neste programa, tal como em todos os outros constantes deste livro, há inteira liberdade para mudar as cores de acordo com o gosto de cada um ou com as combinações que melhor se ajustem a cada televisor. A palavra "Aguarde" aparece perto do topo do *écran* durante o curto processo de inicialização. Pode considerar-se a hipótese de incluir esta palavra em qualquer programa que tenha um processo de inicialização mais longo do que o habitual, isto para dar a certeza ao utilizador de que realmente se está a passar alguma coisa após RUN e ENTER terem sido premidos.

A linha 1080 dá valores às pedras. Os nomes das variáveis são: C para as peças do computador; H para as do humano; CK para a dama do computador; HK para a dama do humano; E para uma casa vazia, e B para uma casa preta. São definidas nesta linha e a partir daqui em todo o programa as variáveis são conhecidas por estes nomes. Isto tem o objectivo de permitir que cada qual possa, à vontade, mudar os caracteres usados para representar as pedras, alterando simplesmente a linha. Se se desejar, por exemplo, acrescentar gráficos definidos por utilizador, usa-se 144 para "H" (144 é o CODE do gráfico definido pelo utilizador, disponível no modo "G" na tecla "A"), 145 para "C", e assim por diante.

A linha 1090 coloca a variável OF a representar "fora do tabuleiro".

O computador tem, "na cabeça", uma fila invisível de falsas casas, à volta do tabuleiro, por isso sabe onde lhe fica a fronteira e não tenta heroicamente capturar pedras em locais onde nunca existiram.

Uma vez definida OF, o vector Q (que guarda as peças) é DIMensionado e cada um dos seus elementos preenchido com o valor da falsa casa. O próximo ciclo (1110 a 1130) usa DATA nas linhas 1140 a 1230 para preencher as casas no tabuleiro com os seus devidos valores. Todos os elementos do vector Q não utilizados neste ciclo terão, obviamente, o valor de -99, para o computador saber que existem áreas que ele não deve invadir.

A linha 1240 é um ciclo que preenche o vector N (DIMensionado em 1090) com números que representam as jogadas possíveis de qualquer casa para um lance que não envolva captura. O computador sabe que duplicando esses números, em caso de necessidade, lhe dará uma jogada possível de captura. Na linha 1260 as variáveis que contam o número de peças capturadas ao adversário são inicializadas a zero, CO para o computador, HU para o humano.

Ao voltarmos de 1260 para 90 podemos ver, no comentário REM, as palavras "Apague a linha seguinte para o *Spectrum* não jogar primeiro". Se quisermos abrir o jogo, livremo-nos simplesmente da linha 100. Não necessitamos de alterar qualquer outra parte do programa. O *Spectrum* não é o melhor jogador de damas do Mundo, por isso dar-lhe a jogada de abertura é o mínimo que pode fazer. O programa é chamado dentro do ciclo das sub-rotinas 110 a 140, com a linha 150 a recomençar o processo. As várias sub-rotinas contêm as partes vitais do programa, respectivamente:

680 imprime o tabuleiro
850 aceita a jogada adversária
680 reimprime o tabuleiro após essa jogada
160 começa a pesquisa de jogadas do computador.

Vamos começar por analisar o processo de pesquisa do computador por ser a mais interessante e importante secção do programa.

A linha 160 DIMensiona o vector S, que será usado para

armazenar as jogadas seguras descobertas pelo computador enquanto procura capturas.

A variável SC, o "contador de jogadas seguras", é inicializado a zero na linha 170, onde A (usado para indicar o quadrado que se está a considerar), é colocado a 89, uma unidade mais do que o elemento 88, que contém a casa do canto superior esquerdo do tabuleiro. A linha 180 faz decrescer uma unidade à variável A, para indicar a casa do canto superior esquerdo, e a linha 190 verifica se esta casa contém uma peça vulgar do computador (ou seja, verifica se aquele elemento do vector contém o valor C) ou se se trata da dama (valor CK).

A linha 200 inicializa B (que indica qual a jogada potencial que está a ser considerada) a zero. Se A é menor do que 29, o computador sabe que se encontra na fila de trás do adversário humano e que apenas se pode mover para cima (valores de 3 e 4 para B indicam lances nessa direcção), por isso não se preocupa em considerar as jogadas correspondentes a B igual a 1 e a 2.

A linha 210 adiciona uma unidade a B (que então se torna 1 ou 3, se se está a considerar uma peça na fila de trás), e a 220 torna M (que ao longo do processo define a casa para a qual o computador intenta mover-se, enquanto A é a casa da qual o computador se moveu) igual a A mais o elemento do vector N indicado pelo valor de B. Se M é maior do que 88 ou menor do que 11, o computador sabe que a peça está fora do tabuleiro e nunca mais será considerada. A linha 240 é uma das tais que pode ser apagada se se quiser obter a velocidade máxima: esta linha imprime a jogada que está a ser considerada.

A nossa próxima linha descobre uma captura. Se a casa para a qual o computador tenciona mover-se (M) contém uma pedra do adversário (H) ou uma dama inimiga (HK) e a casa para além dessa está vazia (E), o computador sabe que pode capturar, dirigindo por isso a acção para a linha 320, onde se imprime, piscando no *écran*, a mensagem de júbilo "APANHEI-TE!" Também desta vez, querendo-se a velocidade máxima, devem-se apagar as linhas 330, 340 e 350, porque são usadas apenas para dar espectáculo.

De volta ao centro da secção de jogadas do computador, a

linha seguinte (260) conduz a simples pesquisa — que já mencionámos — olhando para a casa seguinte àquela para onde o computador se quer mover com o fim de verificar se, ao avançar, iria convidar a captura da peça movida, ou se — por mover essa pedra — expõe outra a algum perigo. Se uma pedra passa a longa série de testes (IF) desta linha, o *Spectrum* vai para a sub-rotina que começa na linha 470.

O computador consegue armazenar até dez jogadas seguras (uma vez que o vector que as guarda contém apenas dez elementos, e dez é um número mais do que suficiente na maior parte das circunstâncias). O contador de jogadas seguras, SC, que foi inicializado a zero na linha 170, é incrementado em uma unidade se for menor do que 10. A linha 480, outra que se pode apagar para obter velocidade máxima, imprime "Jogada segura... para..." e faz soar um *beep* para assinalar o lance.

A linha 490 é bastante interessante. Converte num único número as casas (A) donde se parte e (M) para onde se vai, número esse que se coloca no elemento disponível a seguir ao vector S, para que mais tarde, em caso de necessidade, possa ser decodificado. A linha 490 faz-nos regressar à secção principal do programa, mas, enquanto estamos neste ponto, vamos observar como se faz a jogada segura. Se não houve captura, o computador movimenta-se para a linha 500. Se encontra SC igual a zero, sabe que não foram encontradas jogadas seguras e vai para a linha 550 para seleccionar uma jogada ao acaso. Se SC é maior do que zero, o programa passa para a próxima linha e um dos números codificados é seleccionado do vector S, como XC na linha 510. A casa A é recapturada de XC na linha 520, e a casa M na linha 530. A linha 540 manda o programa para 650, onde deve fazer a jogada.

Regressando à primeira secção do programa, chegamos à linha 270. Aqui incrementa-se B. Os elementos um e dois do vector N contêm jogadas que podem ser feitas por qualquer peça (excepto alguma na fila de trás), e os elementos três e quatro contêm somente jogadas de dama. Se B, o contador que selecciona os elementos do vector N que serão usados para a jogada, for menor do que o seu máximo (dois para uma peça vulgar, quatro para

uma dama do computador), o programa regressa à linha 210, onde se incrementa o contador, B, e a pesquisa continua. A linha 280 verifica se todas as casas do tabuleiro foram pesquisadas e, caso contrário, regressa à linha 180, onde o contador de casas, A, decresce uma unidade, e a pesquisa continua. Se, no entanto, todas as jogadas foram usadas (isto é, A não é maior do que onze quando testado em 280) o programa continua para as duas linhas seguintes, que rastreiam uma eventual dama. Coloca-se a zero uma *flag* (ou bandeira, variável testada ao longo do programa e cujo valor geralmente indica a execução ou não de uma determinada acção) FL, e o computador verifica os elementos do vector que representam as casas na fila antes daquela onde se fazem as damas do computador. Se aí encontrar uma peça dele, manda o programa para a sub-rotina da linha 1270. Se esta sub-rotina encontra uma jogada que irá produzir dama, a FL é colocada a um (dentro da sub-rotina), o que provoca a ida do programa para a linha 650, para se fazer efectivamente a jogada.

Se tal jogada não é possível, a linha 310 salta sobre a secção que executa a captura (à qual voltaremos em breve) e vai para a linha 500. Agora, tenha em mente que o computador respeita uma rígida hierarquia nas suas verificações:

- Posso capturar, e se puder, posso capturar novamente?
- Posso fazer dama?
- Posso fazer uma jogada que não coloque em perigo a peça movida ou coloque em perigo outra pedra?
- Posso fazer uma jogada legal?

Se a resposta a qualquer destas questões é "sim", então a jogada é executada e o tabuleiro reimprimido e pronto para a jogada humana. Se a resposta à primeira pergunta é "não", então a segunda questão é interrogada... e assim por diante. Se a resposta à pergunta final é "não", o computador desiste do jogo.

Há ainda uma outra questão, que se põe cada vez que se reimprime o tabuleiro, relacionando o número de peças capturadas pelos dois jogadores. Se o computador descobre que qual-

quer dos jogadores tomou as 12 peças do seu adversário, o jogo vai para uma rotina que dá conhecimento da vitória e põe fim à partida.

Esta questão, não adequada à nossa hierarquia de jogadas, menciona-se aqui como complemento para que se consiga compreender facilmente o modo como o computador chega a uma jogada. É com este tipo de ordenação das verificações dos lances que trabalha a maior parte dos programas de jogos, desde os de tipo "arcade" com gráficos em movimento até ao xadrez.

Se o computador alcançou a linha 500, a resposta para as duas primeiras perguntas foi um "não". Para fazer a terceira pergunta ("Posso fazer uma jogada segura?") o computador tem apenas de se reportar a SC (o contador de jogadas seguras). Se ele for zero, é porque não encontrou lances que valessem a pena de serem chamados seguros e, por isso, a resposta à pergunta é "não". Já examinámos o sistema das jogadas seguras, pelo que iremos agora observar a parte do programa em que o computador tem acesso ao seu "último recurso" — a procura de uma jogada aleatória, que começa na linha 550.

Aqui usa-se novamente a variável SC para armazenar, implicando a inicialização de outra. Será, no fim de contas, reposta a zero (pelo linha 170) antes da jogada seguinte, e nesta altura pode-se usar com segurança para outros objectivos. A linha 550 adiciona uma unidade ao valor de SC e escolhe aleatoriamente um quadrado entre 1 e 88. Se este quadrado não contém nem uma peça vulgar do computador nem uma dama do mesmo (linha 560), a acção vai para a linha 630, onde, no caso de terem sido escolhidos menos do que 300 números nesta secção, a acção se dirige de volta à linha 550 para uma nova tentativa.

No entanto, se no quadrado escolhido estiver um C ou um CK, a rotina das linhas 570 a 620 verifica se essa peça se pode mover. Em caso afirmativo, a linha 610 manda o programa para 650, onde se fará a jogada.

Se não se encontrou qualquer jogada e foram escolhidos ao acaso 300 números, o computador desistirá do jogo na linha 640.

Antes de entrarmos na secção que permite ao humano fazer uma jogada, vamos em primeiro lugar regressar à secção que

começa na linha 370. Conseguir-se acesso a esta linha após uma captura. A linha anterior incrementou em uma unidade a pontuação do computador (CO) antes de a linha 370 chamar a sub-rotina de impressão do tabuleiro que mostra a peça movida, a nova pontuação do *Spectrum* e — é evidente — uma casa em branco onde estava anteriormente a peça do computador. Lembremos que A era a casa de onde o computador se moveu, e M a casa para onde ele se vai mover. Agora, o valor de A é renovado para passar a ser igual à casa para onde o computador se moveu; e as linhas de 390 a 460 irão verificar se é possível mais um salto (ou saltos) e, em caso afirmativo, dá-los-á, verificando de seguida cada jogada (após a reimpressão do tabuleiro) para ver se pode efectuar uma captura subsequente. Quando termina a sequência de capturas, o computador entrega o programa ao humano para fazer a sua jogada.

A secção da jogada humana é simples. O jogador move a pedra e assinala a casa da qual se move indicando os números vertical e horizontal correspondentes pela introdução simultânea dos dígitos (como "31"). Então, depois de carregar na tecla ENTER, o jogador introduz os dois dígitos referentes à casa para a qual se quer mover. Faz-se a jogada, reimprime-se o tabuleiro e, se se capturar alguma peça, o computador pergunta se se pode saltar novamente. Se o jogador disser que não, o computador pensará numa jogada. Se disser que sim, pergunta-lhe qual a nova casa para onde deseja ir (porque, é evidente, o computador já conhece a casa onde está, pois mesmo agora aí chegou).

A secção que aceita a jogada humana começa na linha 810, onde os comentários REM assinalam que querendo terminar o jogo se pode introduzir 99. Introduzindo um "1" ordena-se ao computador que copie o *écran* para uma impressora.

Depois de feita a cópia, pede-se novamente que se introduza uma jogada. Depois de feita a movimentação dentro do vector que guarda as peças (na linha 910), dois ciclos (920 a 950) verificam se alguma peça de qualquer dos jogadores atingiu a categoria de dama; nesse caso mudam os símbolos, em concordância. Mencionámos anteriormente que uma peça não se pode mover como dama até à jogada seguinte à sua coroação. As peças

do computador são promovidas aqui porque ele (estando no meio da secção correspondente à jogada humana) está seguramente afastado da secção de geração das suas jogadas, não havendo, por isso, qualquer hipótese de tentar fazer batota movendo uma dama recentemente promovida. A propósito, supõe-se que embora uma peça do humano seja coroada dama imediatamente, ele não fará batota tentando movê-la de seguida. Não há condição no programa que o impeça de fazer batota, mas se o fizer destruirá a razão de ser do jogo, pois o *Spectrum* nunca ousaria ter a audácia de o enganar.

Agora, na linha 960, o computador verifica a diferença entre as casas "de" e "para" (donde vem a peça e para onde ela vai) do jogador humano. Se esta diferença é inferior a 12, então o homem não fez uma captura, em consequência do que o computador regressa ao ciclo principal no começo do programa para reimprimir o tabuleiro antes de seleccionar a sua próxima jogada. No entanto, se a diferença não for inferior a 12, então a captura foi executada e o computador dará — duas vezes em cada três — conhecimento desta captura com um comentário gerado pelas linhas 970 a 990. Comentada ou não, a captura será assinalada por um tom musical crescente produzido pela linha 995; o resultado do homem (HU) será incrementado pela linha 1000; a casa onde se fez a captura será "esvaziada" (pela segunda instrução da linha 1000), e o tabuleiro reimpresso na linha 1030 antes de se dar uma oportunidade ao humano para mover novamente. Caso contrário, a linha 1040 faz a acção regressar ao ciclo principal, evitando a linha 1060 que coloca a casa anterior "para" como a nova casa "de", e remete o programa para a linha 860 para aceitar o novo destino da casa "para".

Já vimos o resto do programa que controla a inicialização e a rotina "Posso fazer dama?". A única parte que não analisámos, e que é auto-explanatória, é a da secção de 680 a 780, que imprime o tabuleiro após cada jogada. Repare-se na linha 770, que secciona a instrução de regresso ao ciclo do programa principal caso os contadores humanos (HU) ou do computador (CO) igualem 12.

Eis o tabuleiro no início do jogo, e um pouco depois já com o *Spectrum* a ganhar:

Spectrum: 0 Humano: 0

```

12345678
8 0 0 0 0 0 0 0
7 0 0 0 0 0 0 0
6 0 0 0 0 0 0 0
5 0 0 0 0 0 0 0
4 x x x x x x x
3 x x x x x x x
2 x x x x x x x
1 x x x x x x x
12345678

```

Spectrum: 5 Humano: 1

```

12345678
8 0 0 0 0 0 0 0
7 0 0 0 0 0 0 0
6 0 0 0 0 0 0 0
5 0 0 0 0 0 0 0
4 x x x x x x x
3 x x x x x x x
2 x x x x x x x
1 x x x x x x x
12345678

```

```

10 REM JOGO DAS DAMAS
50 RANDOMIZE
80 GO SUB 1070
90 REM Apague a linha seguinte
para o Spectrum nao ser o pri-
meiro a jogar
100 GO TO 130
110 GO SUB 880
120 GO SUB 840
130 GO SUB 880
140 GO SUB 160
150 GO TO 110
160 DIM s(10)
170 LET sc=0: LET a=89
180 LET a=a-1
190 IF q(a)<>c AND q(a)<>ck THE
N GO TO 280

```

```

200 LET b=0: IF a<29 THEN LET b
=20
210 LET b=b+1
220 LET m=a+n(b)
230 IF m>88 OR m<11 THEN GO TO
230
240 IF 10*INT (m/10)<>m THEN PR
INT AT 0,0;a;" para ";m;"?

250 IF (q(m)=h OR q(m)=hk) AND
q(m+n(b))=e THEN GO TO 320
260 IF q(m)=e THEN IF (q(m-11)<
>h AND q(m-11)<>hk) THEN IF (q(m
-9)<>h AND q(m-9)<>hk) AND q(m+9
)<>hk THEN IF ((q(m+22)<>hk OR q
(m+18)<>hk) AND (q(m+9)<>c OR q(
m+9)<>ck OR q(m+11)=c OR q(m+11)
=ck)) AND q(m+11)<>hk THEN GO SU
B 470
270 IF b<2 OR (q(a)=ck AND b<4)
THEN GO TO 210
280 IF a>11 THEN GO TO 180
290 LET fl=0: IF q(22)=c OR q(2
4)=c OR q(26)=c OR q(28)=c THEN
GO SUB 1270
300 IF fl=1 THEN GO TO 550
310 GO TO 500
320 LET q(m+n(b))=q(a): LET q(m
)=e: LET q(a)=e
330 PRINT AT 0,0; FLASH 1; INK
3;"*****"
340 FOR t=-10 TO 55 STEP 3: SEE
P 0.008,t: NEXT t
350 PRINT AT 0,0; INK 2; PAPER
7; FLASH 1; BRIGHT 1;" " apa
nhei-te.....
360 LET co=co+1
370 GO SUB 680
380 LET a=m+n(b).
390 LET b=0
400 LET b=b+1
410 IF (a+2*n(b)<11 OR a+2*n(b)
>88) AND b<4 THEN GO TO 400
420 LET m=a+n(b)
430 IF q(m)=c AND b>3 THEN RETU
RN
440 IF (q(m)=h OR q(m)=hk) AND
q(m+n(b))=e THEN GO TO 320
450 IF b<2 OR (q(a)=ck AND b<4)
THEN GO TO 400
460 RETURN

```

```

470 IF sc<10 THEN LET sc=sc+1
480 PRINT AT 0,0; INVERSE 1;"Jo
gada segura: ";a;" para ";m;"
": BEEP 0.5,sc
490 LET s(sc)=100*a+b+20: RETUR
N
500 IF sc=0 THEN GO TO 550
510 LET xc=INT (RND*sc)+1
520 LET a=INT (s(xc)/100)
530 LET m=a+n(s(xc)-100*a-20)
540 GO TO 650
550 LET sc=sc+1: LET a=INT (RND
*sc)+1
560 IF q(a)<>c AND q(a)<>ck THE
N GO TO 630
570 LET b=0
580 LET b=b+1
590 LET m=a+n(b)
600 IF m>88 OR m<11 THEN GO TO
620
610 IF q(m)=e THEN GO TO 650
620 IF b<2 OR q(a)=ck AND b<4 T
HEN GO TO 580
630 IF sc<300 THEN GO TO 550
640 PRINT AT 0,0;"Desisto...":
STOP
650 LET q(m)=q(a): LET q(a)=e
660 PRINT AT 0,0;"De ";a;" para
";m;"
TO -30 STEP -2: BEEP 0.008,t: N
EXT t
670 RETURN
680 PRINT AT 4,2; BRIGHT 1; INK
6;" Spectrum: ";co;" Humano:
";hu;
710 PRINT TAB 6; BRIGHT 1; INK
6;"12345678"
720 FOR f=80 TO 10 STEP -10
730 PRINT TAB 6; BRIGHT 1;f/10;
740 FOR g=1 TO 8: PRINT CHR$(q(
f+g)); NEXT g
750 PRINT f/10; NEXT f
760 PRINT TAB 6; BRIGHT 1; INK
6;"12345678"
770 IF co=12 OR hu=12 THEN GO T
O 790
780 PRINT AT 0,0;" ": RETURN
790 IF hu=12 THEN PRINT AT 0,0;
"Ganhou... Desta vez...": STOP
800 IF co=12 THEN PRINT AT 0,0;
"Ganhei eu... Vai outro?": STOP

```

```

810 REM 99 para desistir
820 REM 1 para copiar o ecran
830 INPUT FLASH 1;" De? ";a
840 IF a<>120 AND a<>75 THEN
GO TO 850
850 IF a=99 THEN PRINT AT 0,0;"
Nao desanime!"; STOP
870 IF a=1 THEN COPY : GO TO 85
0
900 INPUT FLASH 1;(a);" Para ?
";b
905 IF q(b)<>e THEN GO TO 900
910 LET q(b)=q(a); LET q(a)=e
920 FOR t=11 TO 17: IF q(t)=c T
HEN LET q(t)=ck
930 NEXT t
940 FOR t=82 TO 88: IF q(t)=h T
HEN LET q(t)=hk
950 NEXT t
960 IF ABS (a-b)<12 THEN RETURN
970 LET ty=RND
980 IF ty<0.3 THEN PRINT AT 0,0
;"Bom lance"
990 IF ty>0.7 THEN PRINT AT 0,0
;"Apanhou-me!"
995 FOR t=-20 TO -1 STEP 0.7: B
EEP 0.01,t; NEXT t: PRINT AT 0,0
"
1000 LET hu=hu+1: LET q((a+b)/2)
=e: GO SUB 680
1010 FOR t=82 TO 88: IF q(t)=h T
HEN LET q(t)=hk
1020 NEXT t
1030 INPUT FLASH 1;" Pode comer
outra vez? (s/n) ";a$
1040 IF a$<>"S" AND a$<>"s" THEN
RETURN
1050 LET a=b: GO TO 850
1070 PAPER 1: BORDER 1: INK 7: C
LS
1075 PRINT AT 0,0;"Aguarde"
1080 LET h=120: LET hk=75: LET c
=111: LET ck=36: LET e=32: LET b
=143
1090 LET of=-99: DIM q(99): DIM
n(4)
1100 FOR m=1 TO 99: LET q(m)=of:
NEXT m
1110 FOR m=1 TO 64
1120 READ d: READ g
1130 LET q(d)=g: NEXT m
1140 DATA 81,b,82,c,83,b,84,c,85
,b,86,c,87,b

```

```

1150 DATA 88,c,71,c,72,b,73,c,74
,b,75,c,76,b
1160 DATA 77,c,78,b,61,b,62,c,63
,b,64,c
1170 DATA 65,b,66,c,67,b,68,c,61
,e,62,b
1180 DATA 53,e,54,b,55,e,56,b,57
,e,58,b
1190 DATA 41,b,42,e,43,b,44,e,45
,b,46,e
1200 DATA 47,b,48,e,31,h,32,b,33
,h,34,b,35,h
1210 DATA 38,b,37,h,38,b,21,b,22
,h,23,b,24,h
1220 DATA 25,b,26,h,27,b,28,h,11
,h,12,b,13,h
1230 DATA 14,b,15,h,16,b,17,h,18
,b
1240 FOR m=1 TO 4: READ n(m): NE
XT m
1250 DATA -11,-9,11,9
1260 LET co=0: LET hu=0: RETURN
1270 IF q(22)=c AND q(11)=e THEN
LET a=22: LET m=11: LET fl=1: R
ETURN
1280 IF q(22)=c AND q(13)=e THEN
LET a=22: LET m=13: LET fl=1: R
ETURN
1290 IF q(24)=c AND q(13)=e THEN L
ET a=24: LET m=13: LET fl=1: R
ETURN
1300 IF q(24)=c AND q(15)=e THEN
LET a=24: LET m=15: LET fl=1: R
ETURN
1310 IF q(26)=c AND q(15)=e THEN
LET a=26: LET m=15: LET fl=1: R
ETURN
1320 IF q(26)=c AND q(17)=e THEN
LET a=26: LET m=17: LET fl=1: R
ETURN
1340 RETURN

```

JOGO DO GALO

No conhecido JOGO DO GALO cada adversário coloca à vez os seus símbolos (um zero ou uma cruz) nas posições vagas de uma grelha de três por três, por forma a tentar obter três símbolos iguais alinhados em qualquer direcção, quer seja na vertical, na horizontal ou na diagonal. O jogo foi tão bem analisado que não há muitas surpresas ao jogá-lo. Em vez dos previsíveis progra-

mas nos quais o homem empata na melhor das hipóteses (e geralmente perde), demo-nos ao trabalho de escrever um programa que joga de um modo não previsível, mas mesmo assim bastante inteligente.

O programa que apresentamos permite que seja o computador ou o seu adversário a jogar primeiro (a decisão de quem joga primeiro é determinada pela linha 50).

Além de escolher o quadrado do meio se este estiver livre, as jogadas do *Spectrum* são difíceis de prever (excepto, é claro, se quiser completar uma linha de três símbolos, caso ela exista, ou pretender bloquear uma das suas potenciais linhas de três símbolos).

Se RND na linha 50 é maior do que 0.5, o computador decide jogar primeiro e, após uma pausa, vai para a linha 50, que manda a acção para a sub-rotina da linha 700, que faz imprimir o quadro. No regresso desta sub-rotina, o computador vai para a sub-rotina da linha 430, que verifica as vitórias. Esta rotina, que verifica todas as possíveis vitórias, continua na linha 550.

De volta dessa rotina, o computador analisa a casa central (número cinco). Se encontrar esta posição vazia, o computador ocupa-a e (como se indica no fim da linha) vai imediatamente para a linha 60, onde a sub-rotina de impressão do quadrado é chamada antes de ser aceite o lance do jogador humano.

Se esta jogada simples não pode ser feita, o computador imprime a mensagem "Aguarde"; procura uma jogada vitoriosa da primeira vez que passa pela rotina de 140 a 280 e, se não encontrar alguma, procura uma potencial linha de três símbolos que possa bloquear. Se não encontra nada disto, o computador escolhe até 20 jogadas ao acaso, procurando um local para colocar o seu símbolo. Se não encontrar qualquer local, o computador (usando a rotina de 350 a 380) verifica os quadrados um por um. Se não encontrar qualquer movimentação possível, o computador sabe que todos os quadrados estão cheios. Também sabe, porque já fez anteriormente uma verificação de "jogadas vitoriosas", que nem ele nem o adversário ganharam, e por isso o resultado tem de ser um empate. A linha 400 imprime isso mesmo. Após um ou dois trinos (rotina de 590 a 600), o

programa volta a correr automaticamente.

A rotina do lance do jogador, a partir da linha 620, permite a este introduzir as suas jogadas através de INKEY\$, não sendo por isso necessário carregar em ENTER após seleccionar um número.

Note-se como todo o jogo é chamado dum ciclo infinito de 60 a 100, imprimindo o quadro, aceitando o lance do jogador, imprimindo de novo o quadro, verificando uma eventual vitória ou bloqueio, fazendo a jogada do computador... e assim por diante, até final do jogo. A vantagem de trabalhar assim, com o programa feito de chamadas a sub-rotinas a partir de um ciclo infinito, é que podem modificar-se as diferentes partes do programa sem ter de desalojar algumas escritas anteriormente à medida que vão sendo melhoradas.

Desejando melhorar a maneira de jogar do programa (embora à custa de o programa se tornar mais previsível), pode-se fazê-lo examinar os quadrados na secção "jogada ao acaso" (linha 290) numa ordem preestabelecida.

```

10 REM JOGO DO GALO
20 DIM A(9)
30 RANDOMIZE
40 BORDER 1: INK 7: PAPER 1: C
LS
50 IF RND>0.5 THEN PRINT AT 5,
S: FLASH 1: "EU Jogo primeiro..."
: FOR e=1 TO 50: BEEP 0.08,e: NE
XT e: CLS: GO TO 90
60 GO SUB 700
70 GO SUB 430
80 GO SUB 620
90 GO SUB 700
100 GO SUB 430
110 IF a(5)=0 THEN LET a(5)=1:
GO TO 60
120 REM PARA COMPLETAR FILA/BLO
QUEAR
130 PRINT AT 1,1: FLASH 1: "Aguar
de"
140 LET d=1
150 LET b=1
160 IF b=1 THEN LET x=1: LET y=
2: LET z=3

```

```

170 IF b=2 THEN LET x=1: LET y=
4: LET z=7
180 IF b=3 THEN LET x=1: LET y=
5: LET z=9
190 IF b=4 THEN LET x=3: LET z=
7
200 LET c=1
210 IF a(x)=d AND a(y)=d AND a(
z)=0 THEN LET a(z)=1: GO TO 50
220 IF a(x)=d AND a(y)=0 AND a(
z)=d THEN LET a(y)=1: GO TO 50
230 IF a(x)=0 AND a(y)=d AND a(
z)=d THEN LET a(x)=1: GO TO 50
240 IF b=1 THEN LET x=x+3: LET
y=y+3: LET z=z+3
250 IF b=2 THEN LET x=x+1: LET
y=y+1: LET z=z+1
260 IF c<3 THEN LET c=c+1: GO T
O 210
270 IF b<4 THEN LET b=b+1: GO T
O 170
280 IF d<2 THEN LET d=d+1: GO T
O 150
290 REM JOGADA AO ACASO
300 LET b=1
310 LET c=INT (RND*9)+1
320 IF a(c)=0 THEN LET a(c)=1:
GO TO 50
330 LET b=b+1
340 IF b<21 THEN GO TO 310
350 LET b=0
360 LET b=b+1
370 IF a(b)=0 THEN LET a(b)=1:
GO TO 50
380 IF b<9 THEN GO TO 360
390 GO SUB 700
400 PRINT "FLASH 1;TAB 11;"
Empatamos!
410 GO TO 590
420 REM VERIFICAR VITORIAS
430 FOR b=1 TO 4
440 IF b=1 THEN LET x=1: LET y=
2: LET z=3
450 IF b=2 THEN LET x=1: LET y=
4: LET z=7
460 IF b=3 THEN LET x=1: LET y=
5: LET z=9
470 IF b=4 THEN LET x=3: LET z=
7
480 FOR c=1 TO 3
490 IF a(x)=a(y) THEN IF a(y)=a
(z) THEN IF a(x)<>0 THEN GO TO 5

```

```

50
510 IF b=1 THEN LET x=x+3: LET
y=y+3: LET z=z+3
520 IF b=2 THEN LET x=x+1: LET
y=y+1: LET z=z+1
530 NEXT c
540 NEXT b
545 IF flag=0 THEN GO TO 400
550 RETURN
560 BEEP 1,1: BEEP 2,2
570 IF a(x)=1 THEN PRINT "F
LASH 1;TAB 8;"Ganhei! Ganhei!
580 IF a(x)=2 THEN PRINT "FL
ASH 1;TAB 12;"Ganhou...
590 FOR a=1 TO 25: BEEP 0.05,a+
10*(RND-0.5)*(a(x)=2): NEXT a
600 FOR a=50 TO 25 STEP -0.5: B
EEP 0.05,a+5*(RND-0.5)*(a(x)=2):
BORDER RND*7: NEXT a
610 RUN
620 REM LANCE DO JOGADOR
630 PRINT AT 1,1: FLASH 1;"E VO
ce a Jogar...";AT 1,1: OVER 1;"
640 LET a$=INKEY$
650 IF a$<"1" OR a$>"9" THEN GO
TO 640
660 LET b=VAL (a$)
670 IF a(b)<>0 THEN GO TO 640
680 LET a(b)=2
690 RETURN
700 REM IMPRESSAO
710 PRINT AT 1,1;"
720 PRINT AT 6,5;"1 2 3 ";
725 LET flag=0
730 FOR b=1 TO 9
735 IF a(b)=0 THEN LET flag=1:
740 IF a(b)=0 THEN PRINT "- ";
: BEEP 0.05,10
750 IF a(b)=1 THEN PRINT " O ";
: BEEP 0.09,50
760 IF a(b)=2 THEN PRINT " X ";
: BEEP 0.05,-3
770 IF b=3 THEN PRINT AT 8,5;"4
5 6
780 IF b=6 THEN PRINT AT 10,5;"
7 8 9
790 NEXT b
810 RETURN

```


XADREZ PARA PRINCIPIANTES

Em 1978, quando os computadores jogadores de xadrez eram raros, os leitores da revista semanal romena *Magazinul* tomaram parte num jogo contra um computador chamado, naquele país, *Felix C-256*. Programado pelo matemático Viorel Darie, do Instituto de Técnica de Computação de Bucareste, o programa — ASTRO 64 — levou dois anos a desenvolver. *Felix* fazia uma jogada e os leitores eram convidados a responder por escrito com as suas jogadas. A jogada com maior aprovação era introduzida no computador, e a resposta de *Felix* publicada na semana seguinte.

Apesar das 600 horas gastas a escrever o programa, *Felix* não jogava muito bem e, por fim, foi derrotado pelos leitores da *Magazinul*.

Um perito em xadrez afirmou que o computador por vezes fazia boas ou excelentes jogadas mas noutras alturas jogava como um “jogador de xadrez sem experiência” e cometia mesmo “erros de principiante”.

Como sugerem o título deste capítulo e a história de *Felix*, as mesmas críticas podem vir a ser feitas ao nosso programa de XADREZ PARA PRINCIPIANTES, que não joga bom xadrez e se derrota com facilidade, mas foi incluído neste livro por várias razões.

O xadrez foi sempre uma tentação para os programadores.

Nos primeiros e inocentes dias do microcomputador, pensava-se que o facto de se conseguir programar um computador para jogar xadrez constituiria uma prova de que os computadores teriam de considerar-se verdadeiramente inteligentes. Em virtude de ter de suportar o enorme número de permutas que podem ocorrer num jogo, o campo de acção de um programa de xadrez tende a ser muito vasto comparado, por exemplo, com um programa para jogar damas ou o jogo do galo. Existem à volta de 10^{120} jogadas possíveis em 40 lances, um número de jogadas semelhante ao número de átomos do Universo. Programar um computador para abranger todas estas hipóteses é um desafio terrível.

Nunca foram publicados muitos programas de xadrez em *Basic*. O facto de se desenvolver um programa que jogue num tempo razoável é, pensamos nós, de certo interesse e não há dúvida que ver este programa a jogar é fascinante, mesmo que não seja capaz de fazer melhorar o xadrez.

Este programa, na maior parte das vezes, joga muito rapidamente: localiza, armazena e escolhe as posições das suas peças em menos de quatro segundos e geralmente joga dentro de seis a quinze segundos. À medida que as posições no tabuleiro se tornam mais complexas, o ritmo do jogo abrandando, mas o *Spectrum* continua ainda a levar menos de quarenta segundos para jogar.

Após a listagem, acrescentámos uma variante do programa que permite ao computador jogar contra si próprio. É fascinante observá-lo a lutar contra as brancas, depois contra as pretas, dando ao mesmo tempo uma boa noção da fraqueza de jogo do *Spectrum*; esperamos que o leitor se sinta inspirado para remediar essa fraqueza. O programa foi escrito deliberadamente numa forma modular para facilitar o acesso às secções que manobram determinadas peças, por forma a que cada qual possa modificar gradualmente o programa a seu gosto.

O simples facto de *Felix* conseguir jogar xadrez foi uma novidade em 1978.

E é talvez surpreendente que, embora a primeira obra importante sobre as possibilidades do computador disputar este jogo, *Programming a Computer for Playing Chess* (Programar um computador para jogar xadrez), tenha sido escrita em 1949 por Claude Shannon (um vulto importante no desenvolvimento dos computadores), o primeiro programa só foi publicado uma década mais tarde, quando Alex Bernstein e três dos seus colegas da IBM conseguiram fazer um programa que funcionava num IBM 704. Apesar deste tardio começo, há actualmente uma grande variedade de programas sobre xadrez, incluindo algumas versões destinadas somente ao *Spectrum*.

Mas o xadrez é, como dissemos, um enorme desafio para os programadores.

Tendo introduzido este programa e trabalhado um pouco com

ele, tem-se uma ideia melhor sobre a maneira como este jogo foi desenvolvido de início: a partir daqui aprecia-se melhor como trabalha a delicada máquina do xadrez, e depois disso pode começar a pensar-se na elaboração de um programa completo a partir do zero.

Usámos neste programa um número de ideias fundamentais na escrita de jogos de tabuleiro, e encontram-se muitos outros programas que utilizam estas ideias.

Como primeira decisão, encarámos um objectivo realista. Quando este jogo ainda era para nós uma novidade, debatíamos-nos com um anterior jogo de damas que tínhamos escrito, tentando imaginar como o computador poderia reconhecer e executar jogadas múltiplas. Trevor Sharples (autor com o qual escrevemos dois livros de parceria) sugeriu que se projectasse o programa apenas tendo em conta duas jogadas antecipadas.

Como o programa não jogava lá muito bem, ele observou que a possibilidade de o computador prever o desenvolvimento de três jogadas contra um adversário humano era tão pouco provável que não merecia ser levada em conta. Mais tarde desenvolvi uma simples sub-rotina para fornecer "n" jogadas, mas nessa altura aquele conselho não me serviu de nada. É de Claude Shannon (que escreveu a obra sobre jogos de xadrez em 1949) a afirmação de que há aproximadamente 10^{120} sequências diferentes de lances possíveis numa partida de 40 jogadas. Shannon acrescentou que um computador rápido levaria 1090 anos para examinar todas as jogadas possíveis antes mesmo de decidir que o primeiro peão avançasse duas casas.

Decidimos que 1090 anos era um pouco demorado para esperar e que 40 vezes 1090 (aproximadamente) poria severamente à prova a nossa paciência.

Uma pesquisa exhaustiva estava obviamente posta de parte, mesmo que tivéssemos habilidade (que não temos) para escrever um programa que, de alguma maneira, determinasse essas jogadas e escolhesse uma que valesse a pena.

Embora estivesse nos nossos planos escrever um programa exclusivamente com os meios básicos para averiguar das sequên-

cias das jogadas, sabíamos que o número de combinações, mesmo ao nível mais simples, tornaria o programa muito, muito lento.

Quando fizemos correr o programa, e embora tivéssemos tentado otimizar-lhe a velocidade, fomos surpreendidos pela rapidez do jogo. Um programa que tem, no fim da listagem, sub-rotinas que são frequentemente chamadas, tende a correr mais devagar do que se as sub-rotinas estiverem mais próximas do início do programa. Isto porque um computador, ao procurar as sub-rotinas do destino, começa no primeiro número de linha do programa e depois atravessa-o linha por linha procurando pelo princípio da sub-rotina. Se a rotina se encontra próxima do início há menos linhas para pesquisar.

Planeámos cuidadosamente todo o programa, escrevendo a primeira versão totalmente no papel, antes de introduzir no *Spectrum* um único carácter. Isto permitiu-nos correr o programa "à mão", mais vezes do que nos conseguimos lembrar, e baralhar as diversas partes para tornar a estrutura o mais lógica possível. Aparentemente este facto manteve baixo o tempo de resposta. Portanto, as duas primeiras ideias que informam este programa são a necessidade de nos limitarmos a um horizonte moderado, e ter uma estrutura que limite ao mínimo o tempo de resposta.

A seguir, a ideia mais importante é a relativa à apresentação do tabuleiro.

Nas primeiras experiências em jogos de tabuleiro para computador usámos o sistema de numerar o tabuleiro, que desenvolvemos baseados numa ideia exposta por A. L. Samuels num artigo da *Scientific American* (ver "Systems Analysis and Programming", de C. Strachey, em *Readings from Scientific American* — W. H. Freeman and Co., São Francisco, 1971).

Embora isto resultasse satisfatoriamente, requeria uma longa rotina para converter os lances introduzidos pelo jogador em números utilizáveis pelo computador.

Um sistema de numeração do tabuleiro bem mais satisfatório foi-nos sugerido nos finais de 1981 por Graham Charlton (co-autor de *The Turing Criterion — Machine Intelligent Pro-*

grams for the 16K ZX81, Interface, Londres, 1982) e que se encontra no coração dos programas deste livro DAMAS e XADREZ PARA PRINCIPIANTES. Todo o tabuleiro é guardado numa matriz e as jogadas são feitas trocando simplesmente elementos da matriz entre si, substituindo um elemento da matriz com o carácter gráfico que representa um espaço em branco quando uma pedra deixa uma casa, e colocando o código dessa peça no elemento da matriz representando a posição para a qual a pedra está a mover-se. Quando se imprime o tabuleiro, o computador tem apenas de imprimir de cada vez na matriz o carácter (CHR\$) de cada elemento. O sistema de numeração do tabuleiro sugerido por Charlton também vai de encontro a uma simples conversão entre os números introduzidos por um jogador para indicar a jogada que quer fazer, e o número do elemento da matriz que a jogada daquele jogador representa.

Soubemos antes de iniciar o programa que o jogador deveria ser capaz de se referir às casas do tabuleiro pela notação algébrica, que é bastante comum nos círculos ligados ao xadrez; por isso preparámos o tabuleiro por forma a facilitar ao máximo esta tarefa. O diagrama a seguir mostra as casas do tabuleiro; à volta do tabuleiro estão os números e letras da notação algébrica do xadrez, e no interior do tabuleiro os elementos da matriz que se referem a cada casa.

	A	B	C	D	E	F	G	H
8	18	28	38	48	58	68	78	88
7	17	27	37	47	57	67	77	87
6	16	26	36	46	56	66	76	86
5	15	25	35	45	55	65	75	85
4	14	24	34	44	54	64	74	84
3	13	23	33	43	53	63	73	83
2	12	22	32	42	52	62	72	82
1	11	21	31	41	51	61	71	81

A vantagem real do sistema de Charlton sobre outros sistemas de numeração reside, na nossa opinião, em que qualquer jogada

pode ser descrita como uma simples adição "à", ou subtracção "da" casa de partida. Uma explicação deve clarificar. Imagine-se que há um cavalo na casa com o número 45. Como se sabe, no xadrez o cavalo movimenta-se em L, duas casas e depois uma ou uma casa e depois duas.

Uma jogada possível para um cavalo é ir da sua casa actual (45) para casa com o número 66, o que corresponde a um aumento de 21 unidades. Agora imagine-se que o cavalo estava na casa 11. Adicione-se 21 e obtém-se 32, outra jogada legal possível. Começando em 52 e adicionando 21, obtém-se um destino (correcto) para a casa 73. As oito jogadas do cavalo a partir de qualquer posição (supondo que a jogada não o coloca fora do tabuleiro) podem determinar-se adicionando os seguintes números ao quadrado de partida:

19, -19, 21, -21, 8, -8, 12, -12.

De um modo semelhante, as jogadas do peão (à excepção da primeira jogada, em que pode avançar-se duas casas) são determinadas por adição pura e simples de uma unidade ao quadrado de partida (por isso um peão pode mover-se legalmente de 45 para 46 ou de 23 para 24). Um peão captura outra peça movendo-se uma casa na diagonal, e por isso pode capturar em casas designadas como menos 11 ou mais 9. Com base nestes conhecimentos, programaremos facilmente um computador para procurar os movimentos legais possíveis da casa de partida.

Foi isso exactamente o que fizemos neste programa. Observando as linhas seguintes a 2640 na secção de inicialização do XADREZ PARA PRINCIPIANTES vê-se como isso se faz. Em primeiro lugar (na linha 2650) DIMENSIONA-se um certo número de vectores para serem usados em várias tarefas. Entre eles encontram-se vectores para guardar as potenciais jogadas de determinadas peças.

O vector N é guardado em DATA na linha 2870. Nessa linha vêem-se os mesmos oito números, tal como foram observados antes para guardarem os lances do cavalo. Uma maneira de fazer um lance do cavalo (método que usamos, com algumas limita-

ções, em determinado ponto do programa) é gerar um número aleatoriamente entre um e oito, e testar esse elemento do vector N para ver se a casa que o cavalo ocupa, mais o número nesse elemento do vector N, é uma casa vazia ou tem uma peça do adversário. Se se souber que se trata de uma peça adversária e alguns testes indicarem que este quadrado não enfrenta perigo de captura por parte de outras peças, pode fazer-se a jogada.

Esta é a técnica básica do XADREZ PARA PRINCIPIANTES.

Medita-se bem no que escrevemos. Quem estiver disposto a investir o volume de trabalho metódico necessário, provavelmente será agora capaz de escrever um programa de xadrez a partir do rascunho, munido da informação já fornecida.

Um programa em que se progrida simplesmente pelo movimento de dois dados, como esta explicação sugere, será presa fácil e rápida mesmo para um principiante. Portanto o processo de geração de jogadas tem de ser modificado em qualquer parte, como se faz neste programa, mas a execução das jogadas funciona como foi descrita. Os vectores das peças estão no programa como a seguir se indica:

Cavalo — 2860 a 2870

Torre — 2880 a 2920

Bispo — 2930 a 2970

Rainha — 2990 carrega o vector Q com os elementos dos vectores bispo e torre, porque o movimento da rainha é apenas uma combinação de todos os movimentos do bispo e da torre

Rei — 3000 a 3010

O vector que se guarda depois, o vector S (linhas 3020 a 3100), armazena o segredo seguinte do nosso programa. Muitos programas "inteligentes" de jogos de tabuleiro (tal como o de damas neste livro) percorrem cada casa do tabuleiro desde a do canto superior esquerdo até à do canto inferior direito, casa por casa, à procura de peças, capturas e movimentações. A nossa

decisão, e aquela que — mais do que qualquer outra — conta para a velocidade do programa, foi de amostrar as casas numa ordem que considerámos a mais adequada ao jogo de xadrez. A ordem segundo a qual se mostram as casas é guardada nas instruções DATA 3030 a 3100.

Eis os números para comparar com as localizações no tabuleiro numerado dadas atrás.

46	56	66	76	86	96	47	57	67	77
36	46	56	66	76	86	37	47	57	67
26	36	46	56	66	76	27	37	47	57
16	26	36	46	56	66	17	27	37	47
06	16	26	36	46	56	07	17	27	37
45	55	65	75	85	95	44	54	64	74
35	45	55	65	75	85	34	44	54	64
25	35	45	55	65	75	24	34	44	54
15	25	35	45	55	65	14	24	34	44
05	15	25	35	45	55	04	14	24	34

O programa olha sempre para a casa 46 em primeiro lugar, depois para a 56 e assim por diante, ao longo da lista. Vale a pena estudar isto por momentos, uma vez que esta ordem é uma das coisas alteráveis ao gosto de cada um na devida altura.

Vamos recapitular as partes do programa que estudámos até agora.

Salientámos que este programa foi incluído porque o xadrez tem-se demonstrado de constante fascínio para os programadores de computadores, e embora o XADREZ PARA PRINCIPIANTES não jogue bem, ensina bastante acerca da programação de xadrez e de jogos de tabuleiro similares. Apesar de ser em Basic, joga notavelmente depressa.

À listagem do programa segue-se uma modificação das dez primeiras linhas, o que permite ao computador jogar contra si próprio. Isto fornece uma apresentação fascinante e é uma boa maneira de descobrir erros no programa, uma vez que se pode deixar a correr — todo o dia se necessário — jogando contra si próprio a fim de ver se quebra em algum ponto, o que indicará um erro de impressão.

Já se falou do sistema de numeração do tabuleiro e já se explicou que este sistema indica que as jogadas de qualquer peça podem ser expressas em termos de uma adição "ao" ou de uma

subtração "do" elemento do vector que representa a casa no tabuleiro de xadrez. As jogadas possíveis de cada peça são armazenadas num vector, com o nome de cada um desses vectores escolhidos para facilitar a identificação do conteúdo de cada, como Q para o vector que guarda as jogadas da rainha (rainha = Queen) e B para as jogadas do bispo. O vector S guarda a sequência segundo a qual as casas do tabuleiro são verificadas quando o computador está à procura de uma jogada.

Eis o esquema do programa:

- 10-30 Informação inicial, assegura que CAPS LOCK está ligada.
- 40-60 Estas três linhas reimprimem o tabuleiro após a jogada do computador, chamando a atenção ao jogador para introduzir a sua jogada, e reimprime novamente o tabuleiro.
- 70-2350 Este é o eixo do programa, e contém toda a parte pensante. Analisá-la-emos oportunamente em pormenor.
- 2360-2460 Esta parte reimprime o tabuleiro após cada jogada e (nas linhas 2410 e 2420) promove peões que chegaram a rainhas ao alcançarem a linha de trás.
- 2570-2630 Esta secção aceita o lance do jogador, chamando a sub-rotina dos comentários se foi tomada uma peça do computador, e permite ao jogador assinalar vários factos ao programa após a jogada ter sido introduzida. O programa pede a introdução de informação sob a forma de uma letra e um número (tal como A5) introduzido como uma simples palavra para designar a casa donde se está a

mover, e depois uma letra e um número subsequentes para indicar a casa para onde se deseja mover. Feito isto (e o programa verifica o comprimento da informação introduzida, mas não verifica a legalidade da jogada), são-lhe apresentadas as seguintes opções:

- X — Tem o computador em xeque
- C — Quer uma cópia impressa do tabuleiro
- T — Deseja trocar de lados
- P — Para terminar o jogo
- ENTER — Para continuar o jogo

Indicada a opção desejada, as linhas 2625 e 2630 fazem a sua jogada.

Aqui terminam as partes de funcionamento do programa, excepto para as sub-rotinas que comentam a captura pelo jogador e para a rotina chamada quando se deseja trocar de lados.

O resto do programa é inicialização, como se segue:

- 2640 Localiza o gerador de número aleatório, coloca a variável MM a zero, e coloca o vector literal A\$ (que aceita a primeira parte do lance do jogador e é usado para assinalar o xeque e jogadas semelhantes) a "". MM é a mais importante variável de todo o programa. Quando o computador encontra uma jogada, MM (assim chamada evocando o "movimento da máquina") muda de zero para um. A *flag* MM é então verificada durante a execução do programa e apenas quando o seu estado muda de zero para um fará o computador interromper a busca de uma jogada. Repetirá, se necessário, o processo de procurar uma jogada oito vezes, depois das quais desiste, se não tiver tomado qualquer decisão que mude MM de zero para 1.

2650 Esta secção DIMensiona os vectores. Aqui está o que eles fazem:

- A — guarda o tabuleiro, o vector mestre
- R — jogadas da torre
- B — jogadas do bispo
- N — jogadas do cavalo
- Q — jogadas da rainha
- K — jogadas do rei
- Z — usado quando se trocam os lados
- S — sequência segundo a qual as casas são verificadas
- T — guarda a localização de cada peça, conteúdo alterado por cada jogada

2660-2670 Estas duas linhas guardam as variáveis usadas para as peças. Estas, por sinal, são representadas por letras (P para peão, K para rei, N para cavalo, Q para rainha, B para bispo e R para torre) com o computador jogando com as pedras pretas no topo do *écran* representadas por maiúsculas, e o jogador humano movendo-se com as pedras representadas por minúsculas.

Eis o aspecto do tabuleiro no início do jogo:



Como se pode ver, o tabuleiro tem pontos em vez de casas pretas e brancas, e não há gráficos definidos pelo utilizador. O objectivo deste programa consiste em colocar o computador a jogar xadrez e não em dar ao jogo um aspecto bonito;

mas se o quiserem alindar adicionando peças de xadrez com uma forma "atraente", é tentá-lo. (Nesse caso mandem-nos uma cópia, porque teríamos muito prazer em apreciá-la.)

Cada casa "de" e "para" é indicada pela letra da vertical no topo ou no fundo seguida pelo número da horizontal, pelo que a rainha do jogador começa em D1 e o cavalo do rei relativo ao computador começa em G8.

Aqui estão as variáveis para as peças:

Humano — brancas: P (peão) — 112; R (torre) — 116; N (cavalo) — 99; B (bispo) — 98; Q (rainha) — 160; K (rei) — 114.

Computador — pretas: PB (peão preto) — 80; RB (torre preta) — 84; NB (cavalo preto) — 67; BB (bispo preto) — 66; QB (rainha preta) — 68; KB (rei preto) — 82

2680-3100 Esta secção engloba os vectores dos movimentos possíveis para as peças. Estes já foram estudados.

3110 Coloca o fundo e o enquadramento a azul, e a cor da tinta a branco. Mas o utilizador pode mudá-los a seu gosto.

3120-3210 Esta rotina permite ao humano trocar de lados em qualquer altura do jogo após ter feito a sua jogada. A mudança funciona como se segue: o computador actua como se um espelho tivesse sido colocado a meio do tabuleiro e as peças se reflectissem nele; por isso uma peça colocada em A2 acabaria por ir parar a A7. Embora a posição das peças seja trocada, o computador continua a jogar com letras

maiúsculas e o adversário com as minúsculas, embora tenham herdado as posições um do outro. A opção de "autojogo" faz apenas uma mudança após cada jogada, e imprime o tabuleiro de dois em dois lances (sempre que as peças estão "às direitas") para que se fique com um registo do jogo. Este sistema de trocas é fácil de compreender uma vez utilizado.

3310-3380 Se se toma uma peça do computador (detectada na linha 2625) o *Spectrum* usa esta sub-rotina para comentar a sua captura. Três vezes em quatro piscará o enquadramento, dizendo: "Estou frito!", e toca um pequeno trinado, seguido de três tristes notas musicais. À quarta vez apenas as três notas soarão, e o enquadramento piscará muito rapidamente para acusar a captura.

Finalmente, antes de chegarmos propriamente à listagem do programa, vamos mostrar o que acontece dentro dele quando o jogo está a decorrer. Esperamos que isto permita interpretar o seu funcionamento, por forma a que, se acharem bem melhorá-lo, saibam qual a secção que deve pôr-se em causa.

Na linha 70, MM (como se sabe) guarda a jogada da máquina. Ela tem de ser reposta a zero após cada jogada. A propósito, querendo ser o primeiro a jogar, faz-se apagar o GO TO 60 do fim da linha 30. UK conta o número de vezes que o programa circulou procurando uma jogada. Esta variável é definida por forma a assegurar que o rei não se moveu senão por uma boa razão e para encorajar o programa a não ceder a derrota muito facilmente. As próximas linhas (80 a 100) actuam nas intenções expressas pelo jogador, após ele se ter movido.

A linha 110 DIMensionia o vector T. Sempre que o faz enche o vector com zeros, preparando-o para armazenar as localizações de cada peça. U é a variável que conta o número de peças que o computador tem no tabuleiro. "Aguarde" aparece no topo do *écran* enquanto o computador procura uma jogada. A rotina nas

linhas 120 e 130 revê todas as casas do tabuleiro, na ordem ditada pelos elementos do vector S, armazenando-os no vector T segundo a ordem em que os encontra e contando-os. A variável KM (*Kingmarker*) é o "marcador do rei" e o computador usa-a para ter sempre sob controle a sua peça mais importante. KM é atribuída à localização do rei no fim da linha 120. A linha 130 completa o ciclo. A variável U, como já mencionámos, conta o número de peças no tabuleiro. Se U for inferior a três, o computador concede-lhe a vitória.

A acção decorre agora na linha 580, onde se atribui à variável Z o valor do marcador do rei. A rotina seguinte, desde essa linha até 810, verifica se o rei está em perigo. Faz esta verificação caso tenha ou não assinalado "xeque" após a conclusão da sua jogada. Se encontrou perigo, o computador vai para as linhas a partir de 1710 com o fim de desenhencilhar-se dele.

De volta ao princípio do programa, na linha 150, o computador revê as suas pedras e coloca o rei no fim delas, para que apenas se mova se não houver outras jogadas a fazer. A partir dos testes da rotina 580, o computador sabe que o seu rei não está em perigo; por isso não há razão para expor o rei a novo perigo, movendo-o, se ele está seguro onde se encontra.

A linha 170 escolhe na sequência das peças aquela com que o computador vai começar, escolhendo uma das três primeiras. Se o número escolhido é menor do que o número total de peças no tabuleiro, adiciona-se uma unidade ao valor de Q para determinar qual o elemento do vector T que será verificado para uma jogada em primeiro lugar.

A linha 190 atribui à variável Z (usada ao longo da sequência do computador como a localização "onde") o valor do elemento escolhido do vector T, e o programa salta três linhas para a sub-rotina 230. Um *beep* produzido por 230 mostra que o computador está a trabalhar. O *beep* deste local soa de tempos a tempos à medida que o computador procura uma jogada e, assim, o adversário sabe que ele ainda está a trabalhar, mesmo que pareça estar em dificuldades para encontrar o lance.

As cinco linhas seguintes mandam o programa para uma sub-rotina específica, dependendo da peça considerada. Dos

destinos destas sub-rotinas, pode descobrir-se quais as partes do programa que cuidam das acções "posso capturar" do computador:

Rainha — 820
Torre — 1060
Bispo — 1300
Cavalo — 1540
Peão — 2070

A linha 280 devolve-nos à linha 200, onde se verifica MM. Se for igual a 1, o programa vai para 2310 para fazer a jogada, depois volta a 40, que chama a rotina para reimprimir o tabuleiro antes de autorizar ao humano a introdução da sua jogada. Se MM for igual a zero, o programa continua até à próxima linha e, se Q for menor do que U, manda a acção de volta a 180, onde Q é incrementada e a pesquisa continua. Se Q não é menor do que U, isso significa que todas as peças, desde a primeira seleccionada aleatoriamente, foram verificadas para capturas.

Deste ponto o programa salta para 2180 (os planos que estudámos para ter um fluxo claro através do programa foram frustrados pelas exigências deste quando se chegou ao ponto da introdução no computador) onde se atribui a Q um número aleatório entre zero e quatro (com o uso duplo de RND servindo para desviar os números na direcção do extremo inferior). Este número é verificado para assegurar que não excede o número de peças disponível. Talvez se pretenda alterar o cinco nesta expressão para um número mais elevado, a fim de reduzir a previsibilidade com que o computador desenvolve certas jogadas iniciais. Qualquer número até nove, inclusive, será bastante adequado. Mais uma vez se adiciona uma unidade a Q na linha 2190 e se atribui a Z o valor da peça seleccionada. Os destinos das sub-rotinas indicarão onde se encontra o controle para as peças:

Peão — 2140
Cavalo — 1630

Bispo — 1420
Torre — 1180
Rainha — 940
Rei — 1710 (apenas se o rei não estiver em xeque e o número aleatório escolhido for menor do que .07, para desencorajar o rei de vagar).

O computador, uma vez conduzido à sub-rotina relevante, verifica se a *flag* MM ainda é zero. Se for, e caso todas as peças, desde a inicialmente escolhida aleatoriamente, não tenham sido verificadas, o programa vai para 2180 com o fim de escolher nova peça. Se MM tomou o valor um, o computador vai para a sub-rotina de 2310 para executar a jogada, voltando depois à linha 40 para a reimpressão do tabuleiro, antes de uma nova jogada do computador.

A linha seguinte verifica o valor da variável UK, e se este for maior do que oito sabe que toda a rotina foi verificada oito vezes e não foi encontrada nenhuma jogada adequada, indo por isso para 2060 para conceder a vitória.

A sequência seguinte faz realmente as jogadas do computador. A linha 2310 rejeita mover o rei nove vezes em dez, se este não estiver em xeque, através da reposição a zero de MM e regressando a 2180 para procurar outra peça para jogar. A linha 2312 é disparada se a peça a mover é um peão, que verifica se o programa não está a tentar mover o peão para trás e impede a execução de tal jogada. Se o rei branco se encontra no quadrado para o qual o computador ia mover-se, o humano está em xeque, e por isso aparece "Xeque!" impresso no *écran*, e o computador regressa à linha 190 para procurar outra captura.

Se a jogada contemplada passou três barreiras (e esta é uma das áreas do programa fácil de modificar se houver jogadas particulares no programa que se queira desencorajar ou simplesmente banir), a linha 2315 faz a jogada, e as linhas 2320 a 2330 dizem (usando uma linha aparentemente muito complexa para transformar o número do vector em coordenadas que reconheça)

que o lance foi efectuado. A linha 2340 faz o programa regressar ao ciclo inicial e coloca-nos agora na secção de impressão do tabuleiro.

Apesar da rotina para determinar perigo para o rei, o programa nem sempre é capaz de sair do xeque. Poderá considerar tal falhanço como um desejo de desistir ou, caso se sinta generoso, permitir ao programa permanecer em xeque por mais uma jogada. De igual modo, se o programa se movimenta ele próprio para xeque (uma possibilidade muito remota), considere isso como indicação de que o computador desistiu.

```

10 REM XADREZ PARA PRINCIPIANT
ES
20 PRINT "PASSE AO MODO "; FLA
SH 1; "C"; FLASH 0; "A SEGUIR"
PRIMA ENTER: INPUT A$: CLS: PR
INT "Obrigado, Aguarde."
30 GO SUB 2240: GO TO 50
40 GO SUB 2300
50 GO SUB 2350
60 GO SUB 2350
70 LET mm=0: LET uk=0
80 IF A$="P" THEN STOP
90 IF A$="T" THEN PRINT AT 0,0
: FLASH 1; "Troca de lados": GO S
UB 2120: GO SUB 2350: PRINT AT 0
: 0; " ": LET A$=""
100 IF A$="C" THEN COPY
110 DIM t(16): LET u=0: PRINT A
T 0,0: INK RND*5+2: PAPER 9: " Ag
uarde."
120 FOR q=1 TO 64: IF a(s(q))=
b AND a(s(q))<=r THEN LET u=u+
1: LET t(u)=s(q): IF a(s(q))=kb
THEN LET km=s(q)
130 NEXT q: IF u<3 THEN GO TO 2
060
140 GO TO 580
150 FOR q=1 TO u: IF a(t(q))=kb
THEN LET t(q)=t(u): LET t(u)=km
160 NEXT q
170 LET q=INT (RND*3)
180 IF q<u THEN LET q=q+1
190 IF 0<=u THEN LET z=t(q): GO
SUB 2330
195 IF 0>u THEN GO TO 2180
200 IF mm=1 THEN GO SUB 2310: G

```

```

0 TO 40
10 IF q<u THEN GO TO 180
20 GO TO 2180
30 BEEP .005: RND*30+20: IF a(z
)=qb THEN GO SUB 820
40 IF a(z)=rb THEN GO SUB 1060
50 IF a(z)=bb THEN GO SUB 1300
60 IF a(z)=nb THEN GO SUB 1540
70 IF a(z)=pb THEN GO SUB 2070
80 GO TO 200
90 IF a(x)=114 THEN PRINT AT 0
: 0: FLASH 1; "Xeque! ": LET
q=q+1: LET mm=0: GO TO 190
300 IF X-10*INT (X/10)=8 AND Z-
10*INT (Z/10)<>8 AND A(X)=E AND
RND<.96 THEN RETURN
340 LET ad=0
350 LET ay=1
360 LET ax=x+q(ay+ad)
370 IF ax<11 OR ax>88 THEN GO T
O 400
380 LET ap=a(ax)
390 IF ap=100 OR ap=r AND ad<28
AND RND>.2 OR ap=b AND ad>=28 A
ND RND>.5 THEN RETURN
395 IF ap<=e THEN GO TO 420
400 LET ay=ay+1
410 IF ay<9 THEN GO TO 360
420 LET ad=ad+7
430 IF ad<56 THEN GO TO 350
440 LET ay=1
450 LET ax=x+n(ay)
460 IF ax<11 OR ax>88 THEN GO T
O 480
470 IF a(ax)=n THEN RETURN
480 LET ay=ay+1
490 IF ay<9 THEN GO TO 450
500 LET ay=1
510 LET ax=x+k(ay)
520 IF ax<11 OR ax>88 THEN GO T
O 540
530 IF a(ax)=k AND RND>.1 THEN
RETURN
540 LET ay=ay+1
550 IF ay<9 THEN GO TO 510
555 IF (a(x+9)=p OR a(x-11)=p)
AND RND>.05 THEN RETURN
560 LET mm=1
570 RETURN
580 LET z=km
590 LET y=0
600 LET y=y+1

```

```

810 LET X=Z+N(Y)
820 IF X<11 OR X>88 THEN GO TO
840
850 IF A(X)=N THEN GO TO 1710
860 IF Y<8 THEN GO TO 600
870 LET D=0
880 LET Y=1
890 LET X=Z+Q(Y+D)
900 IF X<11 OR X>88 THEN GO TO
920
930 IF A(X)=B OR A(X)=100 OR A(
X)=R THEN GO TO 1710
940 IF A(X)<>E THEN GO TO 730
950 LET Y=Y+1
960 IF Y<8 THEN GO TO 670
970 LET D=D+7
980 IF D<56 THEN GO TO 660
990 LET X=Z+Q
1000 IF X>88 THEN GO TO 780
1010 IF A(X)=P THEN GO TO 1710
1020 LET X=Z-11
1030 IF X<11 THEN GO TO 150
1040 IF A(X)=P THEN GO TO 1710
1050 GO TO 150
1060 LET D=0
1070 LET Y=1
1080 LET X=Z+Q(Y+D)
1090 IF X<11 OR X>88 THEN GO TO
1100
1110 IF A(X)=42 OR A(X)>=BB AND
A(X)<=RB THEN GO TO 910
1120 IF A(X)>=B AND A(X)<=R THEN
GO SUB 290: IF MM=0 THEN GO TO
1130
1140 IF MM=1 THEN RETURN
1150 LET Y=Y+1
1160 IF Y<8 THEN GO TO 1080
1170 LET D=D+7
1180 IF D<28 THEN GO TO 1070
1190 RETURN
1200 LET D=0
1210 LET Y=1
1220 LET X=Z+R(Y+D)
1230 IF X<11 OR X>88 THEN GO TO
1240
1250 IF A(X)<>E THEN GO TO 1270
1260 IF AND<1 THEN GO SUB 290
1270 IF MM=1 THEN RETURN
1280 LET Y=Y+1
1290 IF Y<8 THEN GO TO 1200
1300 LET D=D+7
1310 IF D<28 THEN GO TO 1190
1320 RETURN
1330 LET D=0
1340 LET Y=1
1350 LET X=Z+B(Y+D)
1360 IF X<11 OR X>88 THEN GO TO
1370
1380 IF A(X)=42 OR A(X)>=BB AND
A(X)<=RB THEN GO TO 1390
1390 IF A(X)>=B AND A(X)<=R THEN
GO SUB 290: IF MM<>1 THEN GO TO
1400
1410 IF MM=1 THEN RETURN
1420 LET Y=Y+1
1430 IF Y<8 THEN GO TO 1320
1440 LET D=D+7
1450 IF D<28 THEN GO TO 1310
1460 RETURN
1470 LET D=0
1480 LET Y=1
1490 LET X=Z+B(Y+D)
1500 IF X<11 OR X>88 THEN GO TO
1510

```

```

1040 IF D<56 THEN GO TO 950
1050 RETURN
1060 LET D=0
1070 LET Y=1
1080 LET X=Z+R(Y+D)
1090 IF X<11 OR X>88 THEN GO TO
1100
1110 IF A(X)=42 OR A(X)>=BB AND
A(X)<=RB THEN GO TO 1150
1120 IF A(X)>=B AND A(X)<=R THEN
GO SUB 290: IF MM=0 THEN GO TO
1130
1140 IF MM=1 THEN RETURN
1150 LET Y=Y+1
1160 IF Y<8 THEN GO TO 1080
1170 LET D=D+7
1180 IF D<28 THEN GO TO 1070
1190 RETURN
1200 LET D=0
1210 LET Y=1
1220 LET X=Z+R(Y+D)
1230 IF X<11 OR X>88 THEN GO TO
1240
1250 IF A(X)<>E THEN GO TO 1270
1260 IF AND<1 THEN GO SUB 290
1270 IF MM=1 THEN RETURN
1280 LET Y=Y+1
1290 IF Y<8 THEN GO TO 1200
1300 LET D=D+7
1310 IF D<28 THEN GO TO 1190
1320 RETURN
1330 LET D=0
1340 LET Y=1
1350 LET X=Z+B(Y+D)
1360 IF X<11 OR X>88 THEN GO TO
1370
1380 IF A(X)=42 OR A(X)>=BB AND
A(X)<=RB THEN GO TO 1390
1390 IF A(X)>=B AND A(X)<=R THEN
GO SUB 290: IF MM<>1 THEN GO TO
1400
1410 IF MM=1 THEN RETURN
1420 LET Y=Y+1
1430 IF Y<8 THEN GO TO 1320
1440 LET D=D+7
1450 IF D<28 THEN GO TO 1310
1460 RETURN
1470 LET D=0
1480 LET Y=1
1490 LET X=Z+B(Y+D)
1500 IF X<11 OR X>88 THEN GO TO
1510

```



```

1460 IF a(x) <> e THEN GO TO 1510
1470 IF AND >.05 THEN GO SUB 290:
1480 IF mm <> 1 THEN GO TO 1510
1490 IF mm=1 THEN RETURN
1500 LET y=y+1
1510 IF y<8 THEN GO TO 1440
1520 LET d=d+7
1530 IF d<28 THEN GO TO 1430
1540 RETURN
1550 LET y=1
1560 LET x=z+n(y)
1570 IF x<11 OR x>88 THEN GO TO
1580
1590 IF a(x)=42 THEN GO TO 1600
1600 IF a(x)>=b AND a(x)<=r THEN
GO SUB 290:
1610 IF mm=1 THEN RETURN
1620 LET y=y+1
1630 IF y<8 THEN GO TO 1550
1640 RETURN
1650 LET y=0
1660 LET x=z+n(INT (AND*8+1))
1670 IF x<11 OR x>88 THEN GO TO
1680
1690 IF a(x)=42 THEN GO TO 1640
1700 LET y=y+1
1710 IF a(x)=e THEN GO SUB 290
1720 IF mm=1 OR y>20 THEN RETURN
1730
1740 GO TO 1640
1750 LET yk=1
1760 LET z=km
1770 LET x=z+k(yk): LET x1=x
1780 IF x<11 OR x>88 THEN GO TO
1790
1800 IF a(x)=42 OR a(x)>55 AND a
(x)<85 THEN GO TO 2030
1810 LET z=x
1820 LET y=0
1830 LET y=y+1
1840 LET x=z+k(y)
1850 IF x<11 OR x>88 THEN GO TO
1860
1870 IF a(x)=k THEN GO TO 2030
1880 IF y<8 THEN GO TO 1772
1890 LET y=0
1900 LET y=y+1
1910 LET x=z+n(y)
1920 IF x<11 OR x>88 THEN GO TO
1930
1940 IF a(x)=n THEN GO TO 2030
1950 IF y<8 THEN GO TO 1790

```

```

1960 LET d=0
1970 LET y=1
1980 LET x=z+q(y+d)
1990 IF x<11 OR x>88 THEN GO TO
2000
2010 IF a(x)=b AND d>=28 OR a(x)
=100 OR a(x)=r AND d<28 THEN GO
TO 2030
2020 IF a(x) <> e THEN GO TO 1920
2030 LET y=y+1
2040 IF y<8 THEN GO TO 1850
2050 LET d=d+7
2060 IF d<56 THEN GO TO 1850
2070 LET x=z+q
2080 IF x>88 THEN GO TO 1970
2090 IF a(x)=p THEN GO TO 2030
2100 LET x=z-11
2110 IF x<11 THEN GO TO 2000
2120 IF a(x)=p THEN GO TO 2030
2130 LET x=x1: LET z=km
2140 LET mm=1
2150 GO SUB 2310: GO TO 40
2160 LET yk=yk+1
2170 LET z=km
2180 IF yk<9 THEN GO TO 1720
2190 PRINT AT 0,0: FLASH 1: "Eu d
esisto, campeao!": STOP
2200 LET x=z+q
2210 IF a(x)>=b AND a(x)<=r THEN
LET mm=1: IF a(x)=p AND AND<.2
THEN LET mm=0
2220 IF mm=1 THEN RETURN
2230 LET x=z-11
2240 IF a(x)>=b AND a(x)<=r THEN
LET mm=1: IF a(x)=p AND AND<.2
THEN LET mm=0
2250 RETURN
2260 IF z-10+(INT (z/10))=7 AND
a(z-1)=e AND a(z-2)=e AND (a(z-1
3)=e OR a(z-13)=42) AND (a(z+7)=
e OR a(z+7)=42) THEN LET x=z-2:
LET mm=1: RETURN
2270 IF z=12 THEN GO TO 2155
2280 IF a(z-1)=e AND a(z-12)<98
AND a(z+8)<98 THEN LET x=z-1: LE
T mm=1: RETURN
2290 IF z=12 AND a(z)=11 AND a(2
0)<98 THEN LET x=10: LET mm=1: R
ETURN
2300 IF AND<.05 AND a(z-1)=e THE
N LET x=z-1: LET mm=1
2310 RETURN

```

```

2180 LET q=INT (RND*RND*5): IF q
>U THEN GO TO 2180
2190 IF q<U THEN LET q=q+1
2200 LET z=t(q)
2210 IF a(z)=pb THEN GO SUB 2140
2220 IF a(z)=nb THEN GO SUB 1630
2230 IF a(z)=bb THEN GO SUB 1420
2240 IF a(z)=rb THEN GO SUB 1180
2250 IF a(z)=qb THEN GO SUB 940
2260 IF a(z)=kb AND a$<>"X" AND
RND<.07 THEN GO SUB 1710
2270 IF mm=0 AND q<U THEN GO TO
2190
2280 IF mm=1 THEN GO SUB 2310: G
O TO 40
2300 LET uk=uk+1: IF uk>8 THEN G
O TO 2050
2305 GO TO 2180
2310 IF a(z)=kb AND a$<>"X" AND
RND>.1 THEN BEEP .05,-3: LET mm=
0: GO TO 2180
2312 IF a(z)=pb AND ((x-10*INT (
x/10)>z-10*INT (z/10) OR ABS (x-
z)>11)) THEN LET mm=0: GO TO 218
0
2314 IF a(x)=k THEN PRINT AT 0,0
: "Xaque!": LET mm=0
: LET u=u+1: GO TO 190
2315 LET a(x)=a(z): LET a(z)=e
2320 PRINT AT 0,0: "Movi de "
2330 LET fz=INT (z/10): PRINT CH
R$ (fz+64);z-10*fz: " para " : LE
T fx=INT (x/10): PRINT CHR$ (fx+
64);x-10*fx
2340 RETURN
2350 IF mm=0 THEN GO TO 2370
2360 BEEP 1,RND*10: BEEP 1,10+RN
D*10: BEEP 1,-RND*10
2370 PRINT AT 0,0: "
";AT 4,0: GO SU
B 2450
2380 FOR x=8 TO 1 STEP -1
2390 PRINT TAB 10; INVERSE 1;x;
INVERSE 0;
2400 FOR y=10 TO 80 STEP 10
2410 IF a(y+1)=pb THEN LET a(y+1
)=qb
2420 IF a(y+8)=p THEN LET a(y+8)
=100
2430 PRINT CHR$ a(x+y);
2440 NEXT y: PRINT INVERSE 1;x;
NEXT x: LET mm=0

```

```

2450 PRINT TAB 10; INVERSE 1;" A
BCDEFGH "
2460 RETURN
2470 IF qk<8 THEN GO TO 2490
2480 IF a$<>"X" THEN RETURN
2490 GO TO 2050
2570 INPUT "DE (LETRA, NUMERO)?
":a$: BEEP .008,-3
2580 IF LEN a$<>2 THEN GO TO 257
0
2590 INPUT "DE ";(a$);" PARA ? "
:b$: BEEP .008,-10
2595 IF LEN b$<>2 THEN GO TO 259
0
2600 LET x=10*(CODE a$-64)+VAL a
$(2)
2610 LET y=10*(CODE b$-64)+VAL b
$(2)
2620 INPUT "      Prima X - xaque
      C - copiar
      o tabuleiro      T - para t
      rocar lados      P - para p
      arar o jogo      ENTER - para c
      ontinuar" a$
2625 IF a(y)>=66 AND a(y)<=84 TH
EN GO SUB 3310
2630 BEEP .008,-3: LET a(y)=a(x)
: LET a(x)=46: RETURN
2640 RANDOMIZE : LET mm=0: LET a
$=""
2650 DIM a(99): DIM r(28): DIM b
(28): DIM n(8): DIM q(56): DIM k
(8): DIM z(88): DIM s(64): DIM t
(16)
2660 LET p=112: LET r=116: LET n
=99: LET b=98: LET q=100: LET k=
114: LET e=46
2670 LET pb=80: LET rb=84: LET n
b=67: LET bb=66: LET qb=68: LET
kb=82
2680 FOR z=1 TO 99: LET a(z)=42:
NEXT z
2690 FOR z=1 TO 64: READ x: READ
y: LET a(x)=y: NEXT z
2700 DATA 18,84,28,67,38,66,48,6
8
2710 DATA 58,82,68,66,78,67,88,8
4
2720 DATA 17,80,27,80,37,80,47,8
0
2730 DATA 57,80,67,80,77,80,87,8
0

```



```

740 DATA 16,46,26,46,36,46,46,4
750 DATA 56,46,66,46,76,46,86,4
760 DATA 15,46,25,46,35,46,45,4
770 DATA 55,46,65,46,75,46,85,4
780 DATA 14,46,24,46,34,46,44,4
790 DATA 54,46,64,46,74,46,84,4
800 DATA 13,46,23,46,33,46,43,4
810 DATA 53,46,63,46,73,46,83,4
820 DATA 12,112,22,112,32,112,4
830 DATA 52,112,62,112,72,112,8
840 DATA 11,116,21,99,31,98,41,
850 DATA 51,114,61,98,71,99,81,
860 FOR z=1 TO 8: READ n(z): NE
870 DATA 19,-19,21,-21,-8,8,12,
880 FOR z=1 TO 28: READ r(z): N
890 DATA 10,20,30,40,50,60,70
900 DATA -1,-2,-3,-4,-5,-6,-7
910 DATA -10,-20,-30,-40,-50,-6
920 DATA 1,2,3,4,5,6,7
930 FOR z=1 TO 28: READ b(z): N
940 DATA -11,-22,-33,-44,-55,-6
950 DATA 11,22,33,44,55,66,77
960 DATA 9,18,27,36,45,54,63
970 DATA -9,-18,-27,-36,-45,-54
980 RESTORE 2890
990 FOR z=1 TO 56: READ q(z): N
1000 FOR z=1 TO 8: READ k(z): NE
1010 DATA 1,11,9,10,-10,-9,-11,-
1020 FOR z=1 TO 64: READ s(z): N

```

```

3030 DATA 46,56,66,66,47,57,45,5
3040 DATA 37,67,35,65,28,78,27,7
3050 DATA 44,54,26,76,38,68,17,8
3060 DATA 18,68,34,64,25,75,16,8
3070 DATA 48,24,74,15,85,14,64,4
3080 DATA 53,33,63,23,73,52,42,6
3090 DATA 32,63,13,72,22,12,82,4
3100 DATA 51,31,61,21,71,11,81,5
3110 PAPER 1: BORDER 1: INK 7: C
3120 FOR z=11 TO 88: LET z(z)=a(
3130 FOR z=11 TO 88: LET x=z-10*
3140 IF x=0 OR x=9 THEN GO TO 31
3150 LET a(z)=z(z+9-x*2)
3160 NEXT z
3170 FOR z=11 TO 88: LET m=a(z)
3180 IF m>=b THEN LET a(z)=a(z)+
3190 IF m<=rb AND m>=bb THEN LET
3200 NEXT z
3210 RETURN
3220 LET comment=INT (RND*4): BC
3230 RND*5+2: GO SUB 3330+10*com
3240 BEEP 1,1: BEEP 1,2: BEEP 1,
3250 BORDER 1: RETURN
3260 PRINT AT 0,0;"Sela Jogada!"
3270 GO SUB 3370: RETURN
3280 PRINT AT 0,0;"Grande lance!"
3290 GO SUB 3370: RETURN
3300 PRINT AT 0,0;"Estou frito!"
3310 FOR i=1 TO 10: BEEP .05,i:
3320 PRINT AT 0,0;"
3330 RETURN

```

Eis alguns aspectos do tabuleiro nas fases iniciais do jogo de XADREZ PARA PRINCIPIANTES:

Aguarde

```

      ABCDEFGH
      TCDBRB.T
      PPP.,PPP
      ...P..C
      ...P...
      ...C.P...
      P.P.,PPP
      t.bdrbct
      ABCDEFGH
  
```

Aguarde

```

      ABCDEFGH
      T.BDRB.T
      PP.,PPP
      C.P.,P.C
      ...P..C
      ...P...
      ...C.P...
      P.P.,PPP
      t.bdrbct
      ABCDEFGH
  
```

Posição atingida a meio de outro jogo:

Aguarde

```

      ABCDEFGH
      T.BDR..T
      .P.C...
      .P.,P.P
      .P.PC.P
      .P.,P.P
      .P.,P.P
      .P.,P.P
      tCbdrb.t
      ABCDEFGH
  
```

É possível modificar o programa no sentido de jogar "contra ele próprio", conseguindo que troque de lados após cada jogada. Também imprimirá uma cópia permanente do jogo imprimindo para isso o tabuleiro após cada segunda jogada. Para transformar o programa em "autxadrez", deve-se modificá-lo por forma a que o seu início seja o seguinte:

```

10 REM XADREZ PARA PRINCIPIANT
ES
15 LET pr=1
20 PRINT "LIGUE "; FLASH 1;"CA
PS LOCK"; FLASH 0;" A SEGUIR";
PRIMA ENTER": INPUT A$: CLS : PR
INT "Obrigado,Aguarde."
30 GO SUB 2640: GO TO 60
40 GO SUB 2350
70 LET mm=0: LET uk=0
80 IF a$="P" THEN STOP
85 LET a$="T"
90 IF a$="T" THEN PRINT AT 0,0
; FLASH 1;"Troca de lados": GO S
UB 3120: GO SUB 2350: PRINT AT 0
,0;"
100 IF pr=0 THEN COPY : LET pr=
1: GO TO 110
105 LET pr=0
  
```

Se não quiser uma listagem impressa, apague as linhas 15, 100, e 105.

Eis o início duma partida jogada pelo programa modificado:

```

      ABCDEFGH
      TCDBRBCT
      P.P.P.P.P.P
      .....
      .....
      .....
      P.P.P.P.P.P
      tCbdrbct
      ABCDEFGH
  
```


A	B	C	D	E	F	G	H
T	C	B	D	A	B	C	T
P	P	P	.	P	P	P	P
.	.	.	P
.	.	.	P
.	.	.	P
.	.	.	P
P	P	P	.	P	P	P	P
T	C	B	D	A	B	C	T
A	B	C	D	E	F	G	H

A	B	C	D	E	F	G	H
T	C	B	D	A	B	C	T
P	P	.	P	.	P	.	P
.	.	.	P
.	.	.	P
.	.	.	P
.	.	.	P
P	P	.	P	.	P	.	P
T	C	B	D	A	B	C	T
A	B	C	D	E	F	G	H

A	B	C	D	E	F	G	H
T	C	B	D	A	B	C	T
P	P	.	P	.	P	.	P
.	.	.	P
.	.	.	P
.	.	.	P
.	.	.	P
P	P	.	P	.	P	.	P
T	C	B	D	A	B	C	T
A	B	C	D	E	F	G	H

A	B	C	D	E	F	G	H
T	C	B	D	A	B	C	T
P	P	.	P	.	P	.	P
.	.	.	P
.	.	.	P
.	.	.	P
.	.	.	P
P	P	.	P	.	P	.	P
T	C	B	D	A	B	C	T
A	B	C	D	E	F	G	H

A	B	C	D	E	F	G	H
T	.	B	D	A	B	C	T
P	P	.	P	.	P	.	P
.	.	.	P
.	.	.	P
.	.	.	P
.	.	.	P
P	P	.	P	.	P	.	P
T	.	B	D	A	B	C	T
A	B	C	D	E	F	G	H

A	B	C	D	E	F	G	H
T	.	B	D	A	B	C	T
P	P	.	P	.	P	.	P
.	.	.	P
.	.	.	P
.	.	.	P
.	.	.	P
P	P	.	P	.	P	.	P
T	.	B	D	A	B	C	T
A	B	C	D	E	F	G	H

A	B	C	D	E	F	G	H
T	.	B	D	A	B	C	T
P	P	.	P	.	P	.	P
.	.	.	P
.	.	.	P
.	.	.	P
.	.	.	P
P	P	.	P	.	P	.	P
T	.	B	D	A	B	C	T
A	B	C	D	E	F	G	H

A	B	C	D	E	F	G	H
T	.	B	D	A	B	C	T
P	P	.	P	.	P	.	P
.	.	.	P
.	.	.	P
.	.	.	P
.	.	.	P
P	P	.	P	.	P	.	P
T	.	B	D	A	B	C	T
A	B	C	D	E	F	G	H

```

ABCEDEFGH
T,BDRBCT
PP.....PP
.....PC...
.....P.P..
.....P.P..
.....P.P..
PP.....PP
tc bdr bct
ABCEDEFGH

```

```

ABCEDEFGH
T,BDRBCT
PP.C...PP
.....P.P..
.....P.P..
.....P.P..
PP.C...PP
tc bdr bct
ABCEDEFGH

```

```

ABCEDEFGH
T,BDRB.T
PP.CC.PP
.....P.P..
.....P.P..
.....P.P..
PP.....PP
tc bdr bct
ABCEDEFGH

```

A propósito, correr os jogos em autxadrez é, como já dissemos, uma boa maneira de descobrir e corrigir erros do programa, uma vez que eventualmente será testada a maior parte dos destinos das instruções GO TO e GO SUB. Elimine-se, querendo, a possibilidade de fazer cópias impressas, e deixe-se o computador ligado todo o dia, jogando partida após partida. Se ele parar em qualquer ponto (outro que não seja conceder a vitória a si próprio), saberá mais ou menos onde o erro ocorreu.

O programa foi ainda um pouco modificado para permitir a

impressão, de tempos a tempos, de posições do tabuleiro em vez de imprimir após cada segundo lance.

Para obter essas figuras aleatórias, o início do programa deverá ser:

```

100 IF PR=0 THEN LET PR=1
101 IF PR=1 AND RND<.1 THEN COP
Y : GO TO PR=0
105 LET PR=0

```

Eis uma selecção de «retratos» duma partida jogada desta nova maneira:

```

ABCEDEFGH
GT,CBDRBCT
PP.....PP
.....P.P..
.....P.P..
.....P.P..
PP.....PP
tc bdr bct
ABCEDEFGH

```

```

ABCEDEFGH
GT,BDRBCT
PP.....PP
C.P.P.P..
.....P.P..
C.P.P.P..
PP.....PP
tc bdr bct
ABCEDEFGH

```

```

ABCEDEFGH
GT,BDRB.T
P.....PP
CPP.P.P.C
.....P.P..
C.P.P.P..
PP.....PP
tc bdr bct
ABCEDEFGH

```


Aventura / simulações

A VINGANÇA NO CASTELO DO TERROR

Nenhuma colecção de programas de jogos estaria completa sem uma aventura. Decidimos que a aventura neste livro deveria ser um dos programas mais importantes; por isso escrevemos VINGANÇA NO CASTELO DO TERROR, jogo de aparência formidável (em termos de comprimento).

O mérito de terem produzido o primeiro programa de aventuras vai para Crowther e Woods, da Universidade de Standford, que o escreveram e que lhe deram simplesmente o título "Aventura". O computador servia de suporte a um número de terminais ligados num sistema *time-sharing*, sendo a saída impressa em longos rolos de papel usando um aparelho semelhante a uma máquina de telex.

"Aventura" tornou-se imensamente popular, e em breve cópias piratas corriam pela América e também na Grã-Bretanha.

Os fãs dos computadores, à medida que iam pondo as mãos no programa, modificavam-no e melhoravam-no, moldando versões radicalmente diferentes do original.

No entanto, vinte anos depois de "Aventura" ter sido posta a funcionar pela primeira vez, algumas empresas de *software* norte-americanas ainda vendem um programa com o rótulo de ser a "Aventura original".

Escrever um programa de aventuras é um exercício fascinante. Na essência inventa-se e planeia-se um universo completo e logicamente coerente.

Por logicamente coerente queremos dizer que numa "aventura" construída correctamente o jogador pode atravessar uma ponte para passar por uma estrada, e ao regressar encontrará ainda a ponte, ou haverá uma razão convincente para a sua ausência, tal como: "Os invasores de Epsilon IV levaram-na para o museu do seu planeta com um transportador de matéria". Aventuras aleatórias são frustrantes e oferecem desafios pouco duradouros ao jogador.

Na VINGANÇA NO CASTELO DO TERROR a acção tem

lugar num castelo onde existe um certo número de monstros, incluindo um "feiticeiro", um "papão que cospe fogo" e o odioso "Guardião da Lagoa Negra", que vagueiam pelo castelo durante o jogo e que têm de ser combatidos (combate onde o humano tem apenas 50% de hipóteses de ser bem sucedido nas suas tentativas de fugir ao perigo). Além disso, há ainda quatro arcos que contêm boas e más surpresas, uma garrafa com uma poção mágica que tanto pode ser benéfica como malévola, e assim por diante.

É improvável que num único jogo se encontrem todos os monstros e surpresas.

Não se fornece qualquer mapa do castelo. Parte da diversão de jogar a "aventura", seja com um computador ou num jogo de "faz de conta", é tentar deduzir as relações geográficas das várias "divisões" encontradas quando se vagueia por tais ambientes. O castelo, neste programa, foi planeado com alguma minúcia, e quisemos ter a certeza de que as pistas oferecidas pelas instruções PRINT coincidem rigorosamente com a relação (geográfica) existente entre os diversos quartos do castelo. A intenção é encorajar o jogador a elaborar um mapa do castelo à medida que o atravessa e para ensinar, através de um exemplo, como são construídos os "ambientes da aventura".

Logo que se considere ter um mapa que corresponda razoavelmente ao que se encontra codificado neste programa, pode-se "vaguear" pelo castelo e verificá-lo.

O objectivo do jogo é simples e relaciona-se com uma alteração que incluímos para tornar o desenho do mapa mais difícil do que seria noutras condições.

Todas as instruções são dadas no programa em termos de direcções (norte, sul, leste e oeste), e todas as portas estão voltadas para uma delas.

Deve-se começar a execução do mapa marcando as quatro direcções numa folha de papel. Começa-se a aventura no topo do papel em direcção ao fundo (ou seja, em direcção ao sul). A primeira instrução inclui o seguinte:

Encontra-se na entrada de um castelo de aspecto antigo e misterioso. Esta na ala NORTE do castelo e, olhando para SUL pela estrutura degradada, repára no portal de entrada, aberto e sem guarda.

Que quer fazer agora?

A última linha, "Que quer fazer agora?", vai seguir-se a cada descrição do ambiente — à parte algumas perguntas, tais como: "Quer beber a poção?", que exige uma resposta "S" (sim) ou "N" (não) — e pode ser atendida com uma de sete respostas.

Introduzindo-se isto: O computador compreenderá isto:

"N"	Dirija-se para norte
"S"	Dirija-se para sul (é a resposta que tem de se dar ao primeiro quadro)
"E"	Dirija-se para este
"W"	Dirija-se para oeste (West)
"L"	Lute
"F"	Fuja

Qualquer outra resposta apenas provocará a limpeza do *écran*, que será depois novamente impresso ("Q" também pode ser usado nesta altura para desistir do jogo).

"Jogos de faz de conta" jogados com pessoas e um árbitro (como, por exemplo, um jogo popular em Inglaterra mas pouco conhecido em Portugal e intitulado "Chefe das Masmorras") seguem geralmente certas convenções que ditam que o poder e personalidade de uma determinada personagem sejam atribuídos no princípio do jogo tirando um número aos dados e lendo as características (tal como "subtileza") a partir de uma tabela. Quando duas personagens entram em conflito (os jogadores de "Chefe das Masmorras" chamam a este ponto alto do jogo "a embrolhada") é necessário mais um lançamento de dados e a consulta à tabela para determinar o resultado do combate.

Simplificando o sistema para que não seja sobrecarregado com regras difíceis de compreender antes de apreciar este programa, restringimo-nos ao essencial mas assegurando que o *Spectrum* tem todo o trabalho, tanto da "geração" da sua personalidade e da dos monstros que irá encontrar, como para resolver alguma briga que o jogador tenha de defrontar.

O roteiro e objectivos do jogo são simples. O jogador, na pele de um intrépido explorador, descobriu as ruínas de um castelo degradado. Com grande intrepidez, atravessa o portal da entrada, situado na ala norte, e vê que ele está aberto.

Desde que entre no castelo por esse portal, não poderá sair por ele. Terá de explorar o castelo e tentar encontrar outra saída. Enquanto se encontra na ala de entrada, "ornamentada com ricos tapetes", recorda-se de uma velha lenda que se conta acerca deste castelo. O último dono era, há 300 anos, um poderoso feiticeiro que com a sua força mortal criou uma legião de horrores para guardar de intrusos o seu lar. Além do portal norte, que se encontra agora magicamente selado para sempre, só se pode sair através da Lagoa Negra, que se encontra sob o castelo, mas onde se chega pelo interior através de uma porta vulgaríssima. Há rumores de que o feiticeiro procurou proteger mais ainda o seu castelo criando uma desagradável criatura da qual não se sabe nada senão o nome, "Guardião da Lagoa Negra".

A personalidade do explorador neste jogo é condensada em três atributos: força mágica, energia e sabedoria. A cada criatura que rasteja eternamente pelas salas em ruína também são concedidos estes três atributos.

O estado corrente da sua personalidade aparece no *écran* como se segue:

```
PAULO, OS SEUS atributos são:
MAGIA: 100
FORÇA: 100
SABEDORIA: 100
```

Quando o explorador encontra um monstro e decide combatê-lo (ou tenta fugir e não consegue), luta apenas em termos de

um atributo escolhido à sua vontade. Geralmente, quanto maior for a diferença a favor dele entre os seus atributos e os do monstro, mais provável é a sua vitória.

Portanto, se a sua MAGIA é 20 e a do monstro é apenas 4, e a diferença entre os outros atributos for menor, será aconselhável que escolha a MAGIA como arma.

```
PAULO, os seus atributos são:  
MAGIA: 14  
FORÇA: 11  
SABEDORIA: 13
```

Esta na sala de entrada, ornamentada com ricos tapetes. Portas apontam para leste e sul, e um portal aberto aponta para oeste.

Na sala encontra-se um demônio flamejante. Tem uma sabedoria de 15, uma força de 19, e um poder mágico de 14.

Que quer fazer agora?

```
PAULO, os seus atributos são:  
MAGIA: 14  
FORÇA: 11  
SABEDORIA: 13
```

Encontra-se na Câmara Real. Esta pequena sala tem como principal característica uma estatua ornamental da deusa da Lua, num pedestal situado no canto nordeste. Tem portas conduzindo a sul, leste e oeste.

Na sua frente esta uma arca marcada com o número 1

Quer abri-la (S/N)?

Finalmente, a Sabedoria:
Você: 15 Guardião 19
A diferença é 4

O combate acarretou uma penalização de 17 pontos de sabedoria
E você perdeu:
17 pontos de sabedoria

No fim da batalha crucial você tem:
MAGIA: 25
FORÇA: 9
SABEDORIA: 0

Conseguiu, oh herói destes negros e perigosos tempos.
EU vos armo cavaleiro **Drum** PAULO

Talvez tudo isto pareça um pouco complicado, mas não há que ter grandes preocupações. O programa faz quase todo o trabalho e guia o explorador através do jogo, ignorando ou rejeitando as informações mal introduzidas.

Cada luta oferece a oportunidade de melhorar o atributo com o qual está a competir; não o conseguindo, ter-se-á de sofrer as consequências pela falta de decisão, perdendo pontos de atributo. Perdendo-se todos os pontos, o jogo termina. É necessário adquirir o maior número possível de pontos do atributo para o Encontro Final. Quando se descobrir onde fica a Lagoa Negra (para causar alguma frustração, pode situar-se em um de dois lugares, por isso os mapas desenhados nunca são 100 % adaptáveis de jogo para jogo), haverá três batalhas, uma para cada atributo, com o Guardião da Lagoa Negra.

Algumas das arcas que se encontram pela casa contêm "ouro de dragão", que vale a pena adquirir porque com ele se compram pontos adicionais depois de avistado o Guardião, mas antes do início do combate.

É indispensável sair desta série final de batalhas pelo menos com 10 pontos no conjunto dos três atributos, para se poder sobreviver.

Eis como funciona cada luta: depois da escolha do atributo com o qual se deseja fazer a guerra, o computador calculará e

apresentará a diferença entre o número de pontos dos atributos do explorador e do monstro. Se, por exemplo, a decisão fosse lutar com SABEDORIA e o número de pontos de sabedoria do explorador fosse 15 e o do monstro 2, o computador imprimiria (linha 5240) "a desigualdade vale 13", seguido de "com vantagem sua", o que significava que o seu número de pontos era o mais elevado.

A seguir, e à semelhança dos jogos "de faz de conta", que, como já salientámos, obrigam a muitos lançamentos de dados, lança-se um dado e o número apurado corresponde ao "custo" do encontro. É gerado um número aleatório entre um e a diferença dos dois números de pontos. Este é então comparado entre o número de pontos do atributo do explorador e o do monstro. O vencedor é aquele cujo número de pontos do atributo é mais próximo da diferença. (Pode-se, por exemplo, mudar isto por forma a que o vencedor seja o que tem o número de pontos mais afastado do número aleatório.) Se a diferença entre os números de pontos dos atributos for apenas de uma unidade, o número aleatório tem de ser um, por isso o jogador com o menor número de pontos do atributo ganhará (há uma sugestão nesta linha para os mais observadores que os ajudará a ganhar mais do que a justa parte das lutas). Deixo-vos a tarefa de descobrir quem ganha na hipótese de os pontos dos atributos serem iguais.

Agora, ganhar e perder são duas coisas bastante diferentes neste jogo. O custo e as recompensas dependem do atributo que se seleccionou e da pontuação actual desse atributo.

O explorador selecciona o atributo com o qual deseja lutar introduzindo uma letra (M para MAGIA, W para SABEDORIA, S para ENERGIA) e o computador usará esta informação na rotina da linha 5340 para determinar a penalização ou recompensa do encontro. Se lutou com MAGIA e saiu derrotado, perde o valor total dos pontos que lhe saíram no dado se tiver pelo menos esse número de pontos à partida. Ou seja, tendo-se apenas dois pontos da MAGIA, e tendo perdido uma luta com uma penalização de três pontos, não se perdeu nenhum ponto. Lembramos mais uma vez que isto é mais simples na prática (uma vez que o computador resolverá se a penalização deve ou

não ser paga) do que parece quando se lê. Perdendo-o num encontro quando se tiver escolhido ENERGIA como atributo para lutar, ser-se-á penalizado de acordo com o dado se tiver pelo menos esse número de pontos à partida. Perdendo-se enquanto combate com SABEDORIA, só se perde metade dos pontos assinalados pelo dado, arredondados, por defeito, em relação ao número inteiro mais próximo (por isso, se a penalização for um e o jogador perde, não custa nada, pois metade de um arredonda-se para zero).

Ganhar é mais aliciante. Numa luta ganha-se mais do que se perde quando se usa MAGIA, que dá a oportunidade de obter a dobrar o valor do dado.

Com ENERGIA nada se obtém por uma vitória (hipótese mais favorável do que perder, mas não o bastante; por isso é aconselhável evitar a luta com ENERGIA, pois na maior parte das vezes fica-se em inferioridade frente aos musculosos monstros), e uma vitória com SABEDORIA vale metade do valor do dado, tal como uma derrota equivale a uma penalização de metade do valor saído no dado.

O programa segue uma estrutura predeterminada, por isso é relativamente fácil chegar às diversas secções se se desejar modificá-lo. Uma vez descoberto o mapa codificado neste programa, pode pretender-se alterá-lo e da maneira que descreveremos na altura própria, por forma a permitir usar um mapa mais complicado embora povoado com os mesmos monstros e seguindo as mesmas regras. Dominada a técnica de alteração de mapas, ter-se-á provavelmente o desejo de adaptá-lo a algum mapa original da autoria do jogador.

O programa está estruturado da seguinte maneira:

- | | |
|------------------|--|
| Linhas 20-170 | Relatório (qual a sala onde se encontra, local onde estão situadas as saídas e similares) e ciclo principal do programa. |
| Linhas 2000-2050 | Descrições dos monstros. |
| Linhas 4000-4940 | Confronto Final (a Lagoa Negra, as três lutas finais e o final). |
| Linhas 5000-5620 | Acções (o jogador introduz a sua direcção |

de movimento, ou expressa o seu desejo de fugir ou de lutar; haverá lutas nesta secção do programa, e serão determinadas as recompensas e os castigos).

Linhas 7000-7500 Conteúdos (esta secção armazena coisas como as quatro arcas e a garrafa de poção mágica; é executado aleatoriamente a partir do fim da secção de acções do programa).

Linhas 8000-8140 Descrição das salas, e chamada da sub-rotina dos monstros.

Linhas 9000-9670 Inicializações.

Linhas 9900-9990 Rotina de som e de espera (chamada repetidamente através do programa, para juntar um pouco de colorido e dar tempo para ler o que está no *écran* antes de este ser limpo).

A parte mais importante deste programa é a chave de qualquer programa de aventuras é o mecanismo que "mantém o mundo" no lugar, determinando a relação entre várias partes do sistema e verificando potenciais movimentos entre elementos do sistema para assegurar a sua legalidade. De duas das salas do nosso castelo avista-se o Jardim Interior mas não se pode atravessá-lo para passar para a outra sala. O jardim foi incluído porque, além de tornar a descrição do castelo mais interessante, dá aos fazedores-de-mapas uma pista que os deverá ajudar a elaborar os seus mapas do castelo.

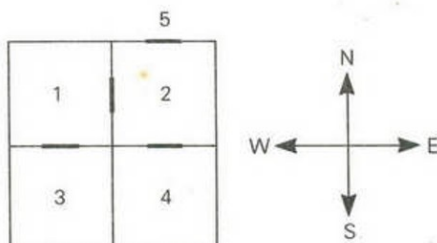
Vamos agora explicar como este programa controla os componentes do sistema, mas não queremos que este ponto seja estudado muito aprofundadamente antes do jogo, porque isso retiraria todos os motivos de interesse. Repare-se nas linhas 9570 a 9680. Aqui, as instruções DATA contêm muita informação útil (e há alguns artifícios nelas, vantajosos no caso de querer descobrir o que está no mapa dispondo simplesmente desta

informação). Veja-se a quarta instrução na linha 9600. Os dois primeiros elementos do vector constituem o número da sala a norte daquela que está guardada pela instrução DATA. Nada existe nessa posição, por isso não há saída virada a norte a partir desse quarto. Os dois elementos seguintes dizem que a sala sete é para sul; os dois elementos imediatos indicam uma saída virada a leste que conduz à sala dois do castelo e os dois elementos que se seguem indicam ainda que a sala cinco está virada a sul. O último zero destina-se a armazenar a informação respeitante ao conteúdo das salas.

Pode-se usar este tipo de referência no interior das instruções DATA para armazenar qualquer mapa. Se nos encontrarmos na sala referida pela linha 9600 e dissermos que queremos ir para norte, o computador sabe que só precisa de olhar para os dois primeiros elementos do vector que "guarda" essa sala. Se forem ambos zero, o computador dá a informação "SEM SAÍDA" e pede uma nova direcção. Se encontrar aí outro número, a variável a que é atribuído o número da sala (que é Z na maior parte do programa) passa a ter o VAL (valor) desses elementos. Este programa faz um uso intensivo de simples técnicas de divisão de vectores em fatias (*string-slicing*) que o Basic Sinclair proporciona, guardando os atributos dos monstros nos elementos do vector literal M\$, e os atributos do jogador em J\$.

É fácil modificá-los à medida que o jogo progride. Se derrotarmos um Papão numa sala e lhe reduzirmos a MAGIA a 9, quando (e se) o encontrarmos novamente algures no castelo, o nível de MAGIA do monstro estará à taxa reduzida. Encontrando-se um monstro bastantes vezes, pode-se acabar por "apagá-lo do mapa". O facto de se manter sempre os mesmos monstros, mesmo que sejam encontrados a rastejar em diferentes salas, é outra característica que ajuda a manter a consistência do universo da aventura.

Tendo um simples universo da aventura de apenas quatro salas, pode-se juntá-las como se segue:



Vêm-se as portas nas paredes que dividem as salas, que podemos designar pelo mesmo sistema usado no CASTELO DO TERROR, que é como segue:

Sala um "00030200" (nada a norte, quarto três a sul, quarto dois a leste, nada a oeste)
 Sala dois "05040001" (note-se que "fora" do mundo necessita de um número, tal como o cinco que aqui usámos)
 Sala três "01000000" (não existe aqui grande coisa)
 Sala quatro "02000000" (nem aqui)

Suponhamos que estava na sala dois. A instrução PRINT que descreve a sala talvez diga: "Encontra-se num pequeno quarto quadrado. Na parede norte há uma porta para o mundo exterior; há outra porta no sul e uma a oeste. Para onde deseja ir?" Se o jogador introduz "E", o computador olha para os "elementos de leste" do vector DATA que representa a sala dois (estes são o quinto e o sexto) e vê que só lá existem zeros. Sabe portanto que o jogador não se pode mover, e assinala: "Impossível atravessar paredes", e espera por nova direcção. Se o jogador desta vez introduz "S", significando que quer ir para sul, o computador olha para os "elementos de sul" do vector da sala dois (elementos três e quatro) e descobre "04". Isto significa que o jogador se está a mover da sala dois para a sala quatro. O

computador apenas tem de atribuir à variável que guarda o número da sala o VAL dos elementos indicados do vector da sala, e sabe assim onde se encontra o jogador. Note-se que nas instruções PRINT não há referência a "Sala um" ou "Sala dois". Estas são apenas para uso interno.

Sugerimos que se introduza agora o programa, fazendo-o correr (RUN) até se conseguir perceber como funciona. Isto oferecerá uma ideia do plano dos andares do castelo. O mapa será mais fácil de desenhar se se fizer uma série de quadrados separados na folha de papel, etiquetando-os com os nomes de cada sala à medida que se vão descobrindo e traçando pequenos "corredores" a interligá-los. Estes podem saltar por cima ou à volta de cada sala à medida que se vai percebendo as relações entre elas, e o mapa pode eventualmente ser desenhado a limpo uma vez que se tenha assegurado de que corresponde ao que se encontra escondido no programa. Uma vez desenhado, pode-se "vaguear" à vontade pelo castelo, comparando-o com o mapa original.

Após jogar algumas vezes com o programa regresse ao livro. Seguindo a listagem, discutiremos algumas maneiras de modificar e melhorar o programa dando-lhe algumas ideias gerais para o desenho de programas de aventuras.

```

10 REM      UINGANCA NO
            CASTELO DO TERROR
20 GO SUB 9000
30 CLS
35 IF J$="000000" THEN GO TO 9
40 PRINT INVERSE 1;N$; ", os se
us atributos são: "
50 IF VAL (J$(1 TO 2))>0 THEN
PRINT TAB 4; INVERSE 1;"MAGIA: "
;J$(1 TO 2)
60 IF VAL (J$(3 TO 4))>0 THEN
PRINT TAB 4; INVERSE 1;"FORÇA: "
;J$(3 TO 4)
70 IF VAL (J$(5 TO 6))>0 THEN
PRINT TAB 4; INVERSE 1;"SABEDORI
A: ";J$(5 TO 6)
80 IF DINHEIRO>0 THEN PRINT TA
B 4; INVERSE 1;"RIQUEZA: ";DINHE
IRO;"$"
```

```

90 IF J$="000000" THEN PRINT "
A aventura terminou.", "Esgotou t
odos os seus", "poderes. Lutou val
orosamente", "mas nao aguentou a
luta...". ADEUS....: STOP
100 GO SUB PAUSA
110 GO SUB SALA
120 LET M=0: IF Z<>1 THEN IF RN
D>.5 THEN GO SUB 7000: POKE 2369
2,-1
130 GO SUB PAUSA
140 GO SUB ACCAO
150 GO SUB PAUSA
170 GO TO 30
180 REM *****
2000 REM MONSTROS
2010 IF Q=1 THEN PRINT "Na sala
esta um feiticeiro furioso.
Tem uma taxa de magia de ";M$(
1,1 TO 2);", tem uma forza de ";M
$(1,3 TO 4);", e a sua sabe
doria e ";M$(1,5 TO 6)
2020 IF Q=2 THEN PRINT "Na sala
encontra-se um demonio flamejan
te. Tem uma sabedoria de ";M$(
2,5 TO 6);", uma forza de ";M$(2
,3 TO 4);", e um poder magico
de ";M$(2,1 TO 2)
2030 IF Q=3 THEN PRINT "Horror d
os horrores! Encontro
u, por azar, o antro de um te
rrivel gargula. Observa
de relance que a sua forza va
le ";M$(3,3 TO 4);", as suas habi
li-dades magicas ";M$(3,1 TO
2);", e a sua sa-bedoria ";M$(
3,5 TO 6)
2040 IF Q=4 THEN PRINT "Esbarrou
com qualquer coisa", "no escuro.
...": GO SUB PAUSA: PRINT "Para
seu azar, acaba de acordar um mi
notauro. A sua magia vale ";M$(
4,1 TO 2);", como forza tem ";M$(
4,3 TO 4);", e a sua sabedoria
vale ";M$(4,5 TO 6)
2050 IF Q=5 THEN PRINT "Agota me
teu-se em sarilhos...": PRINT "
Nesta sala esta o inimigo", "temi
do por todos aqueles", "que entra
m no castelo", "um lobisomem! Co
m uma forza de ";M$(5,3 TO 4);",
, sabedoria de ";M$(5,5 TO 6);",

```

```

e magia de ";M$(5,1 TO 2);", el
e e um inimigo temivel."
2060 GO SUB PAUSA
2070 RETURN
2080 REM *****
4000 REM FIM DO JOGO
4010 PRINT "Atolou-se na lama do
s pantanos circundantes da Lago
a Negra, sob o castelo. Para e
scapar do castelo tem de lutar
contra o Guardiao da Lagoa Ne
gra."
4020 PRINT "A luta deve envolver
todos os atributos...E vai pr
ecisar de 10, pelo menos, para
escapar."
4030 GO SUB PAUSA
4040 IF DINHEIRO<0 THEN PRINT "T
em ouro no valor de: ";DINHEIRO;
"$"
4050 GO SUB PAUSA
4060 PRINT "Os atributos do gua
rdiao."
4070 PRINT TAB 3;"MAGIA: ";M$(6,
1 TO 2)
4080 PRINT TAB 3;"FORCA: ";M$(6,
3 TO 4)
4090 PRINT TAB 3;"SABEDORIA: ";M
$(6,5 TO 6)
4100 PRINT "Os seus atributos:"
4110 PRINT TAB 3;"MAGIA: ";J$(1
TO 2)
4120 PRINT TAB 3;"FORCA: ";J$(3
TO 4)
4130 PRINT TAB 3;"SABEDORIA: ";J
$(5 TO 6)
4140 GO SUB PAUSA
4150 IF DINHEIRO<100 THEN GO TO
4270
4160 PRINT "Pode comprar pontos
de qualquer dos atributos, a 10
0$ cada."
4170 PRINT "Se quiser comprar, i
ntroduza a letra correspondente
ao atribu- to que deseja, segui
da pelo nu- mero de pontos."
4180 PRINT "Introduza 'N' se nao
quer com- prar nada."
4190 INPUT FLASH 1;"Atributo? ";
E$
4200 IF E$="N" THEN GO TO 4270
4210 INPUT FLASH 1;"Quantia a ga

```

```

star? ";AM
4220 IF DINHEIRO-AM<1-OR AM<100
THEN GO TO 4210
4230 IF E$="M" THEN LET J$(1 TO
2)=STR$(VAL (J$(1 TO 2))+INT (A
M/100))
4240 IF E$="F" THEN LET J$(3 TO
4)=STR$(VAL (J$(3 TO 4))+INT (A
M/100))
4250 IF E$="S" THEN LET J$(5 TO
6)=STR$(VAL (J$(5 TO 6))+INT (A
M/100))
4255 PRINT "MAGIA: ";J$(1 TO 2);
" FORÇA: ";J$(3 TO 4); "SABEDORIA
";J$(5 TO 6); " OURO ";DINHEIRO
; "$
4260 LET DINHEIRO=DINHEIRO-AM: I
F DINHEIRO>99 THEN GO TO 4190
4270 CLS: PRINT "E agora, para
a prova final...": GO SUB PAUSA
4275 INPUT FLASH 1; BRIGHT 1; "Ca
rregue em ENTER quando sentir co
ragem suficiente para lutar...";
A$: GO SUB PAUSA: CLS
4280 PRINT "Primeiro, a Magia:"
4290 LET MH=VAL (J$(1 TO 2)): LE
T MG=VAL (M$(6,1 TO 2))
4300 PRINT TAB 3; "Voce: ";MH;"
Guardiao: ";MG
4310 LET DIFF=ABS (MH-MG)
4320 PRINT "A diferenca vale ";D
IFF
4330 IF MH>MG THEN PRINT "a seu
favor"
4340 IF MG>MH THEN PRINT "a favo
r do Guardiao"
4350 GO SUB PAUSA: LET COST=INT
(RND*((MH+MG)/2))+1
4360 PRINT "O combate acarretou
uma penali- zacao de ";COST;" p
ontos de magia": GO SUB PAUSA
4370 LET RESULT=INT (RND*DIFF)+1
4380 LET RESHU=ABS (RESULT-MH)
4390 LET RESGA=ABS (RESULT-MG)
4400 IF RESHU>RESGA THEN PRINT "
E voce ganhou:" COST;" pontos de
magia": LET MH=MH+COST: LET J$(
1 TO 2)=STR$ MH: GO TO 4420
4410 PRINT "E voce perdeu:" COST
;" pontos de magia": LET MH=MH-C
OST: LET J$(1 TO 2)="00": IF MH>
0 THEN LET J$(1 TO 2)=STR$ MH

```

```

4480 INPUT FLASH 1; PAPER 7; INK
2; "Prima ENTER quando se atreve
r a continuar este desafio mortu
o": A$: GO SUB PAUSA: CLS
4485 PRINT "Segundo, a forca:"
4490 LET MH=VAL (J$(3 TO 4)): LE
T MG=VAL (M$(6,3 TO 4))
4500 PRINT TAB 3; "Voce: ";MH;"
Guardiao: ";MG
4510 LET DIFF=ABS (MH-MG)
4520 PRINT "A diferenca e ";DIFF
4530 IF MH>MG THEN PRINT "com va
ntagem sua"
4540 IF MG>MH THEN PRINT "com o
Guardiao a frente"
4550 GO SUB PAUSA: LET COST=INT
(RND*((MH+MG)/2))+1
4560 PRINT "O combate acarretou
uma penali- zacao de ";COST;" p
ontos de forca": GO SUB PAUSA
4570 LET RESULT=INT (RND*DIFF)+1
4580 LET RESHU=ABS (RESULT-MH)
4590 LET RESGA=ABS (RESULT-MG)
4600 IF RESHU>RESGA THEN PRINT "
E voce ganhou:" COST;" pontos de
forca": LET MH=MH+COST: LET J$(
3 TO 4)=STR$ MH: GO TO 4680
4610 PRINT "E voce perdeu:" COST
;" pontos de forca": LET J$(3 TO
4)="00": IF MH>0 THEN LET MH=MH
-COST: LET J$(3 TO 4)=STR$ MH
4680 INPUT FLASH 1; BRIGHT 1; PA
PER 2; INK 7; "Prepare-se agora p
ara o confronto final; quando s
e atrever pri-ma "; INVERSE 1; "E
NTER"; INVERSE 0; a$
4685 PRINT "Finalmente, a Sabedo
ria:"
4690 LET MH=VAL (J$(5 TO 6)): LE
T MG=VAL (M$(6,5 TO 6))
4700 PRINT TAB 3; "Voce: ";MH;"
Guardiao ";MG
4710 LET DIFF=ABS (MH-MG)
4720 PRINT "A diferenca e ";DIFF
4730 IF MH>MG THEN PRINT "e voce
esta em vantagem"
4740 IF MH>MG THEN PRINT "e o GU
ardiao esta a frente"
4750 GO SUB PAUSA: LET COST=INT
(RND*((MH+MG)/2))+1
4760 PRINT "O combate acarretou
uma penali- zacao de ";COST;" p

```



```

ontos de sabedoria": GO SUB PAUS
A
4770 LET RESULT=INT (RAND*DIFF)+1
4780 LET RESHU=ABS (RESULT-MH)
4790 LET RESGA=ABS (RESULT-MG)
4800 IF RESHU>RESGA THEN PRINT "
E voce ganhou:" COST;" pontos de
sabedoria": LET MH=MH+COST: LET
J$(5 TO 6)=STR$ MH: GO TO 4820
4810 PRINT "E voce perdeu:" COST
;" pontos de sabedoria": LET MH=
MH-COST: LET J$(5 TO 6)="00": IF
MH>0 THEN LET J$(5 TO 6)=STR$ M
H
4850 LET A=VAL (J$(1 TO 2))
4860 LET B=VAL (J$(3 TO 4))
4870 LET C=VAL (J$(5 TO 6))
4880 GO SUB PAUSA
4890 PRINT "No fim da batalha c
rucial voce tem: "
4900 PRINT TAB 3;"MAGIA: ";A
4910 PRINT TAB 3;"FORCA: ";B
4920 PRINT TAB 3;"SABEDORIA: ";C
4930 IF A+B+C>9 THEN PRINT "FLA
SH 1; BRIGHT 1; PAPER 7; INK 2;"
Conseguiu,oh heroi destes negros
e perigosos tempos.
Eu vos armo cavaleiro "; INVERSE
1;"Dom"; INVERSE 0;" ";N$: GO S
UB PAUSA: STOP
4940 PRINT "Acabou-se tudo.Preci
sava, pelo menos, de um total d
e 10 pontos para escapar ao Guar
diao. Lutou valentemente,
mas sera devorado por ele....
4960 GO SUB PAUSA: STOP
4970 REM *****
5000 REM SUBROTINA ACCAO
5005 LET D=4: IF B$(Z,9)="0" THE
N LET D=1
5010 PRINT "Que quer fazer agor
a?"
5015 INPUT Z$: IF Z$="0" THEN ST
OP
5020 IF Z$="" THEN CLS : LET Z$=
"#"
5022 IF D=4 AND Z$<>"F" AND Z$<>
"L" THEN LET D=0: GO TO 5160
5025 IF Z$="F" OR Z$="L" THEN GO
TO 5130
5030 IF Z$="N" AND B$(Z,1 TO 2)=

```

```

"00" THEN PRINT "SEM SAIDA": GO
TO 5010
5040 IF Z$="S" AND B$(Z,3 TO 4)=
"00" THEN PRINT "NAO HA PORTA PO
R AI": GO TO 5010
5050 IF Z$="E" AND B$(Z,5 TO 6)=
"00" THEN PRINT "NAO E POSSIVEL"
: GO TO 5010
5060 IF Z$="O" AND B$(Z,7 TO 8)=
"00" THEN PRINT "SE PASSAR ATRAV
ES DA PAREDE,...": GO TO 5010
5070 IF Z$="N" THEN LET Z=VAL (B
$(Z,1 TO 2)): RETURN
5080 IF Z$="S" THEN LET Z=VAL (B
$(Z,3 TO 4)): RETURN
5090 IF Z$="E" THEN LET Z=VAL (B
$(Z,5 TO 6)): RETURN
5100 IF Z$="O" THEN LET Z=VAL (B
$(Z,7 TO 8)): RETURN
5110 RETURN
5130 IF B$(Z,9)="0" THEN PRINT "
Nao ha ninguem com quem lutar":
GO TO 5010
5140 IF Z$="F" THEN LET D=INT (R
AND*2)
5150 IF D=1 THEN PRINT "Qual a d
ireccao? ": GO TO 5015
5160 IF D=0 THEN PRINT "Nao!! Te
m de ficar e lutar"
5170 PRINT "Quais os atributos
que deseja usar na luta? "
5180 LET Q=VAL (B$(Z,9))
5190 INPUT Z$: IF Z$<>"M" AND Z$
<>"F" AND Z$<>"S" THEN GO TO 519
0
5200 IF Z$="M" THEN LET HUM=VAL
(J$(1 TO 2)): LET MON=VAL (M$(Q,
1 TO 2))
5210 IF Z$="F" THEN LET HUM=VAL
(J$(3 TO 4)): LET MON=VAL (M$(Q,
3 TO 4))
5220 IF Z$="S" THEN LET HUM=VAL
(J$(5 TO 6)): LET MON=VAL (M$(Q,
5 TO 6))
5230 LET DIFF=ABS (HUM-MON)
5240 PRINT "FLASH 1; BRIGHT 1;
INK 2; PAPER 7;"A diferenca e "
;DIFF
5250 IF HUM>MON THEN PRINT FLASH
1; BRIGHT 1; PAPER 2; INK 7;" e
voce tem vantagem"
5260 IF HUM<MON THEN PRINT FLASH

```

```

1; BRIGHT 1; PAPER 2; INK 7; "e
o seu adversario tem vantagem"
5270 LET COST=INT (RND*6)+1
5280 PRINT "FLASH 1; BRIGHT 1;
PAPER 3; INK 9; "Este combate ac
arreta uma pena- lidade de"; IN
VERSE 1; COST; INVERSE 0; " pontos
5290 LET RESULT=INT (RND*DIFF)+1
5300 GO SUB PAUSA
5310 LET RESHU=ABS (RESULT-HUM)
5320 LET RESMO=ABS (RESULT-MON)
5330 IF RESHU<RESMO THEN GO TO 5
390
5340 PRINT "FLASH 1; PAPER 2;
INK 6; "Foi derrotado!!"
5350 IF Z$="M" THEN LET M$(0,1 T
O 2)=STR$ (VAL (M$(0,1 TO 2))+2*
COST); IF VAL (J$(1 TO 2))>=COST
THEN LET J$(1 TO 2)=STR$ (VAL (
J$(1 TO 2))-COST)
5360 IF Z$="F" AND VAL (J$(3 TO
4))>=COST THEN LET J$(3 TO 4)=ST
R$ (VAL (J$(3 TO 4))-COST)
5370 IF Z$="S" THEN LET M$(0,5 T
O 6)=STR$ (VAL (M$(0,5 TO 6))+IN
T (COST/2)); IF VAL (J$(5 TO 6))
>=COST THEN LET J$(5 TO 6)=STR$
(VAL (J$(5 TO 6))-COST)
5380 GO TO 5440
5390 REM VITORIA HUMANA
5400 PRINT "FLASH 1; BRIGHT 1;
INK 6; PAPER 2; "Derrotou o inim
igo!!"
5410 IF Z$="M" THEN LET J$(1 TO
2)=STR$ (VAL (J$(1 TO 2))+2*COST
); IF VAL (M$(0,1 TO 2))>=COST T
HEN LET M$(0,1 TO 2)=STR$ (VAL (
M$(0,1 TO 2))-COST)
5420 IF Z$="F" THEN LET M$(0,3 T
O 4)=STR$ (VAL (M$(0,3 TO 4))-CO
ST)
5430 IF Z$="S" THEN LET J$(5 TO
6)=STR$ (VAL (J$(5 TO 6))+INT (C
OST/2)); IF VAL (M$(0,5 TO 6))>=
COST THEN LET M$(0,5 TO 6)=STR$
(VAL (M$(0,5 TO 6))-COST)
5440 GO SUB PAUSA
5450 PRINT "Apos a luta, os seus
atributos sao:
5460 PRINT TAB 3; "MAGIA: "; J$(1
TO 2)

```

```

5470 PRINT TAB 3; "FORCA: "; J$(3
TO 4)
5480 PRINT TAB 3; "SABEDORIA: "; J
$(5 TO 6)
5490 PRINT "E os do ";
5500 IF Q=1 THEN PRINT "Feiticeir
o"
5510 IF Q=2 THEN PRINT "Demonio"
5520 IF Q=3 THEN PRINT "Gargula"
5530 IF Q=4 THEN PRINT "Minotaur
o"
5540 IF Q=5 THEN PRINT "Lobisoho
mem"
5550 PRINT "sao: "
5560 PRINT TAB 3; "MAGIA: "; M$(Q,
1 TO 2)
5570 PRINT TAB 3; "FORCA: "; M$(Q,
3 TO 4)
5580 PRINT TAB 3; "SABEDORIA: "; M
$(Q,5 TO 6)
5590 GO SUB PAUSA
5600 PRINT "Prima ENTER para con
tinuar"
5610 INPUT T$: GO SUB PAUSA: CLS
5620 LET K=2+INT (RND*8)
5630 IF K=Z THEN GO TO 5620
5640 LET B$(K,9)=B$(Z,9)
5650 LET B$(Z,9)="0"
5660 IF RND>.5 THEN RETURN
7000 REM CONTEUDOS
7005 IF B$(Z,9)<>"0" THEN LET M=
1: RETURN : REM NAO PERMITE CON
TEUDOS SE HA UM MONSTRO A COMBA
TER
7010 PRINT : GO SUB 7000+100*INT
(RND*5+1)
7020 GO SUB PAUSA
7030 RETURN
7100 LET ARCAS=ARCAS+1: IF ARCAS
=5 THEN RETURN
7110 PRINT "Na sua frente esta u
ma arca          marcada com o numero
"; ARCAS
7120 PRINT "Quer abri-la (S/N)?
"
7130 GO SUB 7600
7150 IF Z$="N" THEN RETURN
7160 LET J=INT (RND*3): GO SUB P
AUSA
7170 IF J=0 THEN LET OURO=100+IN
T (RND*300): PRINT "Contem ouro

```

```

de um dragao que vale ";C
ASH;"$!": LET DINHEIRO=DINHEIRO
+OURO
7180 IF J=1 THEN PRINT "Um gnomo
saltou da arca e ata- cou-o co
m um punhal!"; LET LOSS=INT (RND
*6)+1; IF VAL (J$(3 TO 4))-LOSS>
=0 THEN LET J$(3 TO 4)=STR$ (VAL
(J$(3 TO 4))-LOSS); RETURN
7190 IF J=2 THEN PRINT "Um estra
nho fumo sai da arca, adormece
ndo-o e retirando-lhe, lentamen
te o seu poder magico"; LET LOSS
=INT (RND*6)+1; IF VAL (J$(1 TO
2))-LOSS>=0 THEN LET J$(1 TO 2)=
STR$ (VAL (J$(1 TO 2))-LOSS); RE
TURN
7200 IF POCOES=1 THEN GO TO 7010
7210 LET POCOES=1
7220 PRINT "Encontrou uma pequen
a garrafa tendo gravadas umas
curiosas letras retorcidas"
7240 PRINT "Vai beber a pocao co
ntida na garrafa (S/N)?"
7250 GO SUB 7600
7260 IF Z$="N" THEN RETURN
7270 GO SUB PAUSA
7280 IF RND>.5 THEN PRINT "Conti
nha uma pocao para aumentara sab
edoria"; LET J$(5 TO 6)=STR$ (VA
L (J$(5 TO 6))+INT (RND*6+1)); R
ETURN
7290 PRINT "Continha uma pocao q
ue o enfra- queceu, fazendo-o do
rmir"; GO SUB PAUSA; LET LOSS=IN
T (RND*6)+1; IF VAL (J$(3 TO 4))
<LOSS THEN RETURN
7295 LET J$(3 TO 4)=STR$ (VAL (J
$(3 TO 4))-LOSS); RETURN
7300 IF PERGAMINHOS=1 THEN RETUR
N
7310 LET PERGAMINHOS=1
7320 PRINT "Encontrou um rolo de
pergaminho.Deseja le-lo (S/N)?"
7325 GO SUB 7600; IF Z$="N" THEN
RETURN
7330 IF RND>.5 THEN PRINT "Nao c
onsegue entender a lingua- gem.";
GO SUB PAUSA; RETURN
7335 PRINT "Tem escrito um feiti
co.Quer le- lo (S/N)?"
7340 GO SUB 7600; GO SUB PAUSA;

```

```

IF Z$="N" THEN RETURN
7350 IF RND>.5 THEN PRINT FLASH
1; BRIGHT 1; INK 7; PAPER 2;"Era
um feitico benefico"; GO SUB PA
USA; LET J$(1 TO 2)=STR$ (VAL (J
$(1 TO 2))+INT (RND*6+1)); RETUR
N
7360 PRINT PAPER 2; INK 7; FLASH
1; BRIGHT 1;" Era um feitico ma
leficoll!"; GO SUB PAUSA; IF VAL
(J$(3 TO 4))>5 THEN LET J$(3 TO
4)=STR$ (VAL (J$(3 TO 4))-INT (
RND*6+1)); RETURN
7400 IF COFRES=1 THEN GO TO 7010
7405 LET COFRES=1
7410 PRINT "Na parede encontra-s
e um cofre- zinho dourado e, em
frente, uma chave"
7415 PRINT "Quer abrir o cofre
(S/N)?"
7420 GO SUB 7600
7425 IF Z$="N" THEN RETURN
7430 IF RND>.3 THEN GO TO 7460
7435 GO SUB PAUSA; PRINT "Uma h
arpia guinchante voa de dentr
o do cofre e crava os dente
s na sua garganta!!"
7440 GO SUB PAUSA
7445 PRINT "Luta com ela e...";
GO SUB PAUSA; PRINT "...finalmen
te conseguiu torcer- -lhe o pesc
oco"
7450 IF VAL (J$(3 TO 4))>5 THEN
LET J$(3 TO 4)=STR$ (VAL (J$(3 T
O 4))-INT (RND*6+1))
7455 GO SUB PAUSA; GO TO 7620
7460 PRINT "Ouve-se um coro de v
ozes ange- licas."; GO SUB PAUS
A; PRINT "Sente-se recomposto e
as suas feridas sararam"
7480 LET J$(3 TO 4)=STR$ (VAL (J
$(3 TO 4))+INT (RND*6+1))
7490 RETURN
7500 GO TO 7100
7600 LET Z$=INKEY$
7610 IF Z$<>"N" AND Z$<>"S" THEN
GO TO 7600
7615 PRINT
7620 BEEP .5,1; BEEP .5,2; BEEP
.5,-1; RETURN
7630 REM *****
8000 REM INTERIOR DAS SALAS

```



```

8010 IF Z=1 THEN PRINT "Encontra
-se na entrada de um
de aspecto antigo e
so. Esta na ala norte
lo e, olhando para sul
rtura degradada repa-
rtal de entrada, aber-
guarda."
8020 IF Z=2 THEN PRINT "Esta na
sala de entrada, or-
a com ricos tapetes.
pontam para leste e
m portal aberto apon-
oeste."
8030 IF Z=3 THEN PRINT "Esta sal
e nao passa de uma
cao. Tem apenas uma
oeste, por onde aca-
ntrar."
8040 IF Z=4 THEN PRINT "Encontra
-se na Camara Real.
uena sala tem como
l caracteristica uma
ornamental da deusa
num pedestal situado
nordeste. Tem portas
do a sul, leste e oes-
8050 IF Z=5 THEN PRINT "A Ala da
s Intrigas, uma
rada com paineis
ra, sugerindo su-
boatos, com sai-
ste e a sul, donde
a enxofre e um can-
terioso."
8060 IF Z=6 THEN PRINT "Entrou n
o covil do Feiticeir
sala com um caldeirao
nte sobre um fogo da
erdes, no canto sudo-
e quarto cheira forte-
enxofre a arder, com
proprio das antigas
ode sair, pelo sul ou
te."
8070 IF Z=7 THEN PRINT "Encontra
-se num local apa-
te pacifico e sosse-
a Galeria dos Qua-
castelo, com uma pin-
lendario Guardiao da
gra no lado esquerdo

```

```

castelo
misterio
do caste
pela est
ra no po
to e sem
Esta na
namentad
Portas a
sul, e u
ta para
Esta sal
arrecada
saida a
bou de e
Encontra
Esta peq
principa
estatuas
da Lua,
no canto
conduzin
te."
"A Ala da
sala for
de madei
surros e
das a le
vem odor
tico mis
Entrou n
ro, uma
borbulha
chamas v
este. Est
mente a
um aroma
magias. P
pelo les

```

```

a da parede leste.
dela veem-se as jane-
eadas da Grande Ala
ncontram do outro la-
m interior. As saidas
ia conduzem a norte e
8080 IF Z=8 THEN PRINT "Esta e,
porventura, a mais
a sala do castelo,
rande, com um tecto
os e pesados barro-
sair pelas portas
o lado norte ou do
te, onde se ouve mu-
as janelas da parede
-se o Jardim interior
la dele, as janelas
a com muitas gravuras
as."
8090 IF Z=9 THEN PRINT "Sons dum
quarteto de cordas
ta sala, a Camara
cos. Pode sair pelas
o oeste ou do sul."
8100 IF Z=10 THEN PRINT "Eis o S
antuario do Silencio,
a de calma desconcer-
hambiente e humido e
avendo saidas a norte
B$(10,3 TO 4)="12" THEN PRINT "
e uma porta conduz a sul"
8110 IF Z=11 THEN PRINT "Este pa
rece ser o Vestibulo
piros, uma sala humi-
cura onde, segundo a
o Guardiao da Lagoa
ode, por vezes, ser
a noite. Tem uma porta
a norte
IF B$(11,3 TO 4)="12" THEN P
RINT " e outra a sul"
8120 PRINT : IF Z=12 THEN GO TO
4000
8125 IF Z<>1 THEN IF B$(Z,9)="0"
AND AND>.55 THEN LET B$(Z,9)=ST
R$ (INT (AND*5)): REM APAGUE PAR
A TER MENOS MONSTROS
8130 IF B$(Z,9)<>"0" THEN LET Q=
VAL (B$(Z,9)): GO SUB 2000
8140 RETURN
8150 REM *****
9000 REM INICIALIZACAO
9010 RANDOMIZE

```

```

Atraves
las grad
que se e
do Jardi
da Galer
oeste."
Esta e,
magnific
a Sala G
de solid
tes. Pode
duplas d
lado es
sica. Pel
oeste ve
e, para
duma sal
pendurad

```

```

9020 LET Z=1
9030 LET PAUSA=9900
9040 LET SALA=8000
9050 LET ACCAO=5000
9060 LET DINHEIRO=0
9070 LET ARCOAS=0
9080 LET POCOES=0
9090 LET PERGAMINHOS=0
9100 LET COFRES=0
9110 DIM B$(12,9)
9120 DIM M$(6,6): DIM J$(6)
9130 POKE 23692,-1: POKE 23658,8
9140 INPUT "Qual o seu nome prop
rio? ";N$
9150 PAPER 1: INK 7: BORDER 1: C
L$
9160 PRINT "Viva, ";N$: PRINT "A
guarde, por favor..."
9320 REM PREENCHIMENTO
DOS QUARTOS
9330 FOR T=1 TO 12
9340 READ T$
9350 LET B$(T)=T$
9360 NEXT T
9365 LET B$((10+INT (RND*2)),3 T
0 4)="12"
9370 REM CRIACAO E DISTRIBUICAO
DOS MONSTROS
9380 FOR A=1 TO 5
9390 LET B$(INT (RND*9+1),9)=STR
$ A
9400 NEXT A
9450 LET B$(1,9)="0"
9460 REM ATRIBUTOS
9470 FOR A=1 TO 6
9480 LET M$(A,1 TO 2)=STR$ (INT
(RND*11)+10)
9490 LET M$(A,3 TO 4)=STR$ (INT
(RND*11)+10)
9500 LET M$(A,5 TO 6)=STR$ (INT
(RND*11)+10)
9510 NEXT A
9520 REM ATRIBUTOS DO JOGADOR
9530 LET J$(1 TO 2)=STR$ (INT (R
ND*11)+10)
9540 LET J$(3 TO 4)=STR$ (INT (R
ND*11)+10)
9550 LET J$(5 TO 6)=STR$ (INT (R
ND*11)+10)
9560 RETURN
9570 DATA "000200000"
9580 DATA "000803040"

```

```

9590 DATA "000000020"
9600 DATA "000702050"
9610 DATA "000604000"
9620 DATA "051007000"
9630 DATA "040000060"
9640 DATA "020009000"
9650 DATA "001100080"
9660 DATA "060000000"
9670 DATA "090000000"
9680 DATA "000000000"
9890 REM SUBROTINA PAUSA
9900 POKE 23692,-1: IF RND>.5 TH
EN GO TO 9950
9905 FOR I=RND*10 TO RND*30+10 S
TEP .5
9910 BEEP .03,I
9920 NEXT I
9930 PRINT
9940 RETURN
9950 FOR I=1 TO 50
9960 IF RND>.7 THEN BORDER=RND*7
9970 NEXT I
9980 BORDER 1
9990 GO TO 9930

```

Quem sobreviveu (ou morreu) no CASTELO DO TERROR, Versão 1, poderá guardá-lo em *cassete* e fazer as modificações que vamos mostrar para produzir a Versão 2, que usa um mapa mais complicado. Modificando-se as linhas seguintes fica-se pronto a recomençar toda a tarefa. Infelizmente para os conservadores, o Jardim (que era mais um pátio interior do primeiro castelo), teve de ser deixado de fora. Pelo menos, ainda pode ser visto das janelas, mas já não se avistam outras salas através dele, por isso não ajuda tão facilmente a encontrar as "coordenadas" do jogador.

Em primeiro lugar modificam-se as instruções DATA como se indica a seguir:

```

9570 DATA "000002000"
9580 DATA "040700000"
9590 DATA "000000070"
9600 DATA "000205000"
9610 DATA "000600040"
9620 DATA "050010090"
9630 DATA "020003000"
9640 DATA "091100000"

```

```

9650 DATA "000806080"
9660 DATA "001200050"
9670 DATA "080012000"
9680 DATA "000000000"

```

Agora é necessário modificar partes das descrições das salas. Para facilitar as alterações acrescentamos as palavras novas em maiúsculas.

Note-se que não são necessárias alterações na linha 8030.

```

8000 REM INTERIOR DAS SALAS
8010 IF Z=1 THEN PRINT "Encontra
-se na entrada de um castelo
de aspecto antigo e misterio
so. Esta na ala OESTE do caste
lo e, olhando para NORTE pela est
rutura degradada repa- ra no po
rtal de entrada, aber- to e sem
guarda."
8020 IF Z=2 THEN PRINT "Esta na
sala de entrada, or- namentad
a com ricos tapetes. Portas a
pontam para NORTE e sul.": R
EM APAGUE O RESTANTE DA LINHA OR
IGINAL
8030 IF Z=3 THEN PRINT "Esta sal
a nao passa de uma arrecada
cao. Tem apenas uma saida a
oeste, por onde aca- bou de e
ntrar."
8040 IF Z=4 THEN PRINT "Encontra
-se na Camara Real. Esta peq
uena sala tem como principa
l caracteristica uma estatua
ornamental da deusa da Lua,
num pedestal situado no canto
nordeste. Tem portas conduzin
do a sul e ESTE E UMA JANELA D
ONDE SE VE O JARDIM INTERIOR
."
8050 IF Z=5 THEN PRINT "A Ala da
s Intrigas, uma sala for
rada com paineis de madei
ra, sugerindo su- surros e
boatos, com sai- das a le
ste e a sul, donde vem odor
a enxofre e um can- tico mis
terioso. A JANELA NA PAREDE N
ORTE DA PARA O JARDIM INTERIOR
."

```

```

8060 IF Z=6 THEN PRINT "Entra n
o covil do Feiticeiro, uma
sala com um caldeirao borbulha
nte sobre um fogo de chamas v
erdes, no canto sudo- este. Est
e quarto cheira forte- mente a
enxofre a arder, com um aroma
proprio das antigas magias. P
ode sair, pelo sul ou pelo les
te."
8070 IF Z=7 THEN PRINT "Encontra
-se num local apa- rentemen
te pacifico e sosse- gado. Eis
a Galeria dos Qua- dros do
castelo, com uma pin- tura do
lendario Guardiao da Lagoa Ne
gra no lado esquerdo da Janel
a da parede NORTE. Atraves
dela veem-se as Jane- las grad
eadas da Grande Ala que se e
ncontram do outro la- do Jardi
m interior. As saidas da Galer
ia conduzem a norte e ESTE."
8080 IF Z=8 THEN PRINT "
Esta e, porventura, a mais
magnifica sala do castelo,
a Sala Grande, com um tecto
de solidos e pesados barro-
tes. Pode sair pelas portas
duplas do lado SUL ou do
lado NORTE, onde se ouve mu-
sica.": REM APAGUE A REFERENCIA
AO JARDIM
8090 IF Z=9 THEN PRINT "Sons dum
quarteto de cordas soam nes
ta sala, a Camara dos Musi
cos. Pode sair pelas portas d
e ESTE ou do sul."
8095 REM NOTE QUE OS 'IF/THEN'
FORAM REMOVIDOS DAS
DUAS LINHAS SEGUINTE
8100 IF Z=10 THEN PRINT "Eis o s
antuário do Silencio, uma sal
a de calma desconcer- tante. O
ambiente e humido e frio, h
avendo saidas a OESTE e uma p
orta que conduz a sul."
8110 IF Z=11 THEN PRINT "Este pa
rece ser o Vestibulo dos Sus
piros, uma sala humi- da e es
cura onde, segundo a lenda, p
o Guardiao da Lagoa Negra p

```


ode, por vezes, ser ouvido
a noite. Tem uma porta a norte
e outra a ESTE."

Não há razão que impeça de continuar a trabalhar neste programa, desenvolvendo tantas versões do CASTELO DO TERROR quantas queiramos. Podemos cingir-nos às mesmas salas e ao mesmo número de salas (12), fazendo o nosso próprio mapa e escolhendo as instruções DATA e PRINT relevantes, desprezando o mapa primitivo e tentando esquecer os seus pormenores à medida que entrarmos no castelo por nós imaginado. É vantajoso "trocar castelos" com os amigos e tentar reconstituir o que eles codificaram nas suas variantes do programa. Embora seja preferível produzir um mundo bem estruturado, passível de ser desenhado em mapa (porque senão o melhor é começar já a carregar em ENTER e a gerar instruções PRINT), nada impede que cada qual faça as suas próprias interligações entre as salas, em duas dimensões, o mais "cruelmente" que se possa. Também é relativamente fácil adicionar mais monstros, quer sejam para povoar o castelo original, quer para se dispersarem mais por um castelo do novo projecto.

Quando o programador começar a confiar nestas técnicas, pode duplicar o número de salas, aumentando o vector literal B\$ para dar conta deste facto e, obviamente, aumentando as instruções DATA introduzidas nos elementos de B\$, e as instruções PRINT que se ligam com cada sala; mas não deverá, no entanto, empenhar-se nesta tarefa antes de ter conhecimento completo do nosso original e da maneira como está codificado, e sem ter idealizado algumas variantes do CASTELO DO TERROR com 12 salas. Os monstros são guardados no vector literal M\$.

Talvez as mudanças mais fáceis de fazer ao programa sejam os conteúdos das arcas, o efeito de beber a poção mágica e o resultado da leitura, ou da tentativa de leitura, do rolo de papiro. De momento o programa não apresentará o papiro, o pequeno cofre dourado ou a poção mais do que uma vez num único jogo, e apenas existem quatro arcas em qualquer esquema do castelo.

No entanto, enquanto o programa não produzir mais do que quatro arcas, não há mecanismo na sua presente forma que evite que as mesmas coisas saltem de diferentes arcas depois de abertas. Mas quem tenha sido beliscado na garganta por quatro Harpias, cada uma reduzindo o atributo de ENERGIA, deve desejar que haja maneira de evitar esta proliferação pelas arcas do castelo.

Muitos possuidores de *Spectrum* já devem ter sentido pena de ter um computador com tantas capacidades gráficas e deparar apenas com jogos na forma escrita, ainda que abrilhantado por alguns FLASH, INVERSE e BRIGHT. Há jogadores que preferem programas que desenhem gravuras completas em cada cena. Mas as listagens já são longas e acreditamos que, se fossem ainda mais compridas, poucas pessoas teriam a coragem e a paciência de as introduzir no computador. Programas que criam figuras minuciosas tendem a tornar-se num labirinto de instruções PLOT e DRAW, necessitando de grande número de linhas para produzir uma figura interessante ou que agrade a todos. Mas não há que ter susceptibilidades quanto a acrescentar novas salas, monstros ou conteúdos de arcas ou melhorar o programa adicionando imagens.

Teríamos muito gosto em ver, nestes programas, melhoramentos gráficos ou quaisquer outras variantes que o leitor tenha desenvolvido. Se nos enviar o produto das suas elocubrações, através do editor, talvez venhamos a incluí-las nas próximas edições deste livro.

Mesmo que não queira ir ao ponto de incluir vastos blocos de instruções com o desenho de salas completas, talvez imagine cores específicas de descrições particulares de monstros para serem impressas nas mesmas combinações de cores, talvez com um "distintivo sonoro". Repare-se nas linhas de 2010 a 2050: é fácil inserir nestas linhas a informação respeitante à cor e à música do distintivo sonoro, seguido "IF Q = THEN".

Outra alternativa para preencher gráficos minuciosos de cada sala consiste em usar os gráficos definidos pelo utilizador para formar um bloco de quatro caracteres para cada um dos cinco monstros, produzindo uma moldura característica quando se

menção a um determinado monstro. Poderia, de facto, encher todo o *écran* com múltiplos gráficos definidos pelo utilizador antes de o limpar para o aparecimento do texto e/ou do distintivo sonoro.

Outros cenários

Mesmo que não tenha descoberto totalmente o mapa usado no CASTELO DO TERROR mas querendo mesmo assim personalizar este programa de aventuras, há alterações significativas a introduzir. Supondo que as entradas e saídas permanecem nos mesmos locais, não há motivo para não chamar às salas aquilo que se quiser. É fácil, por exemplo, colocar a "aventura" a bordo de uma nave espacial abandonada, a apodrecer numa lenta órbita situada muito para além da cintura de asteróides e habitada por criaturas de uma raça desconhecida.

Em vez de (ou além de) "monstros", podem-se imaginar perigosos artefactos abandonados por essas criaturas, como *robots*-sentinelas, cuja missão seria patrulhar a nave, embora as criaturas da raça que eles protegem tenham já desaparecido. E o objectivo das suas pesquisas pode ser, por exemplo, a fonte energética da nave, que se calcula ser exemplo de uma tecnologia muito avançada.

Uma vez encontrada a sala (aquela que corresponde à Lagoa Negra do castelo), encontra-se um sistema de defesa controlado pelo *Spectrum* que tem de ser derrotado.

Em vez de MAGIA, ENERGIA e SABEDORIA, podemos escolher atributos mais realistas, incluindo coisas como oxigénio, rações alimentares e balas de sobra para a arma laser do explorador. Pode-se mudar não apenas a natureza dos atributos com que se leva a cabo a aventura, mas adicionar outros, como carisma, esperteza e genica. Pode-se atribuir à partida, ao jogador, um certo número de pontos distribuídos pelos atributos conforme se prefira, em vez de os mesmos serem determinados pelo lançamento de dados (ou pelo gerador de número aleatório, que é exactamente a mesma coisa).

Outra maneira de escolher os atributos à partida é atribuir ao jogador o tipo de pessoa que deseja ser (sacerdote, mágico, ogro,

lutador, ladrão, duende e outros). Há, claro, grupos específicos de atributos para cada personagem.

A moldura original do programa pode transformar-se (por exemplo) na reconstituição de uma pirâmide a explorar antes de ser destruída para dar lugar a um novo dique no Nilo, atravessando túneis, explorando câmaras no interior da pirâmide, até chegar à câmara mortuária (correspondente à Lagoa Negra do Castelo do Terror), onde se terá de defrontar a maldição do Faraó.

A partir destas ideias, pode-se ver como é fácil elaborar um cenário para a aventura, melhorando-o de tal modo que nenhuma semelhança tenha com o programa original. Adquirida alguma confiança na manipulação de realidades codificadas deste modo, é provável que se fique em pulgas para escrever programas pessoais. Observando as aventuras de origem comercial criadas para o *Spectrum* ou para outros computadores (aventuras tipo *Hobbit* ou *Pharaoh's Tomb*), encontram-se muito boas ideias. Algumas delas, talvez surpreendentemente, não se baseiam em "realidades coerentes" e dependem mais de números aleatórios do que da perícia para traçar mapas ou memorizar caminhos por parte do jogador (não é este o caso dos dois jogos comerciais atrás mencionados).

É aliciante experimentar fazer uma fusão de diferentes programas, adicionando um labirinto a três dimensões, ou um ou dois testes de reacção para ver como o jogador se comporta em combate.

Finalmente, tenha-se em mente que quando se escreve um programa de aventuras é essencial assegurar um claro objectivo para o jogo (vaguear sem um alvo, coleccionando tesouros e matando dragões, não tem interesse e é por vezes bastante aborrecido, a não ser que o jogador esteja em vias de resolver algum problema final, como obter um tesouro ou escapar com vida de uma câmara de tortura).

Além de terem um objectivo, as actividades dentro da aventura devem ser relacionadas com esse mesmo objectivo.

QUIOSQUE DE LIMONADAS

O jogador é o Jaime. Tem por missão dirigir um quiosque de limonadas e fazer fortuna (ou, pelo menos, não ir à falência). O programa condu-lo, passo a passo, às decisões que terá de tomar, e faz um uso bastante eficiente das possibilidades de cor e dos gráficos de alta resolução do *Spectrum*, que avivam o programa à medida que progride. Até aparece um retrato do Jaime, atrás do balcão do seu quiosque. O jogador que não se deixe enganar pelas figuras que aqui vê; os gráficos dos programas são bastante mais eficazes do que as gravuras sugerem.

Uma vez o programa a funcionar, informam-no de que é o dono de um quiosque que vende limonadas. Cada dia recebe a previsão meteorológica. Tempo quente significa muitas vendas; um dia cinzento sugere que o negócio não vai ser famoso. Da previsão meteorológica e do conhecimento do mercado que adquirirá ao longo do jogo, terá o Jaime de decidir quantos copos de limonada deve fazer. Também se lhe dá a informação quanto ao custo de fabrico de cada copo de limonada. Tem pois de decidir qual o preço de venda. É evidente que preços altos tendem a reduzir as vendas. No princípio do jogo o aluguer do quiosque será de 125\$00 por dia. Com o tempo este encargo vai-se tornando mais dispendioso devido à inflação. Este fenómeno também provocará a redução dos lucros, porque a matéria-prima aumenta de preço.

As gravuras exemplificam o tipo de informação com que o Jaime terá de trabalhar cada dia: temperatura moderada, custo de 5\$00 por cada copo de limonada, renda diária de 125\$00 e capital de 5000\$00. A partir destes elementos decide quantos copos de limonada fará, e este número aparece. O programa agora desenha o quiosque completo com o número de copos cheios de limonada e com um cartaz mostrando o preço unitário.

Os copos de limonada são vendidos mesmo à nossa frente (ou seja, são esvaziados). Ao fim do dia, fecham-se as persianas do quiosque e ficará então a saber como o jogador se saiu no programa. "Vendeu 36 copos; recebeu 720\$; fez 40 copos que lhe custaram 200\$; a renda custou 125\$; lucrou 395\$" é o tipo do relatório que se recebe.

As instruções DATA nas linhas 15 e 20 representam, respectivamente, os copos vazios (E) e cheios (F). A instrução DATA a seguir, linha 25, contém o relatório meteorológico e os números que governam o efeito que certos estados de tempo terão nas vendas. Coloca-se a preto a cor dos caracteres, põe-se o *écran* com brilho normal e depois limpa-se (linha 45), isto tudo antes de as linhas 50 e 60 definirem os gráficos, mediante a leitura do carácter da linha 50, e fazerem o POKE dos dados relevantes da linha 60.

As linhas 100 a 160 imprimem as instruções e, após uma pausa (veja-se PAUSE 9000, na linha 160), aparecem as palavras "Boa sorte!". A linha 170 produz, com FLASH, a familiar frase "Carregue em qualquer tecla para iniciar". PAUSE 0 mantém a mesma imagem no *écran* indefinidamente até ser pressionada uma tecla (o equivalente no ZX81 seria PAUSE 4E4 ou PAUSE 40000) e o *écran* ser limpo.

As variáveis são inicializadas na linha 180 usando determinados nomes indicativos das suas funções. Ao seu dinheiro à partida (variável CAPITAL) é atribuído o valor 50000; ao tempo dá-se o valor 4 (variável TEMPO); à matéria-prima é atribuído o valor 5 (variável LIMONADA); e finalmente à renda atribui-se o valor de 125 (variável RENDA). As linhas 200 a 220 escolhem o tempo e a linha 230 inicia o relatório. A linha 250 faz o RESTORE do ponteiro de dados em relação à linha 25, que guarda os tipos do tempo e os números relacionados: os números (a que foram dados os nomes "de" e "até") são usados na linha 840, que verifica o seu sucesso num determinado dia.

Na linha 290 é-se informado do estado das finanças e pergunta-se qual o número de copos de limonada que se deseja fazer. Caso se sugira um número negativo, a linha 302 rejeita a resposta e devolve a acção para a linha 300, para uma nova resposta. A linha 305 faz o *Spectrum* correr rapidamente para a sub-rotina da linha 9000, que verifica se o Jaime tem dinheiro suficiente para comprar tantos copos quantos encomendou. Caso tenha, a sub-rotina faz o RETURN instantaneamente. Alcança-se a linha 9010 ao tentar comprar mais copos do que se pode, fazendo ver que não tem dinheiro que chegue para

tantos copos", e pede um novo número. Isto é verificado na linha 9000, e se o número introduzido é aceitável, o RETURN do fim da linha 9000 é activado. Caso contrário, o programa "cai" novamente para a linha 9010, que faz a pergunta de novo. Estando neste extremo do programa, repare-se na linha 9999. Esta, de facto, não faz parte do programa enquanto tal, mas foi usada quando o programa estava a ser desenvolvido. É uma linha utilizada sempre que a pessoa que está a desenvolver o programa decide olhar para o que já fez.

A moldura, o fundo e a tinta voltam às cores normais, o *écran* é limpo e o programa listado. O comando RETURN no fim significa que esta instrução pode ser chamada do interior do programa como se de uma sub-rotina se tratasse. Verificar-se-á que, ao desenvolver um programa complexo, poderá ser muito útil a inclusão de uma linha como esta.

A linha 307, depois de verificar a sub-rotina do dinheiro, determina (com bastante sensatez) que caso o número de copos encomendado seja menor do que um, o número vendido tem de ser zero, e o programa salta para a linha 950, pulando sobre a parte de vendas da listagem.

Se provar pelo número de copos encomendados que leva este negócio a sério, então o programa imprime (linha 310) o número de copos que encomendou e depois imprime os copos cheios neste mesmo número (com o ciclo da linha 320).

A linha 330 pede-lhe que introduza o seu "programa de vendas" para esse dia com a linha 333, rejeitando preços superiores a 249 ou inferiores a 1. Após um curto *beep* (linha 335), é impresso o preço de venda e pedido para (linha 350) "Carregar em qualquer tecla para iniciar as vendas". Soa um *beep*, e o programa espera (usando PAUSE 0) até outra tecla ser premida. Pelo soar de outro *beep* (no fim da linha 350) damos conta do facto de a tecla ser premida e a linha 360 limpa o *écran*, coloca o brilho (BRIGHT) a funcionar e limpa o conteúdo do *écran* através do ciclo I.

Vem agora a secção visualmente mais interessante. Esta longa e aparentemente complicada rotina, das linhas 370 a 690, cria a primeira parte da figura do Jaime e do seu quiosque, com

prateleiras para os copos de limonada. A secção seguinte, usando a variável CONTADOR (a que é atribuído o valor de COPOS na linha 695) imprime os copos de limonada cheios. A variável CONTADOR decresce uma unidade na linha 715 e é verificada até ser igual a zero, sabendo o computador nessa altura que todos os copos foram impressos. Feito isto e depois de a linha 700 ter determinado que todos os copos foram colocados nas prateleiras para atrair os clientes sequiosos, as linhas de 730 a 830 completam o desenho do quiosque das limonadas.

A linha 840 define o número de transacções do dia em função do tempo e do preço unitário, deixando uma certa margem ao factor sorte. Os ciclos da linha 860 substituem os copos cheios das prateleiras por vazios. Se todos os copos forem vendidos, aparece o cartaz "ESGOTADO" (linha 900). Após uma pausa por alguns segundos no início da linha 910, as persianas do quiosque fecham-se. Outra pausa no início da linha 940 e o computador solicitamente imprime o relatório de vendas do dia. "Vendeu... copos. Recebeu...\$" e assim por diante.

O ambicionado lucro (variável LUCRO) é calculado na linha 1000.

As linhas 1010 e 1030 usam a função SGN para determinar se o dia foi um sucesso ou um desastre em termos de lucro (o resultado da função SGN aplicado a um número negativo é -1; aplicado a um número positivo é +1 e a zero é 0).

A inflação ataca de forma alarmante na linha 1050, aumentando a renda em parcelas de 25\$, aproximadamente dez vezes em cem. Na linha 1060 o gerador de números aleatórios assegura mais uma vez que em 10% das vezes o custo da matéria-prima (variável LIMONADA) será incrementado duas ou três unidades. A linha 1065 espera que uma tecla seja premida para continuar. Se a linha 1067 descobre que o Jaime já não tem dinheiro (ou seja "capital" inferior a um) o programa vai para 1500 e termina delicadamente. Se ainda está em condições de pagar as suas dívidas, terá outra oportunidade no instável mercado da alta finança das beberagens, com um regresso à linha 200 para enfrentar mais um dia de decisões importantes.


A mensagem "FECHADO. Faliu e o quiosque foi vendido

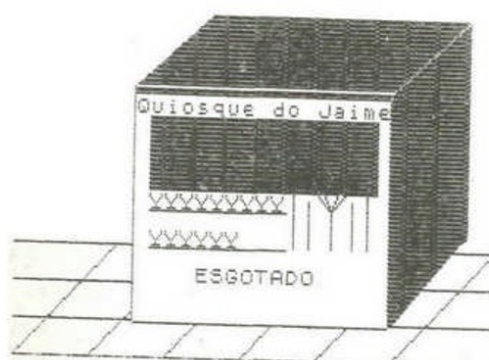
O quiosque de limonadas

Possui um quiosque de bebidas que vende limonadas. Todos os dias tem ao seu dispor o boletim meteorológico e a partir dele decide quantos copos quer fazer. Também sabe o custo de cada copo de limonada e com essa base decide a quanto vai vender cada um. A renda do quiosque custa 120\$ por dia, mas torna-se mais dispendioso com a inflação, como acontece com a limonada,



Quiosque de limonadas
Relatorio do dia
Tempo : Nebuloso

Cada copo sai-lhe a 5\$
Renda: 125\$
Dispo'e de 5000\$
Copo's: 



Quiosque de limonadas
Relatorio de vendas

Vendeu 35 copos
pelos quais recebeu 720\$
Fez 40 copos
que lhe custaram 200\$
Pagou de renda 125\$
Teve um lucro de 395\$

Graficos do programa
Y ... Graphics "f"
Y ... Graphics "g"


```

12 POKE 23609,128
15 DATA 0,BIN 10000010,BIN 100
00010,BIN 01000100,BIN 00101000,
BIN 00010000,BIN 00010000,BIN 01
111100
20 DATA 0,BIN 10000010,BIN 111
11110,BIN 01111100,BIN 00111000,
BIN 00010000,BIN 00010000,BIN 01
111100
25 DATA "Tempestuoso",10,30,"C
huvoso",25,40,"Humido",35,50,"Ne
buloso",45,70,"Ameno",65,120,"Qu
ente",100,180,"Vaga de calor",20
0,300
45 INK 0: BRIGHT 0: CLS
50 FOR i=0 TO 7
55 READ a: POKE USR "e"+i,a: N
EXT i
60 FOR i=0 TO 7
65 READ a: POKE USR "f"+i,a: N
EXT i
70 BEEP .5,0
100 PRINT PAPER 6;"      O quiosq
ue de limonadas
" Possui um quiosque de bebidas
"
110 PRINT "que vende limonadas.
Todos os "dias tem ao seu dis
por o boletim"
120 PRINT "meteorologico e a pa
rtir dele""decide quantos copos
quer fazer,"
130 PRINT "Tambem sabe o custo
de cada copo""de limonada e com
essa base de-"
140 PRINT "cide a quanto vai ve
nder cada ""Um. A renda do quio
sque custa"
150 PRINT "125$ por dia,mas tor
na-se mais""dispendiosa com a i
nflacao, co-"
160 PRINT "mo acontece com a li
monada."" : PAUSE 900: PRINT INK
2;"      Boa sorte !"
170 PRINT FLASH 1;"Prima quique
r tecla para iniciar": PAUSE 0:
CLS
180 LET capital=5000: LET tempo
=4: LET limonada=5: LET renda=12
5
200 LET tempo=INT (RND*7)+1
230 PRINT PAPER 6;"      Quiosqu

```

```

e de limonadas
240 PRINT INK 7; PAPER 1;"
Relatorio do dia
250 RESTORE 25: FOR i=1 TO temp
o: READ w$,de,ate: NEXT i
260 PRINT "Tempo : "; INK 1;w$
270 PRINT "Cada copo sai-lhe a
"; INK 1;limonada;"$
280 PRINT "Renda: "; INK 1;ren
da;"$
290 PRINT "Dispoe de "; INK 1;
capital;"$
300 BEEP .5,10: INPUT INK 2;"Qu
antos copos quer fazer"; FLASH 1
;"?",copos
302 IF copos<0 THEN GO TO 300.
305 GO SUB 9000: BEEP .5,5
307 IF copos<1 THEN LET vendido
s=0: LET preco=1: LET copos=0: G
O TO 950
310 INK 6: PAPER 0: PRINT "Cop
os: "
320 FOR i=1 TO copos: PRINT "Y"
;: NEXT i: PAPER 7: PRINT "
330 BEEP .5,10: INPUT INK 2;"A
quanto vende hoje o copo "; FLAS
H 1;"?";preco
333 IF preco<1 OR preco>249 THE
N GO TO 330
335 BEEP .1,5
340 INK 0: PAPER 7: PRINT "Lim
onada a ""preco;""$ o copo"
350 PRINT FLASH 1: INK 2;"Prim
a qualquer tecla para inici-ar a
s vendas": BEEP .1,10: PAUSE 0:
BEEP .5,5
360 CLS: BRIGHT 1: FOR i=16 TO
21: PRINT AT i,0;" : NEXT i
370 FOR i=0 TO 48 STEP 16
380 PLOT 0,i: DRAW 255,0
390 NEXT i
400 FOR i=0 TO 200 STEP 40
410 PLOT i,0: DRAW 55,48
420 NEXT i
430 PRINT PAPER 6;AT 8,8;"Quios
que do Jaime"; PAPER 6;AT 9,8;"
440 FOR i=92 TO 98
450 PLOT INK 7;72,i: DRAW INK 7
;120,0,-PI/12
460 NEXT i

```



```

470 FOR i=1 TO 6: PRINT PAPER 6
;AT 9+i,8;" ";TAB 24; PAPER 6;"
480 NEXT i
490 FOR i=1 TO 4: PRINT PAPER 6
;AT 15+i,8;"
NEXT i
495 BRIGHT 0
496 INK 6
500 FOR i=112 TO 115
510 PLOT 64,i: DRAW 135,0
520 NEXT i
530 FOR i=64 TO 199
540 PLOT i,116: DRAW 40,31
550 NEXT i
560 FOR i=115 TO 24 STEP -1
570 PLOT BRIGHT 0;200,i: DRAW B
RIGHT 0;40,31
580 NEXT i
590 FOR i=16 TO 23
600 PLOT BRIGHT 0,200,i: DRAW B
RIGHT 0,40,40
610 NEXT i
615 INVERSE 1
620 PLOT PAPER 0;64,116: DRAW P
APER 0;134,0
630 PLOT PAPER 0;64,112: DRAW P
APER 0;134,0
640 INK 0: INVERSE 0
650 PLOT 63,16: DRAW 0,100
660 PLOT 199,16: DRAW 0,96: DRA
W INVERSE 1; INK 6; PAPER 0,0,4
670 PLOT 64,15: DRAW 134,0
680 INK 2: PLOT 72,79: DRAW 72,
0
690 PLOT 72,63: DRAW 72,0
693 PLOT 72,47: DRAW 72,0
695 LET contador=cofos: GO SUB
700: GO TO 730
700 IF contador=0 THEN RETURN
705 FOR i=11 TO 15 STEP 2: FOR
J=1 TO 9
710 BRIGHT 8: PRINT AT i,8+J;"Y
": BEEP .1,20
715 LET contador=contador-1: IF
contador=0 THEN RETURN
720 NEXT J: NEXT i: RETURN
730 PLOT 160,60: DRAW 15,0,-PI*
3/2
740 PLOT 148,48: DRAW 0,27: DRA
W 12,5,-PI/6
750 PLOT 187,48: DRAW 0,27: DRA

```

```

W -12,5,PI/6
760 PLOT 156,48: DRAW 0,20
770 PLOT 179,48: DRAW 0,20
780 PLOT 168,48: DRAW 0,16: DRA
W 8,8: DRAW 0,8: DRAW -8,-16
790 DRAW -8,8: DRAW 0,8: DRAW 8
,-16
800 PLOT 162,83: DRAW 10,0,PI/2
810 CIRCLE 164,92,2: CIRCLE 171
,92,2
830 PRINT AT 17,9: INK 0; PAPER
6;"LIMONADA ";preco;"$
835 IF preco<=limonada THEN LET
frac=1: GO TO 840
837 LET p=preco: LET l=limonada
: LET frac=EXP (-.5*((p-l)/(2+l
)+2)
840 LET vendidos=(INT (RND*(ate
-de)+de))*frac
845 LET vendidos=INT vendidos:
IF vendidos=0 THEN GO TO 910
847 IF vendidos>cofos THEN LET
vendidos=cofos
850 LET contven=vendidos
860 FOR i=11 TO 15 STEP 2: FOR
J=9 TO 17
870 PRINT AT i,J;"Y": BEEP .1,1
0
880 LET contven=contven-1: IF c
ontven=0 THEN GO TO 900
890 NEXT J: NEXT i: GO SUB 700:
GO TO 860
900 IF vendidos=cofos THEN PRIN
T INK 0; PAPER 4;AT 17,9;" ESG
OTADO "
910 PAUSE 100: FOR i=103 TO 47
STEP -1
920 BEEP .02,i-48: PLOT 72,i: D
RAW 120,0
930 NEXT i
940 PAUSE 200: BEEP .5,5
950 CLS : PAPER 7: INK 0: PRINT
PAPER 6;" "; BRIGHT 1;"Quio
sque de limonadas"; BRIGHT 0;"
960 PRINT INK 7; PAPER 1;" "
Relatorio de vendas "
970 PRINT "Vendeu "; INK 1; ven
didos;" cofos"
980 PRINT "Pelos quais recebeu
"; INK 1; vendidos*preco;"$
990 PRINT "Fez "; cofos;" cofos

```

```

"que lhe custaram "; INK 1; co
pos*limonada;"$";
995 PRINT "Pagou de renda "; re
nda;"$";
1000 LET lucro=vendidos*preco-co
pos*limonada-renda
1010 IF SGN lucro=-1 THEN PRINT
"Teve um "; INK 2; "prejuizo"; I
NK 0; "de "; -lucro;"$";
1020 IF SGN lucro=0 THEN PRINT
"Nao perdeu nem ganhou dinheiro";
1030 IF SGN lucro=1 THEN PRINT
"Teve um "; INK 4; "lucro "; INK
0; "de "; lucro;"$";
1040 LET capital=capital+lucro
1050 IF RND>.9 THEN LET renda=re
nda+INT (RND*10+1)
1060 IF RND>.9 THEN LET limonada
=limonada+2+INT (RND*2)
1064 PRINT FLASH 1; INK 2; ""P
rima uma tecla para continuar";
1065 BEEP .5,10: PAUSE 0: BEEP .
5,5
1067 IF capital<1 THEN GO TO 150
0
1070 CLS : GO TO 200
1500 CLS : PRINT INK 1; FLASH 1;
FECHADO
"
1510 PRINT " Faliu e o seu qui
osque foi vendido para pagar
as dividas."
1520 GO TO 1520
9999 STOP
9000 IF copos*limonada<=capital
THEN RETURN
9010 INPUT INK 1; "Nao tem dinhei
ro que chegue para tantos."; INK
2; "Quantos copos quer fazer ";
FLASH 1; "?"; copos
9020 GO TO 9000
9999 INVERSE 0: PAPER 7: BRIGHT
0: INK 0: BORDER 7: CLS : LIST :
RETURN

```

APERFEIÇOAR A MENTE

SINTAXE

Este é um jogo "intelectual" para pessoas que gostam de jogar com palavras. O propósito do jogo consiste em elaborar dez frases por forma a fazerem sentido. Ao correr o programa, o computador pede que esperemos uns momentos, e durante este tempo dispõe aleatoriamente os seus verbos, substantivos e adjectivos por onze frases que certamente não fazem sentido.

A nossa tarefa consiste em colocar tais frases em português correcto.

Vê-se um cursor no lado direito do *écran* com o número da frase que ele está a apanhar. O cursor pode-se mover para cima e para baixo usando as teclas "6" e "7". Carregando em qualquer das teclas de "1" a "5" salientam-se palavras de uma determinada frase a que o cursor esteja apontando, ou seja, carregando em "3" a terceira palavra da frase será destacada.

Há um lote de onze frases que se encontram vazias no início do jogo.

Para introduzir palavras nestas frases é preciso destacar uma palavra (como foi descrito no parágrafo anterior) e premir ENTER. O computador imprimirá então todas as suas frases com a palavra acrescentada e colocará automaticamente a palavra no lugar certo da sua frase.

Ao premir ENTER, o computador perguntará em que frase desejamos colocar a palavra. Se quiser a palavra introduzida na frase número 5, naturalmente que introduzirá o número 5.

Eventualmente haverá um lote de onze frases para alterar várias vezes até se mostrarem coerentes. O computador não diz se encontrámos a combinação certa de palavras, por isso cabe-nos julgar se as frases estão ou não correctas.

A linha 20 selecciona uma posição dentro da lista de números a serem dados pelo gerador de números aleatórios e a linha 30 manda o computador para a linha 1000, onde as matrizes literais são DIMensionadas e se fazem as atribuições das variáveis. As linhas 1040 e 1050 procuram elementos vazios da matriz literal

A\$, regressando a 1040 para seleccionar um novo elemento se o primeiro escolhido não é vazio. A linha 1060 lê o próximo vector literal das instruções DATA da linha 9000. A um elemento da matriz 1 é atribuído o comprimento da palavra que se acabou de ler, e então esta palavra (B\$) é atribuída a um elemento do vector A\$ (linha 1070).

Após X\$ ter sido definido a "A\$" (note-se que não foi A\$ mas sim "A\$"; a diferença será clara quando virmos a linha 1100), a linha 1110 dirige a atenção para a sub-rotina que começa na linha 200. O *écran* é limpo (linha 200) e os ciclos cruzados A e B imprimem aleatoriamente frases desordenadas.

A instrução RETURN na linha 260 manda o *Spectrum* de volta à linha 1160, onde as variáveis U, P e N são inicializadas. Estas serão usadas num dado momento para mover as palavras pelo *écran*. Após a sub-rotina da linha 100 ter impresso o curso (o quadrado a cheio do fim da linha 150), a rotina de 1195 aceita, e actua, nos nossos desejos relativos a movimento das palavras. A linha 1205 atribui a i\$ o valor da tecla carregada ("6" ou "7") para mover, para cima ou para baixo, o cursor relativo às frases. A linha 1260 aceita a escolha dentro da frase indicada e, depois de ter sido atribuído a X\$ o valor de "S\$" (mais uma vez "X\$" e não "S\$"), a linha 1300 dirige o programa de volta à sub-rotina da linha 200. As linhas 1340 e 1350 esperam que tiremos o dedo do teclado, antes de reenviar a acção para 1100.

```

5 REM Sintaxe
10 RESTORE
20 RANDOMIZE
25 LET flag=0
30 GO TO 1000
100 LET e=0
110 LET v=u/2+1
120 FOR b=2*(n/2)+1 TO n-1
130 LET e=e+l(b,v)+1
140 NEXT b
150 PRINT AT u+(n/2),e+3*(n/3);
OVER 1; INVERSE 1; " "
160 RETURN
200 CLS

```

```

205 FOR a=1 TO 11
207 PRINT "Um ";
210 FOR b=1 TO 5
215 LET r=flag*(j(b,a)-l(b,a))+
l(b,a)
220 PRINT VAL$ (X$+"(b,a, TO r)
");
230 IF b=2 THEN PRINT
240 NEXT b
245 PRINT
250 NEXT a
255 LET flag=0
260 RETURN
1000 PRINT AT 11,0;"ESPERE UNS M
OMENTOS..."
1005 DIM a$(5,11,14)
1010 DIM a$(5,11,14)
1015 DIM l$(5,11); DIM j(5,11)
1020 FOR a=1 TO 5
1030 FOR b=1 TO 11
1040 LET c=INT (RND*11)+1
1050 IF a$(a,c,1)<>" " THEN GO T
O 1040
1060 READ b$
1065 LET l(a,c)=LEN b$
1070 LET a$(a,c)=b$
1080 NEXT b
1090 NEXT a
1100 LET x$="a$"
1110 GO SUB 200
1150 LET uu=0
1160 LET u=0
1170 LET p=29
1180 LET n=1
1185 PRINT AT uu,30;" "
1190 PRINT FLASH 1;AT u,p;"<";ST
R$ INT (u/2+1)
1192 GO SUB 100
1195 IF INKEY$<>" " THEN GO TO 11
95
1200 IF INKEY$="" THEN GO TO 120
0
1205 LET i$=INKEY$
1210 PRINT AT u,p;" "
1215 GO SUB 100
1220 BEEP .01,20
1230 LET uu=u; LET u=u+((i$="6")
*(u/20)-(i$="7")*(u/0))+2
1240 IF CODE i$>48 AND CODE i$<5
4 THEN LET n=VAL i$
1250 IF CODE i$<>13 THEN GO TO 1
185

```

```

1260 INPUT "para que frase?" q
1270 IF q<1 OR q>11 THEN GO TO 1
190
1280 LET s$(n,q)=a$(n,u/2+1)
1285 LET j(n,q)=l(n,u/2+1)
1287 LET flag=1
1290 LET x$="s$"
1300 GO SUB 200
1340 IF INKEY$<>"" THEN GO TO 13
40
1350 IF INKEY$="" THEN GO TO 135
0
1360 GO TO 1100
9000 DATA "reluzente","veloz","f
eio","bom","conhecido","velho","
circunspecto","assiduo","pequeni
no","solicito","corpulento"
9005 REM
9010 DATA "jogador","homem do ta
lho","guarda-chuva","banqueiro",
"criado","ministro","robot","ara
nhico","espectador","cavalo","pe
ixe"
9015 REM
9020 DATA "passeia","vira-se","v
oa","observa","tece","fica","inc
lina-se","corta","nada","e detid
o","garante"
9025 REM
9030 DATA "a","pela","sobre a","
sem","com a","na","a","por","uma
","de","a"
9035 REM
9040 DATA "fraude","pradaria","m
esa","vitoria","teia","corrida",
"bicicleta","lagoa","corrente",
"vitela","ventania"

```

Um pequentino criado
inclina-se na vitela
Um velho banqueiro
fica pela corrida
Um bom ministro
corta uma pradaria
Um reluzente peixe
passeia com a bicicleta
Um conhecido jogador
voa por corrente
Um circunspecto espectador
garante a teia
Um solícito aranhico

observa de vitoria
Um veloz robot
nada sobre a ventania
Um feio cavalo
e detido sem mesa
Um corpulento homem do talho
vira-se a lagoa
Um assiduo guarda-chuva
tece a fraude

O POÇO

Apresenta-se neste programa um cenário muito realista e muito verosímil. O jogador vai ser o infeliz prisioneiro de um velho demente e maldoso que decidiu que os seus pecados deveriam ser expiados por afogamento num poço, se não conseguisse demonstrar suficiente capacidade matemática.

Eis como o programa o explica:

Neste jogo esta prisionei-
ro de um professor louco
que o desce lentamente para
a agua. Ele faz-lhe pergun-
tas. Quando errar, o balde
desce mas se acertar o bal-
de sobe.

Eis o que se pode ver enquanto o interrogatório decorre:



O programa é bastante simples, e permite um grande grau de elaboração.

A rotina da linha 1000 desenha a cena. Antes de ela aparecer, o computador pergunta ao jogador como vai na matemática. Introduzindo um número menor do que 12, o computador escolherá um nível de dificuldade ao acaso. Sugerimos que se comece com 15 da primeira vez que o programa correr. A linha 710 atribui a T\$ o sinal da multiplicação (*) ou da adição (+), e na linha 720 dá à variável T um número entre um e o número escolhido para nível de dificuldade. Se T\$ é o sinal de multiplicação, à variável "B" na linha 735 é atribuído um número entre um e dez. Se T\$ é o sinal de adição, B é escolhido entre 0 e cem vezes o número introduzido como nível de dificuldade.

A linha 740 imprime a pergunta no *écran*, substituindo o "*" por um pequeno "x" no caso da multiplicação (pois "x" é o sinal geralmente usado nesta operação). O ciclo G de 777 a 840 dá a oportunidade de introduzir a resposta. Esta rotina aceita os caracteres introduzidos por intermédio de INKEY\$ e converte-os num número de comprimento maior do que um carácter; estudá-la é um bom método para aprender a escrever rotinas semelhantes em futuros programas.

Nas linhas 850 a 985 o professor reage à sua resposta e gera-se uma nova pergunta.

Examinando a rotina de impressão da cena vê-se que está perto o fim do programa; L é a variável que determina a que profundidade do poço deve ser desenhado o balde. Se este chega ao cimo, o professor é puxado pela corda e somos conduzidos a um fim muito agradável (e de que a forma imprevista negará qualquer lei que normalmente governe as relações entre cordas, poços e baldes).

Se o balde chegar à água, estará reservada ao jogador a sensação de um final agradável.

```

5 REM O POÇO
10 LET L=40
20 GO SUB 9000
510 PRINT AT 5,6;"O POÇO"
520 PRINT , ,TAB 5;"Neste jogo

```

esta Prisionei- ro de um pro
fessor louco que o desce
lentamente para a água. Ele
faz-lhe pergun- tas. Quando
errar, o balde desce mas se
acertar o bal- de sobe."

```

630 GO SUB 1005
640 PRINT AT 21,6;"Prima tecla
para iniciar."
650 PAUSE 0
660 GO SUB 1000
670 PRINT AT 5,6;"O professor d
iz:"
680 INPUT "Que tal vai na mate
mática? (%)" d
682 IF d<12 THEN LET d=INT (RND
*7)+4
685 LET l=121-d
690 IF d<0 OR d>100 THEN PRINT
AT 5,6;"Introduza um numero de 1
a 100": GO TO 680
700 GO SUB 1000
710 LET t$=CHR$ (42+INT (RND*2)
)
720 LET a=INT (RND*d)+1
730 IF t$="*" THEN LET b=INT (R
ND*10)+1: GO TO 740
735 LET b=INT (RND*d*100)
740 PRINT AT 5,6;"Quantos faz
";
750 LET u$=t$
760 IF t$="*" THEN LET u$="x"
765 LET c$=STR$ VAL (STR$ a+t$+
STR$ b)
770 PRINT a;" ";u$;" ";b;" ?"
772 LET tt=0
775 LET w$=""
777 FOR g=1 TO LEN c$
780 FOR t=1 TO l*10
785 LET i$=INKEY$
790 IF i$="" THEN NEXT t
800 IF CODE i$<48 OR CODE i$>57
THEN GO TO 910
810 LET w$=w$+i$
815 LET tt=tt+1
820 IF w$(g)<>c$(g) THEN GO TO
910
825 PRINT AT 5,6;w$: BEEP .1,20
830 IF INKEY$<>"" THEN GO TO 83
0
840 NEXT g

```

```

850 PRINT AT 10,6;"Bolas!! Acer
t ou!"
860 FOR a=1 TO tt/10
870 BEEP .01,20
880 NEXT a
890 LET l=(l+10-tt/LEN c$)/10
0
900 GO TO 700
910 PRINT AT 8,6;"#
915 FOR a=1 TO 20
920 BEEP .1,0
930 NEXT a
940 PRINT AT 12,6;"He! He! A re
s post a certa"
950 PRINT AT 13,6;"seria ";c$
960 FOR u=1 TO 3
970 PAUSE 30
980 PRINT AT 3,l/8+3;"ha!"
990 PAUSE 30
1000 PRINT AT 3,l/8+3;" "
1010 NEXT u
1020 LET l=l+4
1030 GO TO 700
1040 STOP
1050 CLS
1060 PLOT 0,150
1070 DRAW 0,-150
1080 DRAW 40,0
1090 DRAW 0,150
1100 PLOT 40,135: DRAW 215,0
1110 PLOT 1,40: DRAW 39,0
1120 CIRCLE 20,170,5
1130 PLOT 20,175
1140 DRAW l,-40,-(l<85)
1150 PRINT AT 4,l/8+2; OVER 1;"*
"
1070 PLOT 15,170
1080 DRAW 0,-l
1090 DRAW -4,-8
1100 DRAW 8,0
1110 DRAW -4,8
1120 PLOT 11,152-l
1130 DRAW 2,-8
1140 DRAW 4,0
1150 DRAW 2,8
1160 IF l=0 THEN BEEP 1,22: GO T
O 2000
1165 IF l=153 THEN BEEP 1,22: GO
TO 3000
1170 IF l<20 THEN LET l=0: BEEP
1,20: GO TO 1000
1175 IF l>122 THEN LET l=153: BE

```

```

EP 1,20: GO TO 1000
1180 RETURN
2000 PRINT AT 4,2; OVER 1;"*"
2010 FOR a=6 TO 16
2020 PRINT AT a-1,2;" "
2030 PRINT AT a,2;"*"
2035 BEEP .05,a
2040 NEXT a
2050 FOR a=17 TO 21
2055 PRINT AT a,2;"*"
2060 PRINT AT a-1,2;" "
2070 BEEP .2,21-a
2080 NEXT a
2090 PRINT AT 21,2; OVER 1;"_"
2100 PRINT AT 16,2;"_"
2110 STOP
3000 STOP
9000 DATA 24,24,254,58,24,36,10:
0
9010 FOR a=0 TO 7
9020 READ b
9030 POKE USR "a"+a,b
9040 NEXT a
9050 RETURN
UDG-      *=A

```

Sugestões para programação

COMO APERFEIÇOAR OS PROGRAMAS

'Os artistas foram sempre dos primeiros a explorar novas tecnologias...' (David Thornburg, "Computadores e Sociedade", revista *Compute!*, Março de 1982, pág. 16)

O advento do computador pessoal deu a muitas pessoas um ensejo para descobrir a sua criatividade. Para muitos, o simples facto de terem fácil acesso a um computador deu-lhes também acesso a uma actividade criadora, deixando vir ao de cima faculdades que de outro modo continuariam latentes.

Como em todas as formas de arte, o conhecimento da técnica facilita a expressão do desejo de programação criadora. Ter muitas técnicas e truques de reserva permite inventar e desenvolver programas sem sermos dominados pelo medo de que o computador não actue como queremos. Por exemplo, saber como se faz mover objectos permite que nos concentremos noutras características que o nosso programa deverá apresentar, em vez de ficarmos atolados nos mecanismos da criação de movimento.

Examinar os programas deste livro e de outros, assim como os publicados em revistas da especialidade, ensina bastante sobre programação.

Decerto que à medida que se introduzem os programas e lêem as notas que escrevemos a acompanhá-los, se tirará deles um certo número de ideias que facilmente podemos incluir nos novos programas. Quando ainda não se tem prática do jogo da programação deve-se começar por alterar apenas as listagens deste livro, mudando elementos como os gráficos definidos pelo utilizador, ou as teclas que as rotinas INKEY\$ lêem, prosseguindo mais tarde com a escrita de programas completamente originais.

Ao desenvolver o programa, deve ter em mente o seu lado "humano". Em muitos casos, os programas virão a ser utilizados por outras pessoas e, por esta razão, é conveniente procurar

tornar a sua execução o mais fácil e agradável possível.

Por exemplo, muitos programas (incluindo alguns deste livro) escritos para o *Spectrum* usam as teclas "5", "6", "7" e "8" para controlar o movimento, deslocando um objecto sob o controle do jogador nas direcções indicadas pelas setas dessas teclas. Mas, pensando um pouco sobre isso, conclui-se que não são as teclas mais convenientes para usar em todas as circunstâncias. Estão muito juntas no topo do teclado e não são, certamente, as de mais fácil acesso. São muito mais convenientes, se um programa apenas exige movimentos para a esquerda e para a direita, as teclas "Z" (para a esquerda) e "M" (para a direita). Outras teclas podem ser utilizadas para os movimentos para cima e para baixo. Observa-se este facto no uso de programas deste livro, como em CORRIDA DA MORTE.

Vale a pena pôr em causa todas as acções de um programa que se fazem por hábito (tal como usar as teclas "5" e "8") para verificar se esse hábito ajuda ou não a produzir um programa cujo uso seja o mais agradável possível.

Escolhendo, para o movimento, outras teclas que não "5" e "8", refiram-se aquelas que pareçam "óbvias", por forma a que o jogador se divirta sem dificuldades, em vez de gastar energia mental a tentar lembrar-se das funções de cada tecla. Devem escolher-se teclas cómodas de usar e com posições de algum modo relacionadas com a sua função no programa (tal como no canto inferior esquerdo do teclado — "Z" — para mover para a esquerda). Para outros controles num programa, como disparar uma mortal arma láser anti-invasores, fará sentido escolher uma tecla de fácil acesso, ou uma que esteja relacionada (como "F" para Fogo!, ou "L" para lançar os dados) com a função que vai activar.

Observa-se isto na VINGANÇA NO CASTELO DO TERROR, onde se introduz "E" para mover-se para este, "N" para norte ou "S" para sul. Lidas as instruções para um jogo como este, é pouco provável esquecer a função de cada tecla. Se possível, devem incluir-se nos programas comentários bem explícitos para que o jogador saiba exactamente o que fazer em cada situação.

Embora não seja necessário fornecer instruções completas com o programa (e na maior parte das vezes elas ocupam demasiado tempo de memória e de escrita), os comentários de INPUT devem ser claros. O uso de instruções-guia para o utilizador, colocadas entre aspas numa intrução INPUT do *Spectrum*, torna simples a tarefa de dar ao operador comentários explícitos ou linhas mestras. Como é sabido, podem-se usar todas as intruções padrão de controle do PRINT, tais como FLASH, BRIGHT e INVERSE, dentro dos comentários de INPUT, realçando esses comentários e aumentando o interesse geral criado pelo programa.

Uma vez o programa a funcionar, deve-se fazer um esforço por tornar o aspecto visual o mais atractivo possível. Mesmo que as saídas para o *écran* sejam apenas compostas de instruções PRINT, não há razão para que não sejam ordenadas no *écran* de maneira interessante, clara e atraente. O uso da cor nas instruções PRINT ou ainda mudanças na moldura ou enquadramento (BORDER) de tempos a tempos, melhoram a mensagem e aperfeiçoam a sua legibilidade.

Por exemplo, no programa de xadrez que se encontra neste livro usamos mudanças aleatórias no enquadramento para dar conhecimento da captura de uma peça do computador por outra do jogador.

O som também melhora os programas. É surpreendente o que alguns *beeps* podem proporcionar a um programa, não apenas para coisas óbvias, como um invasor do espaço a desintegrar-se, ou bolas a bater nas paredes, mas também para aplicações como as do XADREZ PARA PRINCIPIANTES. Numa determinada parte desse programa, o computador faz soar *beeps* à medida que procura uma peça para jogar. Isto assegura ao jogador, mesmo que o computador leve tempo a tomar uma decisão, que o programa não se perdeu num ciclo sem fim.

Qualquer uso da cor e do som aumenta o impacto de um programa.

Enquanto uns são contra o uso da cor pela cor, não parece sensato que ao usar um computador como o *Spectrum* (que tem na cor uma das suas características principais) não se tire

vantagem destas características sempre que se possa. O mesmo argumento aplica-se aos gráficos definidos pelo utilizador. Deverão ser usados sempre que se sinta que isso vai melhorar o programa. No nosso programa de xadrez não criámos os nossos próprios gráficos porque tínhamos por objectivo pôr em realce a qualidade do jogo e não a apresentação. No entanto, não há desculpa (excepto, talvez, a preguiça) para usar asteriscos a representar invasores espaciais em vez de um ou dois gráficos definidos pelo utilizador.

Sem esquecer a valorização que se obtém ao adicionar cores aos caracteres ou ao fundo, ou realçar certas palavras com INVERSE, FLASH e BRIGHT, há coisas mais subtis para melhorar a mensagem, tais como incluir certas palavras em maiúsculas quando o resto da mensagem está em minúsculas. Vê-se um exemplo disto nas descrições da segunda versão da VINGANÇA NO CASTELO DO TERROR, onde incluímos as palavras alteradas em maiúsculas, para se saber quais as modificações a introduzir naquela variante do jogo. Embora as palavras fossem colocadas em maiúsculas para facilitar a tarefa de conversão e não para as fazer realçar no programa, elas ilustram como é eficaz uma simples alteração como esta. A posição da saída PRINT para o *écran* também pode ser importante, e a instrução PRINT AT pode facilitar a colocação das mensagens onde desejarmos.

AUMENTO DA VELOCIDADE DO PROGRAMA

Em alguns programas, a qualidade do jogo é mais importante do que a velocidade (tais como PIRANDELLO ou DAMAS incluídos neste livro). Mas mesmo em programas como estes, deverá ainda ter como objectivo produzi-los com uma velocidade que seja a mais rápida possível. O factor velocidade é geralmente a primeira prioridade em jogos de gráficos animados. Gráficos lentos conduzem a um programa insatisfatório.

Na introdução a alguns dos programas deste livro, dirigimos a nossa atenção para vários dos seus aspectos que ajudam a maximizar a sua velocidade de execução. Pensámos que seria útil

reunir essas sugestões, acrescentar ainda mais algumas, neste capítulo.

Quando um programa chega a uma instrução GO TO ou GO SUB, tem de pesquisar, linha por linha, desde o primeiro número de linha até encontrar o número designado na instrução GO TO ou GO SUB. Portanto, os programas que colocam sub-rotinas perto do início do programa tendem a correr mais rapidamente do que os que as colocam no fim da listagem. Se houver um ciclo mestre, chamado várias vezes num programa, vale a pena organizar a listagem no sentido de colocar este ciclo o mais perto possível do início. Vêm-se muito programas assim organizados ao longo deste livro. Em contrapartida, as partes do programa utilizadas uma única vez (como a inicialização das variáveis, as instruções ao jogador e os gráficos definidos pelo utilizador) deverão ser colocadas mesmo no fim do programa, para assegurar que não se tem de saltar por cima delas, como aconteceria cada vez que o computador começasse a procurar pelo destino de uma instrução GO TO ou GO SUB.

Ver-se-á que, de maneira geral, quanto menor for o número de linhas do programa mais rapidamente ele correrá. Instruções "encadeadas" na mesma linha são executadas mais rapidamente do que seriam em linhas consecutivas de apenas uma instrução. Este facto nota-se especialmente em programas com gráficos animados; por isso deve-se tentar encadear o maior número de instruções PRINT AT (com ponto e vírgula) sempre que se possa. Geralmente, a quebra de velocidade como resultado de se terem muitas linhas bem como abundantes instruções REM não é nem crítica nem particularmente notada. No entanto, quando a velocidade tem de ser optimizada, valerá a pena fazer duas versões do mesmo programa. A primeira, com vários REM's e usando linhas com uma instrução apenas, para haver a máxima clareza durante os estágios de desenvolvimento e de pesquisa de erros, pode ser transformada numa segunda versão. Este segundo programa dispensa REMs e faria o maior número possível de "empacotamentos" de instruções na mesma linha.

Consegue-se por vezes eliminar completamente certas instruções GO TO, graças a cuidadosa programação. Examinem-se

todos os GO TO que não sejam qualificados por um IF/THEN para se ver se é possível eliminá-los. Alguns puristas da programação proclamam que um GO TO não qualificado é sempre mau, mas isso não é razão para banir tão útil instrução. No entanto, deve-se olhar com olhos de ver para os programas que usam GO TO, especialmente GO TOs não condicionais, por forma a descobrir maneira de os contornar. E, muitas vezes, podem ser completamente eliminados. Se há apenas uma curta sub-rotina só chamada duas ou três vezes num programa, talvez seja preferível incluir essa sub-rotina completa, cada vez que for necessária, em vez de consumir tempo com instruções GO SUB.

Uma vez que um programa esteja a funcionar, vale a pena fazer uma listagem na impressora (quem a tiver, claro), a fim de examinar com objectividade "blocos" gerais de maior ou menor auto-insuficiência. Descobrir-se-á que se podem encerrar muitos programas desta maneira, mesmo que não tenham sido programados com uma estrutura modular em mente. Sondado desta forma o programa, pode-se descobrir que os blocos não estão na melhor das ordens para assegurar a execução lógica do programa o mais rápido possível. Tentar-se-á então dispor os blocos numa ordem mais "limpa" e, talvez, mais lógica.

Não se devem usar parêntesis a não ser quando absolutamente necessários, porque ocupam memória e tempo de processamento. O *Spectrum*, em geral, exige menos parêntesis do que a maior parte dos outros *Basics*, permitindo, por exemplo, PRINT CHR\$ 134 em vez de PRINT CHR\$ (134).

É natural haver problemas para fazer caber num *Spectrum* de 16 K programas escritos originalmente para um ZX81 de 16 K. Os gráficos consomem uma porção significativa da memória dum *Spectrum* de 16 K. Ao converter, deixem-se de fora todos os comentários REM e condensem-se as instruções PRINT o mais que se possa. Examine-se a listagem completa para verificar se pode ser encurtada de alguma forma, como substituindo frases ou palavras usadas frequentemente por vectores literais ou por adição duma sub-rotina para congregar blocos de instruções utilizadas mais do que uma vez no programa. Isto abrandará um pouco a velocidade de execução, mas é um pequeno preço que se

paga para "espremer" o programa. Como se sabe, o *Spectrum* corre muito mais rapidamente do que o *ZX81*; por isso, este sacrifício de velocidade provavelmente constitui um problema menor. Se o programa corria a uma velocidade satisfatória no *ZX81*, só pode melhorar num *Spectrum*.

Apêndices

O GERADOR DE LETRA GÓTICA

Este programa não é um jogo, mas, sim, uma ajuda de programação e pode ser de execução bastante divertida. O programa permite definir um ou todos os caracteres impressos pelo *Spectrum*, com caracteres da nossa escolha. Tem pouco mais de trinta linhas de comprimento mas permite produzir efeitos extraordinários.

Aqui está, por exemplo, a listagem do programa com caracteres góticos criado por redefinição de todo o conjunto de caracteres:

```

100 clear 31400
200 input "que letra deseja rep
205 if (000 (a$)=226 then go to
400
210 let a=31145+8*(000 a$-32)
220 let f=0 to 7
230 let e$=chr$ (196)
240 input e$
245 if len e$ < 8 and e$ < " the
n bee p 5, 20: 80 to 240
250 let b=val (b$+e$)
255 print a+f, b
260 print f+1; tab 3; e$, b
270 next f
280 print
290 print a$
300 print
310 print
320 print
330 go to 200
400 print "por favor aguardar al
4105 moren - to 6, .."
420 let p=15016
430 let a=31401
440 for f=0 to 255
450 print a+f, chr$ (p+f)
460 next f
470 let v=31145
480 let n=23606
490 print n, v-256*int (v/256)
500 print n+1; int (v/256)
510 cls: stop

```


Como se pode verificar, é muito eficaz. Na maior parte das vezes não é necessário redefinir o alfabeto completo e bastará apenas alterar alguns caracteres. Faça o leitor correr o programa e espere alguns momentos até aparecer a frase "Que letra deseja reprogramar?". Tendo respondido, introduza (linha 240) um número BINÁRIO para C\$, que o computador processará, regressando para o seu próximo BINÁRIO. Quando terminar a redefinição, prima simplesmente na tecla STOP para deter o processo. Descobrirá agora — supondo que reprogramou a letra "a" —, que cada vez que carregar nesta mesma letra "a" obterá o próprio carácter que reprogramou.

Aqui está o programa nos caracteres habituais:

```

100 CLEAR 31400
200 INPUT "Que letra deseja reprogramar?" : a$
205 IF CODE (a$)=225 THEN GO TO 400
210 LET a=31145+8*(CODE a$-32)
220 FOR f=0 TO 7
230 LET b$=CHR$ (195)
240 INPUT c$
245 IF LEN c$<>8 AND c$<>" " THEN
  BEEP .5,20: GO TO 240
250 LET b=VAL (b$+c$)
255 POKE a+f,b
260 PRINT f+1;TAB 3;c$,b
270 NEXT f
280 PRINT
290 PRINT a$
300 PRINT
310 PRINT
320 PRINT
330 GO TO 200
400 PRINT "Por favor aguarde alguns
  moments - tos..."
420 LET p=15515
430 LET a=31401
440 FOR f=0 TO 255
450 POKE a+f,PEEK (p+f)
460 NEXT f
470 LET v=31145
480 LET n=23500
490 POKE n,v-255*INT (v/255)
500 POKE n+1,INT (v/255)
510 CLS : STOP

```

É mais simples reprogramar alfabetos completos usando entradas (INPUTS) decimais do que binárias. Se realmente necessita de um alfabeto gótico, acrescente as seguintes linhas:

```

1000>FOR a=31401 TO 32192
2000 INPUT (a);"?",p
3000 POKE a,p
4000 NEXT a
5000 GO TO 470

```

Quando fizer correr (com RUN 1000) ser-lhe-á oferecido um vasto número de entradas (INPUTS). Introduza apenas o número patente na lista seguinte, depois do número mostrado pelo computador. Feito isto, o seu alfabeto tornar-se-á gótico. É ideal para imprimir economicamente cartões de boas festas com o *Spectrum*.

31401	0	31403	0
31402	0	31404	0
31405	0	31406	0
31407	0	31408	0
31409	0	31410	16
31411	16	31412	16
31413	16	31414	0
31415	16	31416	0
31417	0	31418	36
31419	36	31420	0
31421	0	31422	0
31423	0	31424	0
31425	0	31426	36
31427	126	31428	36
31429	36	31430	126
31431	36	31432	0
31433	0	31434	8
31435	62	31436	40
31437	62	31438	10
31439	62	31440	8
31441	0	31442	38
31443	100	31444	8
31445	16	31446	38
31447	70	31448	0
31449	0	31450	16
31451	40	31452	16
31453	42	31454	62
31455	38	31456	0
31457	0	31458	8
31459	16	31460	0

1461	0	1462	0
1463	0	1464	0
1465	0	1466	0
1467	8	1468	8
1469	8	1470	8
1471	4	1472	0
1473	0	1474	32
1475	16	1476	16
1477	16	1478	16
1479	32	1480	0
1481	0	1482	0
1483	20	1484	8
1485	62	1486	8
1487	20	1488	0
1489	0	1490	8
1491	8	1492	32
1493	8	1494	8
1495	8	1496	0
1497	0	1498	0
1499	0	1500	0
1501	0	1502	8
1503	8	1504	16
1505	0	1506	0
1507	0	1508	0
1509	62	1510	0
1511	0	1512	0
1513	0	1514	0
1515	0	1516	0
1517	0	1518	24
1519	24	1520	0
1521	0	1522	4
1523	2	1524	16
1525	32	1526	0
1527	32	1528	22
1529	0	1530	66
1531	38	1532	56
1533	100	1534	24
1535	16	1536	24
1537	0	1538	0
1539	56	1540	24
1541	24	1542	24
1543	60	1544	0
1545	0	1546	56
1547	76	1548	36
1549	12	1550	16
1551	126	1552	0
1553	0	1554	126
1555	4	1556	8
1557	28	1558	70
1559	124	1560	0
1561	0	1562	12
1563	28	1564	44

1565	126	1566	12
1567	306	1568	0
1569	0	1570	126
1571	64	1572	124
1573	6	1574	66
1575	60	1576	0
1577	0	1578	24
1579	48	1580	112
1581	76	1582	100
1583	56	1584	0
1585	66	1586	126
1587	24	1588	4
1589	48	1590	16
1591	0	1592	0
1593	26	1594	28
1595	28	1596	24
1597	60	1598	28
1599	0	1600	0
1601	0	1602	28
1603	38	1604	50
1605	14	1606	12
1607	24	1608	0
1609	0	1610	0
1611	0	1612	16
1613	0	1614	0
1615	16	1616	0
1617	0	1618	0
1619	16	1620	0
1621	0	1622	16
1623	16	1624	32
1625	0	1626	0
1627	4	1628	8
1629	16	1630	8
1631	4	1632	0
1633	0	1634	0
1635	0	1636	62
1637	0	1638	62
1639	0	1640	0
1641	0	1642	0
1643	16	1644	8
1645	4	1646	8
1647	16	1648	0
1649	0	1650	60
1651	66	1652	4
1653	8	1654	0
1655	8	1656	0
1657	0	1658	60
1659	74	1660	86
1661	94	1662	64
1663	60	1664	0
1665	0	1666	56
1667	72	1668	24

01669	40	01670	104
01671	124	01672	0
01673	52	01674	96
01675	52	01676	56
01677	52	01678	52
01679	52	01680	524
01681	16	01682	56
01683	104	01684	96
01685	96	01686	96
01687	48	01688	34
01689	16	01690	52
01691	16	01692	24
01693	44	01694	44
01695	44	01696	16
01697	16	01698	56
01699	104	01700	12
01701	96	01702	96
01703	48	01704	24
01705	14	01706	26
01707	16	01708	60
01709	16	01710	16
01711	48	01712	24
01713	16	01714	40
01715	108	01716	44
01717	28	01718	44
01719	70	01720	60
01721	52	01722	52
01723	44	01724	54
01725	54	01726	44
01727	2	01728	44
01729	16	01730	52
01731	0	01732	48
01733	88	01734	24
01735	26	01736	28
01737	34	01738	0
01739	56	01740	24
01741	24	01742	24
01743	26	01744	96
01745	52	01746	96
01747	230	01748	107
01749	114	01750	108
01751	98	01752	19
01753	16	01754	48
01755	16	01756	16
01757	16	01758	24
01759	28	01760	126
01761	48	01762	42
01763	170	01764	42
01765	42	01766	42
01767	106	01768	0
01769	0	01770	40
01771	116	01772	52

01773	52	01774	52
01775	52	01776	0
01777	24	01778	44
01779	108	01780	108
01781	108	01782	108
01783	108	01784	48
01785	56	01786	116
01787	52	01788	52
01789	56	01790	48
01791	48	01792	64
01793	0	01794	48
01795	88	01796	48
01797	88	01798	56
01799	12	01800	8
01801	0	01802	44
01803	116	01804	52
01805	52	01806	52
01807	96	01808	48
01809	2	01810	28
01811	52	01812	108
01813	118	01814	6
01815	108	01816	48
01817	16	01818	48
01819	126	01820	48
01821	48	01822	48
01823	114	01824	60
01825	0	01826	40
01827	116	01828	52
01829	52	01830	52
01831	0	01832	10
01833	118	01834	54
01835	50	01836	50
01837	60	01838	54
01839	0	01840	16
01841	0	01842	24
01843	274	01844	107
01845	107	01846	107
01847	106	01848	20
01849	54	01850	24
01851	24	01852	60
01853	24	01854	24
01855	42	01856	68
01857	108	01858	52
01859	28	01860	52
01861	124	01862	64
01863	48	01864	12
01865	126	01866	68
01867	6	01868	28
01869	6	01870	2
01871	12	01872	48
01873	0	01874	14
01875	8	01876	8

011877	8	011878	8
011879	14	011880	0
011881	0	011882	0
011883	64	011884	32
011885	16	011886	8
011887	4	011888	0
011889	0	011890	56
011891	72	011892	24
011893	40	011894	104
011895	124	011896	0
011897	32	011898	16
011899	56	011900	84
011901	16	011902	16
011903	16	011904	0
011905	0	011906	0
011907	0	011908	0
011909	0	011910	0
011911	0	011912	255
011913	0	011914	28
011915	24	011916	120
011917	32	011918	32
011919	126	011920	0
011921	0	011922	56
011923	72	011924	24
011925	40	011926	104
011927	124	011928	0
011929	32	011930	96
011931	32	011932	56
011933	52	011934	52
011935	52	011936	24
011937	16	011938	56
011939	104	011940	96
011941	96	011942	96
011943	48	011944	24
011945	16	011946	32
011947	16	011948	24
011949	44	011950	44
011951	44	011952	16
011953	16	011954	56
011955	104	011956	120
011957	96	011958	96
011959	48	011960	24
011961	14	011962	26
011963	16	011964	60
011965	16	011966	16
011967	48	011968	24
011969	16	011970	40
011971	108	011972	44
011973	28	011974	44
011975	70	011976	60
011977	32	011978	32
011979	44	011980	54
011981	34	011982	34

011983	2	011984	4
011985	16	011986	32
011987	0	011988	48
011989	88	011990	24
011991	26	011992	28
011993	24	011994	0
011995	56	011996	24
011997	24	011998	24
011999	16	020000	96
020001	32	020002	96
020003	230	020004	107
020005	114	020006	108
020007	98	020008	19
020009	16	020010	48
020011	16	020012	16
020013	16	020014	16
020015	28	020016	24
020017	42	020018	126
020019	170	020020	42
020021	42	020022	42
020023	106	020024	0
020025	0	020026	40
020027	116	020028	52
020029	52	020030	52
020031	52	020032	0
020033	24	020034	44
020035	108	020036	108
020037	108	020038	108
020039	104	020040	48
020041	56	020042	116
020043	52	020044	52
020045	56	020046	48
020047	48	020048	64
020049	0	020050	48
020051	88	020052	88
020053	88	020054	56
020055	12	020056	8
020057	0	020058	44
020059	116	020060	52
020061	32	020062	32
020063	96	020064	48
020065	2	020066	28
020067	32	020068	108
020069	118	020070	6
020071	108	020072	48
020073	16	020074	48
020075	126	020076	48
020077	48	020078	48
020079	114	020080	60
020081	0	020082	40
020083	116	020084	52
020085	52	020086	52
020087	52	020088	10

322089	0	322090	34
322091	113	322092	50
322093	50	322094	54
322095	60	322096	16
322097	0	322098	34
322099	234	322100	107
322101	107	322102	107
322103	106	322104	20
322105	54	322106	34
322107	34	322108	60
322109	24	322110	24
322111	42	322112	63
322113	103	322114	33
322115	53	322116	33
322117	134	322118	64
322119	43	322120	12
322121	126	322122	63
322123	3	322124	23
322125	6	322126	2
322127	12	322128	43
322129	0	322130	14
322131	3	322132	43
322133	3	322134	3
322135	14	322136	0
322137	0	322138	3
322139	3	322140	3
322141	3	322142	3
322143	3	322144	0
322145	0	322146	0
322147	0	322148	0
322149	0	322150	0
322151	0	322152	0
322153	0	322154	20
322155	40	322156	0
322157	0	322158	0
322159	0	322160	0
322161	60	322162	66
322163	153	322164	161
322165	161	322166	153
322167	66	322168	60
322169	0	322170	0
322171	0	322172	0
322173	0	322174	0
322175	0	322176	0
322177	0	322178	0
322179	0	322180	0
322181	0	322182	0
322183	0	322184	0
322185	0	322186	0
322187	0	322188	0
322189	0	322190	0
322191	0	322192	0

RENUMERAÇÃO COM CÓDIGO MÁQUINA

Eis um programa em código máquina que irá renumerar um programa em *Basic*, alterando onde necessário os números que se seguem às instruções GO TO, GO SUB, RESTORE, LINE e RUN. Está escrito para o endereço 5CD0, ou seja, a primeira linha de programa. Os endereços podem ser alterados (os endereços apropriados estão sublinhados), mas esse processo não se recomenda.

O programa funciona deste modo:

1. O computador procura pelos *bytes* das instruções RUN, GO TO, GO SUB e RESTORE através do programa, verificando que eles não pertencem a um comentário REM, instrução PRINT, número de linha. Se se alcança o fim do programa, este é renumerado. Se for encontrado o *byte* de uma instrução GO TO ou GO SUB ou semelhante, o programa verifica se CHR\$ 14 aparece nos próximos cinco caracteres. Caso não apareça, o programa salta para o início.

O programa agora carrega em bloco o número na variável A.

2. Seguidamente, o computador faz o POKE do número na segunda parte do programa — agora em hexadecimal. Procura os números de linha que aparecem a seguir aos GO TO's e GO SUB's e, se estes forem maiores do que a última linha do programa, vai de volta para a alínea 1. Converte a posição do número de linha seguinte ao GO TO ou GO SUB no seu novo destino, convertendo, logicamente, este número num vector literal em A\$. Agora carrega em bloco o novo destino dos cinco *bytes* que se seguem a CHR\$ 14.

3. O programa agora carrega o vector literal na posição imediatamente a seguir aos GO TO, GO SUB e similares. Apagará ou introduzirá os espaços necessários para acondicionar o vector literal, ao mesmo tempo que altera as variáveis do sistema e o comprimento da linha do programa. Vai então de volta para a alínea 1. Para meter o programa introduza o carregador de números hexadecimais escrito em *Basic*, e um comentário REM contendo exactamente 372 caracteres, atribuindo-lhe o número de linha 1. Isto ocupará 11 linhas e 20 caracteres. Note-se que não há espaços dentro do b\$, na linha 2.

Faz-se depois correr o carregador hexadecimal, introduzindo os pares escritos em hexadecimal dados na segunda listagem. O carregador aceitará qualquer número de *bytes* de cada vez, devendo o código ser escrito sem espaços entre os *bytes*. O código usa maiúsculas e permite reintroduzir quaisquer números que ele não reconheça. No fim da introdução dos dados o programa pára com o relatório de código 9/18:1/.

Verifique-se o seguinte:

PEEK 23765 deve dar 33
PEEK 23891 deve dar 11
PEEK 23993 deve dar 33
PEEK 24072 deve dar 93

Deve-se agora apagar todo o programa original à excepção da linha 1.

Introduz-se a listagem 3, notando que a linha 5 contém o tamanho do passo e o primeiro número. Neste programa, foi-lhes atribuído o número 10, mas pode-se alterá-los à sua vontade. No entanto, ambos devem ter quatro caracteres de comprimento, e por esta razão temos dois zeros à frente do número 10. Ao tentar alterar estes valores, deve-se fazer POKE 23901 com o primeiro número menos o tamanho do STEP (passo), e fazer POKE 23909 com o tamanho do STEP. Apenas se introduz RUN e ENTER para renumerar um programa. Sugere-se que se mantenha o RENUMERADOR numa *cassete* separada e se faça o MERGE com o programa que se está a desenvolver e que é necessário renumerar.

Listagem um, o carregador hexadecimal:

```
2>LET b$="0123456789ABCDEF"
3 FOR a=23760 TO 24131
4 INPUT a$
5 PRINT a$( TO 2); " ";
6 LET c$=a$(1)
7 GO SUB 19
8 IF b=16 THEN GO TO 4
9 LET c=b*16
10 LET c$=a$(2)
```

```
11 GO SUB 19
12 IF b=16 THEN GO TO 4
13 POKE a,c+b
14 LET a$=a$(3 TO )
15 LET a=a+NOT NOT LEN a$
16 IF LEN a$ THEN GO TO 5
17 NEXT a
18 STOP
19 FOR b=0 TO 15
20 IF b$(b+1) <> c$ THEN NEXT b
21 RETURN
```

Listagem dois, o código hexadecimal:

4B	59	61	63	65	21	19	5F
23	23	23	23	22	F0	5C	09
1D	5F	16	60	FE	EC	08	FE
ED	08	FE	E5	08	FE	CA	08
FE	F7	09	EB	2A	4B	5C	37
ED	52	EB	09	E1	7E	E5	06
05	21	CF	5C	23	E5	6E	26
5C	A7	26	07	34	20	02	2C
34	18	05	35	20	02	2C	35
E1	10	E9	09	D5	CD	0C	5E
00	2A	65	5C	A7	ED	52	44
4D	62	6B	23	ED	B0	CD	FC
5C	00	D1	09	C5	D5	E5	2A
65	5C	E5	23	A7	ED	52	44
4D	19	D1	EB	ED	B8	CD	FC
5C	40	CD	0C	5E	40	E1	23
D1	C1	09	0B	2A	F0	5C	7E
CD	F3	5C	30	1D	01	00	00
2A	D6	5C	E5	21	0A	00	09

```

44 4D E1 CD F3 5C D8 70
23 71 23 5E 23 56 23 19
18 E9 23 11 04 00 FE EA
20 04 3E 0D 18 04 FE 22
20 05 47 ED B1 18 0A FE
0D 28 05 FE 0E 20 04 13
19 18 BC CD E4 5C 20 B7
22 E0 5C 3E 0E 01 05 00
ED B1 20 A7 22 E2 5C ED
58 4B 5C 13 0E 05 ED B0
C9 21 D1 5C 36 D8 2A D6
5C 01 00 00 03 56 23 5E
23 E5 21 0A 00 37 ED 52
E1 D8 5E 23 56 23 19 CD
F3 5C 30 E8 01 00 00 C9
2A 4B 5C 23 ED 5B E2 5C
01 05 00 ED B0 ED 5B E0
5C 23 4E 23 46 23 1A FE
0E CC 34 5C ED A0 78 B1
20 F4 1A FE 0E C8 CD 1C
5C 18 F7 C9 21 D1 5C 36
C8 2A E0 5C 3E 0D ED B9
E5 D5 23 23 46 23 4E 03
ED 43 C6 5C CD BE 5C D1
E1 78 B1 28 E7 01 04 00
09 C1 0A 05 A7 28 07 34

```

```

20 02 23 34 18 05 35 20
02 23 35 C9

```

Listagem três, o programa de renenumeração:

```

2>LET a=USR 23785
3 IF USR 23891 THEN STOP
4 POKE 24011,a-INT (a/256)*25
5: POKE 24012,INT (a/256): LET a
=USR 23993: IF NOT a THEN GO TO
3
5 LET a=(a-1)*0010+0010: LET
a$=STR$ a: GO TO 3 AND NOT USR 2
4032
6 REM

```

Listagem quatro, o renenumador em *assembly*:

```

5CD0 10 ORG #5CD0
5CD0 20 DEFB #4B
5CD1 30 DEFB #59
5CD2 40 DEFB #61
5CD3 50 DEFB #63
5CD4 60 DEFB #65
5CD5 70 LD HL,#5F19
5CD6 80 INC HL
5CD7 90 INC HL
5CD8 100 INC HL
5CD9 110 INC HL
5CDA 120 LD (L5CE0),H
5CDF 130
5CE0 140 L5CE0 RET
5CE1 150 DEFB #1D
5CE2 160 L5CE2 DEFB #5F
5CE3 170 DEFB #16
5CE4 180 L5CE4 DEFB #60
5CE5 190 CP #EC
5CE6 200 RET Z
5CE7 210 CP #E5
5CE8 220 RET Z
5CE9 230 CP #CA
5CEA 240 CP Z
5CEB 250 RET #F7
5CEC 260 RET
5CF0 270 L5CF3 EX DE,HL
5CF1 280 LD HL, (#5C4B)
5CF4 290 SCF
5CF5 300 SBC HL,DE
5CF7 310 EX DE,HL

```

5000	310	RET	HL
5001	320	POP	HL
5002	330	LD	A, (HL)
5003	340	PUSH	HL
5004	350	LD	B, #05
5005	360	LD	HL, #50CF
5006	370	INC	HL
5007	380	PUSH	HL
5008	390	LD	L, (HL)
5009	400	LD	H, #5C
500A	410	AND	A
500B	420	JR	Z, LSD13
500C	430	INC	(HL)
500D	440	JR	NZ, LSD11
500E	450	INC	L
500F	460	INC	(HL)
5010	470	JR	LSD18
5011	480	DEC	(HL)
5012	490	JR	NZ, LSD18
5013	500	INC	L
5014	510	DEC	(HL)
5015	520	POP	HL
5016	530	DJNZ	LSD04
5017	540	RET	
5018	550	PUSH	DE
5019	560	CALL	LSE0C
501A	570	NOP	
501B	580	LD	HL, (#5C65
501C	590	AND	A
501D	600	SBC	HL, DE
501E	610	LD	B, H
501F	620	LD	C, L
5020	630	LD	H, D
5021	640	LD	L, E
5022	650	INC	HL
5023	660	LDIR	
5024	670	CALL	LSCFC
5025	680	NOP	
5026	690	POP	DE
5027	700	RET	
5028	710	PUSH	BC
5029	720	PUSH	DE
502A	730	PUSH	HL
502B	740	LD	HL, (#5C65
502C	750	PUSH	HL
502D	760	INC	HL
502E	770	AND	A
502F	780	SBC	HL, DE
5030	790	LD	B, H
5031	800	LD	C, L
5032	810	ADD	HL, DE

503F	820	POP	DE
5040	830	EX	DE, HL
5041	840	LDDR	
5042	850	CALL	LSCFC
5043	860	LD	B, B
5044	870	CALL	LSE0C
5045	880	LD	B, B
5046	890	POP	HL
5047	900	INC	HL
5048	910	POP	DE
5049	920	POP	BC
504A	930	RET	
504B	940	DEC	BC
504C	950	LD	HL, (L5CE0
504D	960	LD	A, (HL)
504E	970	CALL	LSCF3
504F	980	JR	NC, LSD7A
5050	990	LD	BC, #0000
5051	1000	LD	HL, (#5CD6
5052	1010	PUSH	HL
5053	1020	LD	HL, #000A
5054	1030	ADD	HL, BC
5055	1040	LD	B, H
5056	1050	LD	C, L
5057	1060	POP	HL
5058	1070	CALL	LSCF3
5059	1080	RET	C
505A	1090	LD	(HL), B
505B	1100	INC	HL
505C	1110	LD	(HL), C
505D	1120	INC	HL
505E	1130	LD	E, (HL)
505F	1140	INC	HL
5060	1150	LD	D, (HL)
5061	1160	INC	HL
5062	1170	ADD	HL, DE
5063	1180	JR	LSD63
5064	1190	INC	HL
5065	1200	LD	DE, #0004
5066	1210	CP	#EA
5067	1220	JR	NZ, LSD86
5068	1230	LD	A, #0D
5069	1240	JR	LSD8A
506A	1250	CP	#22
506B	1260	JR	NZ, LSD8F
506C	1270	LD	B, A
506D	1280	CP	#0D
506E	1290	JR	LSD99
506F	1300	CP	#0D
5070	1310	JR	Z, LSD98


```

SD90 1320 CP #0E
SD92 1330 JR NZ,LSD9B
SD94 1340 INC DE
SD96 1350 ADD HL,DE
SD98 1360 JR LSC57
SD9B 1370 CALL LSC54
SD9D 1380 NZ,LSD57
SD9D 1390 LD (LSC50),H

SDA0 1400 LD A,#0E
SDA2 1410 LD BC,#0005
SDA5 1420 CPIR
SDA7 1430 JR NZ,LSD53
SDA9 1440 LD (LSC52),H

SDAC 1450 LD DE,(#5C4B

)
SDB0 1460 INC DE
SDB1 1470 LD C,#05
SDB3 1480 LDIR
SDB5 1490 RET
SDB6 1500 LD HL,LSD01
SDB9 1510 LD (HL),#08
SDBB 1520 LD HL,(#5CD6

)
SDBE 1530 LD BC,#0000
SDC1 1540 LSC4 INC BC
SDC2 1550 LD D,(HL)
SDC3 1560 INC HL
SDC4 1570 LD E,(HL)
SDC5 1580 INC HL
SDC6 1590 PUSH HL
SDC7 1600 LD HL,#000A
SDC8 1610 SCF
SDC9 1620 SBC HL,DE
SDCA 1630 POP HL
SDCB 1640 RET C
SDCC 1650 LSC01 LD E,(HL)
SDCD 1660 INC HL
SDCE 1670 LD D,(HL)
SDCF 1680 INC HL
SDD0 1690 ADD HL,DE
SDD1 1700 CALL LSCF3
SDD2 1710 JR NC,LSDC4
SDD3 1720 LD BC,#0000
SDD4 1730 RET
SDD5 1740 LD HL,(#5C4B

)
SDE0 1750 INC HL
SDE1 1760 LD DE,(LSC52
)
SDE5 1770 LD BC,#0005

```

```

SDE8 1780 LDIR
SDEA 1790 LD DE,(LSC50

)
SDEE 1800 INC HL
SDEF 1810 LD C,(HL)
SDF0 1820 INC HL
SDF1 1830 LD B,(HL)
SDF2 1840 INC HL
SDF3 1850 LSCF6 LD A,(DE)
SDF4 1860 CP #0E
SDF6 1870 CALL Z,LSD34
SDF8 1880 LDI
SDFB 1890 LD A,B
SDFC 1900 OR C
SDFD 1910 JR NZ,LSCF6
SDFE 1920 LSC02 LD A,(DE)
SD00 1930 CP #0E
SD02 1940 RET Z
SD03 1950 CALL LSD1C
SD06 1960 JR LSC02
SD08 1970 RET
SD09 1980 LSC0C LD HL,LSD01
SD0C 1990 LD (HL),#08
SD0E 2000 LD HL,(LSC50

)
SD11 2010 LSC14 LD A,#00
SD13 2020 CPDR
SD15 2030 PUSH HL
SD16 2040 PUSH DE
SD17 2050 INC HL
SD18 2060 INC HL
SD19 2070 LD B,(HL)
SD1A 2080 INC HL
SD1B 2090 LD C,(HL)
SD1C 2100 INC BC
SD1D 2110 LD (#5DCB),B

)
SD21 2120 CALL LSDBE
SD24 2130 POP DE
SD25 2140 POP HL
SD26 2150 LD A,B
SD27 2160 OR C
SD28 2170 JR NZ,LSC14
SD2A 2180 LD BC,#0004
SD2B 2190 ADD HL,BC
SD2C 2200 POP BC
SD2D 2210 LD A,(BC)
SD2E 2220 PUSH BC
SD30 2230 AND A
SD31 2240 JR NZ,LSC3E
SD32 2250 INC (HL)
SD34 2260 JR NZ,LSC3C
SD35 2270

```

```

55 00000000 00000000 INC HL
56 00000000 00000000 INC (HL)
57 00000000 00000000 JR LSE43
58 00000000 00000000 DEC (HL)
59 00000000 00000000 JR NZ,LSE43
60 00000000 00000000 INC HL
61 00000000 00000000 DEC (HL)
62 00000000 00000000 RET

```

BIBLIOTECA VERBO DE INFORMÁTICA

A seguir, nesta colecção:

Aprofundamento do Basic
de Mike Lord

O Domínio do Código Máquina
de Toni Baker

As 40 Melhores Rotinas em Código Máquina
de John Hardman e Andrew Hewson

Os 20 Melhores Programas para o ZX Spectrum
de Andrew Hewson

Guia Avançado para o Spectrum
de Mike Lord