Este livro é destinado aos possuidores do ZX SPECTRUM (PLUS, 2048, 2068), que ultrapassaram já a fase do simples divertimento, dominam minimamente o BASIC e sentem a necessidade de evoluir, melhorando os seus próprios programas.

Não iremos insistir nas funções do teclado, nem abordaremos os conceitos elementares de programação. Se o leitor necessitar de conhecimentos neste âmbito, tem à sua disposição numerosa bibliografia, mais ou menos desenvolvida, sobre o assunto.

Queremos, sim, compartilhar com o leitor a nossa própria experiência, facultando-lhe, através de inúmeros programas em BASIC e/ou código máquina, a possibilidade de atingir um razoável grau de qualidade, só ultrapassado nos programas comerciais.

Obviamente, não nos limitaremos a apresentar listagens para o leitor introduzir na máquina e executar. Cada programa será analisado exaustivamente, em certos casos linha a linha, pois desejamos que o leitor compreenda o funcionamento e a finalidade de cada instrução. Para as rotinas em código máquina incluídas nalguns programas, limitar-nos-emos a dar a respectiva listagem em números decimais e a explicar a sua aplicação.

É altura de entrarmos no assunto: siga cuidadosamente as nossas instruções e advertências. Procure compreender o funcionamento dos programas, antes de os executar: verá que o seu tempo foi bem empregue.

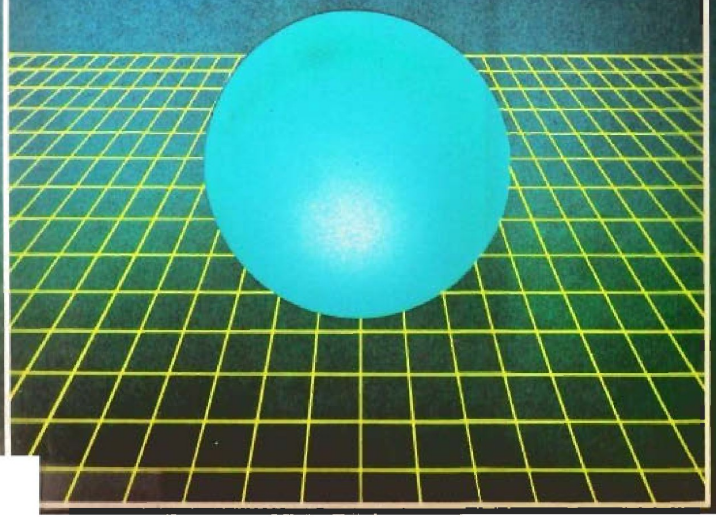
Informática

EUROPA AMÉRICA
Informática

Programe com qualidade no *ZX SPECTRUM, SPECTRUM PLUS, TIMEX 2048 e TIMEX 2068*

Renato Prista Casquilho

BASIC AVANÇADO



Livros publicados nesta colecção:

1 — *O Sinclair QL Funcional,* David Lawrence
2 — *Como Usar o Atari ST,* Jeremy Vine
3 — *Basic Avançado no ZX Spectrum,* Renato Prista Casquilho

Renato Prista Casquilho

BASIC AVANÇADO

Programe com qualidade
no *ZX SPECTRUM, SPECTRUM PLUS, TIMEX 2048 e TIMEX 2068*

Informática

PUBLICAÇÕES EUROPA-AMÉRICA

Capa: estúdios P. E. A.

Editor: Francisco Lyon de Castro

PUBLICAÇÕES EUROPA-AMÉRICA, LDA.
Apartado 8
2726 MEM MARTINS CODEX
PORTUGAL

Edição n.º 125003/4306

Execução técnica:
Gráfica Europam, Lda.,
Mira-Sintra — Mem Martins

Depósito legal n.º 14429/87

ÍNDICE

	Pág.
Apresentação	7
Capítulo I — *Os Abecedários*	9
Capítulo II	28
Capítulo III	66
Capítulo IV	96
Apêndice I	120
Apêndice II	122

APRESENTAÇÃO

Este livro é destinado aos possuidores do ZX SPECTRUM (PLUS, 2048, 2068), que ultrapassaram já a fase do simples divertimento, dominam minimamente o BASIC e sentem necessidade de evoluir, melhorando os seus próprios programas.

Não iremos insistir nas funções do teclado, nem abordaremos os conceitos elementares de programação. Se o leitor necessitar de conhecimentos neste âmbito, tem à sua disposição numerosa bibliografia, mais ou menos desenvolvida, sobre o assunto.

Queremos, sim, compartilhar com o leitor a nossa própria experiência, facultando-lhe, através de inúmeros programas em BASIC e/ou código máquina, a possibilidade de atingir um razoável grau de qualidade, só ultrapassado nos programas comerciais.

Obviamente, não nos limitaremos a apresentar listagens para o leitor introduzir na máquina e executar. Cada programa será analisado exaustivamente, em certos casos linha a linha, pois desejamos que o leitor compreenda o funcionamento e a finalidade de cada instrução. Para as rotinas em código máquina incluídas nalguns programas, limitar-nos-emos a dar a respectiva listagem em números decimais e a explicar a sua aplicação.

É altura de entrarmos no assunto: siga cuidadosamente as nossas instruções e advertências. Procure compreender o funcionamento dos programas, antes de os executar: verá que o seu tempo foi bem empregue.

Lisboa, Dezembro de 1986

CAPÍTULO I

OS ABECEDÁRIOS

1 — INTRODUÇÃO

O leitor já conhece o abecedário do seu ZX SPECTRUM: tem letras minúsculas e maiúsculas, algarismos, sinais de pontuação e uma série de outros caracteres, com múltiplas utilizações. Certamente, quando programa os seus textos sente a falta dos acentos que, em português, são quase sempre indispensáveis para uma correcta interpretação. Por outro lado, seria interessante, e por vezes mesmo conveniente, poder introduzir outros tipos de letra em certas passagens dos textos, tal como é possível fazer com os microcomputadores semiprofissionais.

Pois bem, o seu ZX SPECTRUM é capaz de tudo isto (e muito mais, como veremos adiante).

2 — OS CARACTERES DO ZX SPECTRUM

O conjunto dos 96 caracteres do ZX SPECTRUM de 48K encontra-se localizado na ROM, entre os endereços 15616 e 16383.

Vamos estudar, introduzir e executar um pequeno programa em BASIC que nos permitirá visualizar no écran, ou listar na impressora, o conjunto dos caracteres normais do SPECTRUM, dos códigos 32 a 127:

Programa 1/1

```
5000 CLS : LET a=1: FOR f=32 TO 127: PRINT CHR$ f;":";f;
" ";
5010 LET c=PEEK ((15616+(CODE CHR$ f-32)*8)+(a-1))
```

```
5020 PRINT c;",";: LET a=a+1: GO TO 5050
5030 IF f=127 THEN  STOP
5040 NEXT f
5050 IF a=9 THEN LET a=1: PRINT CHR$ 8;" ": GO TO 030
5060 GO TO 5010
```

Como funciona

Linha 500: damos à variável «a» o valor de 1 e iniciamos o ciclo FOR–NEXT em que a variável «f» irá tomar, sucessivamente, os valores de 32 a 127. Mandamos imprimir o carácter correspondente ao valor de «f» (quando «f» for, por exemplo, igual a 65, será impresso o carácter «A»), seguido de :, do valor de «f» (que será o código decimal do carácter) e, por último, o sinal de =. O ponto e vírgula no fim da linha indica ao computador que a próxima impressão deve localizar–se depois do sinal =.

Linha 5010: damos à variável «c» o valor do conteúdo do endereço 15616 + (o valor do código do carácter que foi impresso — 32) × 8) ao que se soma o valor de (a—1), ou seja 0. Exemplificando: se «f» for igual a 65 e «a»=1, o valor de «c» será o conteúdo do endereço 15880 (15616+264), ou seja 0 (experimente o leitor o comando directo: PRINT PEEK 15880).

Linha 5020: imprimir o valor de «c» seguido de uma vírgula e incrementar o valor de «a» de uma unidade. O programa passa para a linha 5050.

Linha 5050: quando «a» for igual a 9, volta a ter o valor de 1. Limpar a última vírgula. O comando volta à linha 5030.

Linha 5030: quando o valor de «f» atinge 127 o programa pára.

Linha 5040: fecho do ciclo, com incremento do valor de «f».

Linha 5060: o comando volta à linha 5010.

Exemplifiquemos:

• Quando iniciamos a execução do programa, com «a»=1 e «f»=32, o écran apresenta:

```
: 32 =
```

• A linha 5010 calcula o valor de «c» e a linha 5020 manda imprimir este valor. O écran apresenta:

```
: 32 = 0,
```

• A mesma linha 5020 incrementa agora o valor de «a» de uma unidade, passando «a»=2; o programa vai à linha 5050 e, como «a» não é igual a 9, passa à linha 5060, que manda a execução novamente para a linha 5010; é calculado o novo valor de «c», este é impresso e «a» é de novo incrementado. O écran apresenta:

```
: 32 = 0,0,
```

• O ciclo repete–se, até «a» atingir o valor de 9; o écran apresenta finalmente

```
: 32 = 0,0,0,0,0,0,0,0,
```

• O valor de «a» regressa a 1 e a instrução PRINT CHR$ 8;" " apaga a última vírgula. O écran apresentará:

```
: 32 = 0,0,0,0,0,0,0,0
```

• O comando passa à linha 5030, onde é certificado se «f» atingiu o valor de 127; como é inferior a 127, passa à linha 5040, onde o ciclo FOR–NEXT é incrementado, passando «f» a valer 33; «a» vale 1 e o écran apresenta:

```
! : 33 =
```

• O programa continua com o incremento do valor de «a» e o écran apresenta:

```
! : 33 = 0,16,16,16,16,0,16,0
```

• O ciclo repete–se até «f» ter o valor de 127; o écran apresentará:

```
© : 127 = 60,66,153,161,161,153,66,60
```

• O programa pára na linha 5030, pois «f» tem o valor de 127.

Que fez o programa?

A parte essencial encontra–se nas instruções das linhas 5010 e 5020, onde são calculados o endereço inicial de cada um dos 96 caracteres, que, como dissemos, se encontram entre os endereços 15616 e 16383 da ROM, bem como o respectivo conteúdo decimal.

Entretanto lembremos que cada carácter é constituído por oito BYTES, sendo o conteúdo de cada BYTE que define a «forma» do carácter.

Vejamos como se apresentam os caracteres dos exemplos anteriores:

Código do carácter «espaço» = 32

ENDEREÇO	Conteúdo em decimal	Conteúdo em binário
15616	0	00000000
15617	0	00000000
15618	0	00000000
15619	0	00000000
15620	0	00000000
15621	0	00000000
15622	0	00000000
15623	0	00000000

Figura 1/1

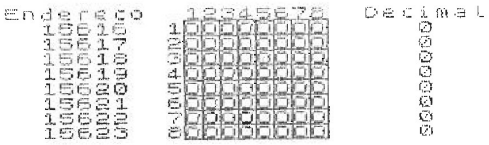
Código do carácter «copyright» = 127

ENDEREÇO	Conteúdo em decimal	Conteúdo em binário
16376	60	00111100
16377	66	01000010
16378	153	10011001
16379	161	10100001
16380	161	10100001
16381	153	10011001
16382	66	01000010
16383	60	00111100

Figura 2/1

Desenhemos uma rede de 8 × 8 *pixels* e, socorrendo-nos do conteúdo em «binário» de cada BYTE, vamos determinar a forma destes caracteres:

Carácter «espaço» = 32



Cada BYTE é constituído por oito «bits», correspondendo o valor «0» a PAPER e o valor «1» a INK. Desta forma, para o carácter «espaço», o computador nada imprime no écran, enquanto para o carácter do código 127 (copyright), são impressos os bits que contêm o valor «1», tal como é mostrado na figura 3/1:

Carácter «copyright» = 127

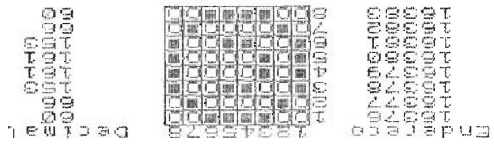


Figura 3/1

Lembramos, ainda, que a função PEEK lê o valor decimal contido em qualquer endereço, pelo que, se fizermos por comando directo: PRINT PEEK 15616, obteremos o valor «0», enquanto PRINT PEEK 16376 dará o valor «60».

Retomemos, então, as instruções das linhas 5010 e 5020:

```
LET c=PEEK ((15616+(CODE CHR$ f-32 )*8)+(a-1))
```

No início do ciclo FOR-NEXT, «f» tem o valor de 32 e «a» o valor de 1, donde:

```
c=PEEK ((15616+(32-32)x8)+(1-1)
```

donde:

```
c=PEEK ((15616+(0x8)+0)
```

donde:

```
c=PEEK 15616=0
```

O valor de «a» é incrementado de uma unidade:

```
c=PEEK ((15616+(32-32)x8+(2-1)
```

donde:

```
c=PEEK 15617=0
```

e assim sucessivamente, até «a» ter o valor de 9 (ver fig.1/1);

• O NEXT f da linha 5040 dá a «f» o valor de 33, que é o código do carácter «!»; «a» voltou a ter o valor de 1; assim:

```
c=PEEK ((15616+(127-32)x8+(1-1)
```

donde:

```
c=PEEK 16376=60
```

(ver fig. 2/1).

Nesta altura, o leitor certamente compreendeu a execução de todo o programa. Agora, dê entrada às instruções e faça RUN ou GO TO 1. O écran apresentará o conjunto dos 96 caracteres, os seus códigos e o conteúdo em decimal dos 8 BYTES que compõem cada carácter. Se dispuser de uma impressora, faça uma listagem e obterá um documento de trabalho de grande utilidade. Eis os resultados do programa 1/1:

CONJUNTO DOS CARACTERES NORMAIS DO ZX SPECTRUM

Códigos de 32 a 127 localizados na ROM entre os endereços 15616 e 16383

```
 :32=0,0,0,0,0,0,0,0,
!:33=0,16,16,16,16,0,16,0,
":34=0,36,36,0,0,0,0,0,
#:35=0,36,126,36,36,126,36,0,
$:36=0,8,62,40,62,10,62,8,
%:37=0,98,100,8,16,38,70,0,
&:38=0,16,40,16,42,68,58,0,
':39=0,8,16,0,0,0,0,0,
(:40=0,4,8,8,8,8,4,0,
):41=0,32,16,16,16,16,32,0,
*:42=0,0,20,8,62,8,20,0,
+:43=0,0,8,8,62,8,8,0,
,:44=0,0,0,0,0,8,8,16,
-:45=0,0,0,0,62,0,0,0,
.:46=0,0,0,0,0,24,24,0,
/:47=0,0,2,4,8,16,32,0,
0:48=0,60,70,74,82,98,60,0,
1:49=0,24,40,8,8,8,62,0,
2:50=0,60,66,2,60,64,126,0,
3:51=0,60,66,12,2,66,60,0,
4:52=0,8,24,40,72,126,8,0,
5:53=0,126,64,124,2,66,60,0,
6:54=0,60,64,124,66,66,60,0,
7:55=0,126,2,4,8,16,16,0,
8:56=0,60,66,60,66,66,60,0,
9:57=0,60,66,66,62,2,60,0,
::58=0,0,0,16,0,0,16,0,
;:59=0,0,16,0,0,16,16,32,
<:60=0,0,4,8,16,8,4,0,
=:61=0,0,0,62,0,62,0,0,
>:62=0,0,16,8,4,8,16,0,
?:63=0,60,66,4,8,0,8,0,
@:64=0,60,74,86,94,64,60,0,
A:65=0,60,66,66,126,66,66,0,
B:66=0,124,66,124,66,66,124,0,
C:67=0,60,66,64,64,66,60,0,
D:68=0,120,68,66,66,68,120,0,
E:69=0,126,64,124,64,64,126,0,
F:70=0,126,64,124,64,64,64,0,
G:71=0,60,66,64,78,66,60,0,
H:72=0,66,66,126,66,66,66,0,
I:73=0,62,8,8,8,8,62,0,
J:74=0,2,2,2,66,66,60,0,
K:75=0,68,72,112,72,68,66,0,
L:76=0,64,64,64,64,64,126,0,
M:77=0,66,102,90,66,66,66,0,
N:78=0,66,98,82,74,70,66,0,
O:73=0,60,66,66,66,66,60,0,
P:80=0,124,66,66,124,64,64,0,
Q:81=0,60,66,66,82,74,60,0,
R:82=0,124,66,66,124,68,66,0,
S:83=0,60,64,60,2,66,60,0,
T:84=0,254,16,16,16,16,16,0,
U:85=0,66,66,66,66,66,60,0,
V:86=0,66,66,66,66,36,24,0,
W:87=0,66,66,66,66,90,36,0,
X:88=0,66,36,24,24,36,66,0,
Y:89=0,130,68,40,16,16,16,0,
Z:90=0,126,4,8,16,32,126,0,
[:91=0,14,8,8,8,8,14,0,
\:92=0,0,64,32,16,8,4,0,
]:93=0,112,16,16,16,16,112,0,
↑:94=0,16,56,84,16,16,16,0,
_:95=0,0,0,0,0,0,0,255,
`:96=0,28,34,120,32,32,126,0,
a:97=0,0,56,4,60,68,60,0,
b:98=0,32,32,60,34,34,60,0,
c:99=0,0,28,32,32,32,28,0,
d:100=0,4,4,60,68,68,60,0,
e:101=0,0,56,68,120,64,60,0,
f:102=0,12,16,24,16,16,16,0,
g:103=0,0,60,68,68,60,4,56,
h:104=0,64,64,120,68,68,68,0,
i:105=0,16,0,48,16,16,56,0,
j:106=0,4,0,4,4,4,36,24,
k:107=0,32,40,48,48,40,36,0,
l:108=0,16,16,16,16,16,12,0,
m:109=0,0,104,84,84,84,84,0,
n:110=0,0,120,68,68,68,68,0,
o:111=0,0,56,68,68,68,56,0,
p:112=0,0,120,68,68,120,64,64,
q:113=0,0,60,68,68,60,4,6,
r:114=0,0,28,32,32,32,32,0,
```



```
U:85=0,66,66,66,66,66,60,0,
V:86=0,66,66,66,66,36,24,0,
W:87=0,66,66,66,66,90,36,0,
X:88=0,66,36,24,24,36,66,0,
Y:89=0,130,68,40,16,16,16,0,
Z:90=0,126,4,8,16,32,126,0,
[:91=0,14,8,8,8,8,14,0,
\:92=0,0,64,32,16,8,4,0,
]:93=0,112,16,16,16,16,112,0,
↑:94=0,16,56,84,16,16,16,0,
_:95=0,0,0,0,0,0,0,255,
`:96=0,28,34,120,32,32,126,0,
a:97=0,0,56,4,60,68,60,0,
b:98=0,32,32,60,34,34,60,0,
c:99=0,0,28,32,32,32,28,0,
d:100=0,4,4,60,68,68,60,0,
e:101=0,0,56,68,120,64,60,0,
f:102=0,12,16,24,16,16,16,0,
g:103=0,0,60,68,68,60,4,56,
h:104=0,64,64,120,68,68,68,0,
i:105=0,16,0,48,16,16,56,0,
j:106=0,4,0,4,4,4,36,24,
k:107=0,32,40,48,48,40,36,0,
l:108=0,16,16,16,16,16,12,0,
m:109=0,0,104,84,84,84,84,0,
n:110=0,0,120,68,68,68,68,0,
o:111=0,0,56,68,68,68,56,0,
p:112=0,0,120,68,68,120,64,64,
q:113=0,0,60,68,68,60,4,6,
r:114=0,0,28,32,32,32,32,0,
s:115=0,0,56,64,56,4,120,0,
t:116=0,16,56,16,16,16,12,0,
u:117=0,0,68,68,68,68,56,0,
v:118=0,0,68,68,40,40,16,0,
w:119=0,0,68,84,84,84,40,0,
x:120=0,0,68,40,16,40,68,0,
y:121=0,0,68,68,68,60,4,56,
z:122=0,0,124,8,16,32,124,0,
{:123=0,14,8,48,8,8,14,0,
|:124=0,8,8,8,8,8,8,0,
}:125=0,112,16,12,16,16,112,0,
~:126=0,20,40,0,0,0,0,0,
©:127=60,66,153,161,161,153,66,60
```

Figura 4/1

3 — OS ACENTOS

Para construir um abecedário com os acentos utilizados na língua portuguesa, torna-se necessário, em primeiro lugar, copiar o conjunto de caracteres normais para um qualquer endereço, superior a 23735. Sugerimos o endereço 64500, o que nos deixa cerca de 40 K de memória livre para um qualquer programa de aplicação do abecedário.

O conjunto dos 96 caracteres ocupa 768 bytes (96 × 8); logo, se o topo da memória (RAMTOP) for fixado no endereço 64500, ficamos com uma margem de segurança de 100 bytes até ao endereço 65368, que é o início da zona dos UDG: 64500 + 768 = 65268 → 65368-65268=100.

Para copiarmos o conjunto dos caracteres para o endereço 64500, utilizaremos a função POKE (na forma POKE endereço,decimal).

Vejamos as instruções:

```
20 FOR a = 0 TO 767 : POKE (64500 + a),PEEK ( 15616 + a )
:  NEXT a
```

Como funciona

• Iniciamos um ciclo FOR-NEXT, em que a variável «a» toma, sucessivamente, os valores de 0 a 767 (são 768 bytes);

• Fazemos POKE (introduzimos) no endereço 64500 + a o conteúdo decimal do endereço 15616 + a (PEEK → retiramos, lemos);

• O fecho do ciclo (NEXT a) vai incrementando o valor de «a», o qual é somado tanto ao endereço 64500 como ao endereço 15616. Deste modo, quando iniciamos o ciclo, e como já vimos (fig. 3/1), o valor «0» que está no endereço 15616 vai ser copiado para o endereço 64500; o valor seguinte, também 0, que está no endereço 15616 + 1 (15617), será copiado para o endereço 64500 + 1 (64501). E assim sucessivamente, até se terem copiado todos os 768 bytes.

Como somos adeptos da experimentação, não só como meio de comprovar a validade dos nossos raciocínios, como também para testarmos as instruções dadas ao computador, convidamos o leitor a introduzir este pequeno programa e, seguidamente, socorrendo-se da função PEEK, verificar se tudo está correcto. Utilize o pequeno programa que se segue e compare os resultados com os da figura 4/1:

PROGRAMA 2/1

```
10 LET endereco=64500
20 INPUT "Codigo do caracter?";n: IF n<32 OR n>127 THEN
GO TO 20
30 LET a=1 : PRINT CHR$ n;":";n;"=";
40 LET c=PEEK ((endereco+(CODE CHR$ n-32)*8)+(a-1)) 50
PRINT c;",";: LET a=a+1
60 IF a=9 THEN LET a=1: PRINT CHR$ 8;" ": GO TO 20
70 GO TO 40
```

Cremos não serem necessárias grandes explicações, pois este programa é muito semelhante ao programa 1/1 que o leitor já estudou. As diferenças consistem em:

Linha 10: atribuição do valor 64500 à variável «endereco».

Linha 20: o computador pede que indiquemos o código decimal do carácter a analisar. A instrução condicional impede que se introduzam valores inferiores a 32 ou superiores a 127.

Linhas 30 a 70: ciclo que imprime o carácter, o seu código e os seus oito decimais (ver fig. 4/1).

NOTA. — Quando quiser interromper o programa, introduza «STOP»: o programa pára com a mensagem «H STOP in INPUT, 20:1».

Agora, que temos o conjunto de caracteres copiado no endereço 64500, podemos passar à construção dos acentos. Trata-se de uma questão de compromisso: o leitor terá de prescindir de uma série de caracteres (normalmente não utilizados nos textos), para os modificar de forma a criar «letras com acentos», pois é a forma de mais fácil processamento do abecedário acentuado.

Vejamos as letras que, normalmente, são acentuadas e os caracteres que lhe propomos utilizar:

Figura 5/1

A modificação dos caracteres que escolhemos pode ser feita utilizando o programa GERADOR DE GRÁFICOS, o qual será apresentado e estudado no capítulo seguinte. Entretanto, recorreremos, mais uma vez, à função POKE, para introduzirmos, nos endereços respectivos, os novos decimais correspondentes às letras acentuadas. Vejamos, primeiro, o quadro completo:

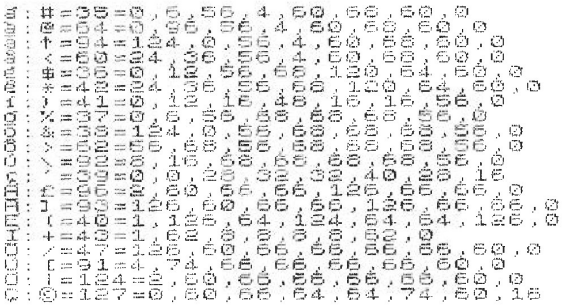


Figura 6/1

Utilize o programa seguinte para introduzir, por meio de POKE, os novos decimais nos endereços respectivos (sempre com início em 64500):

PROGRAMA 3/1

```
10 LET endereco = 64500: FOR i = 1 TO 20
20 BEEP .2,10: INPUT "Codigo do carácter? ";(i);" ";n: IF
n < 32 OR n > 127 THEN GO TO 20
30 FOR f = 0 TO 7
40 BEEP .2,30: INPUT "Decimal? ";(f + 1);" ";c$: IF c$
="" THEN GO TO 40
50 LET c = VAL c$: IF c < 0 OR c > 255 THEN GO TO 40
60 POKE ((endereco +(CODE CHR$ n - 32) * 8 + f),c
70 NEXT f
80 IF i = 20 THEN PRINT "FIM DA INTRODUCAO": BEEP 1,20:
STOP 90 NEXT i
```

O programa é muito semelhante ao programa 2/1. As diferenças consistem em:

Linha 10: Introdução de um ciclo FOR–NEXT que limita a 20 o número de caracteres a introduzir;

Linha 40: Introdução dos novos decimais, dentro de um ciclo FOR–NEXT, que limita a introdução a oito números. O BEEP inicial é útil, pois alerta para o pedido de «INPUT». O «artifício» (f + 1) permite visualizar o número de ordem do byte a introduzir. Finalmente, a utilização da variável alfanumérica «c$» impede qualquer entrada nula: (IF c$ ="" THEN GO TO 40);

Linha 50: A função VAL permite dar à variável «c» o valor contido na variável «c$». A condição seguinte impede a entrada de valores negativos ou superiores a 255, o que ocasionaria a paragem do programa;

Linha 60: Colocamos (POKE) no endereço correspondente ao carácter escolhido os decimais contidos em «c», até completarmos os 8 bytes (comando pelo ciclo FOR–NEXT das linhas 30 e 70);

Linha 80: Quando «i» for igual a 20 (fecho do ciclo), o écran apresenta a mensagem «FIM DA INTRODUCAO», acompanhada de um sinal sonoro;

Linha 90: Enquanto «i» for inferior a 20, o comando passa à linha 20, através do ciclo FOR–NEXT «i».

Sempre seguindo o mesmo princípio, sugerimos ao leitor que verifique os resultados, aplicando o programa 2/1, já anteriormente utilizado, mas antes, e isto é muito importante, grave em cassete os 768 bytes que acabou de criar, de modo a não ser obrigado a repetir todo o trabalho. Use o processo seguinte, em modo directo:

```
– SAVE "ABEC 1" CODE 64500,768
```

e verifique:

```
– VERIFY "" CODE
```

Não se esqueça de identificar a cassete, o lado e, se possível, o número do contador de rotações do seu gravador: este conjunto de bytes vai ser–lhe útil, muitas vezes.

Perguntará o leitor: muito bem, tenho um conjunto de caracteres com acentos no endereço 64500. Mas como tirar partido desta aquisição?

Iremos, seguidamente, tratar deste problema. Começaremos por determinar os «bytes equivalentes» do endereço 64500, isto é, o byte «menor» e o byte «maior», que aplicaremos à variável CHARS (23606 e 23607), por meio de POKE.

Os bytes equivalentes de qualquer endereço (ou número até 65535), podem determinar-se com o seguinte programa:

PROGRAMA 4/1

```
2500 INPUT "Endereco? ";endereco
2505 LET v = endereco - 256
2510 LET b1 = v - 256 * INT (v/256)
2515 LET b2 = INT (v/256)
2520 PRINT "Byte menor = ";b1 ' "Byte maior = ";b2
2525 PRINT "Endereco = ";endereco
```

Introduza o programa, execute-o e indique o endereço 64500. Verificará que o byte «menor» tem o valor de 244 e o byte «maior» tem o valor de 250. São estes os valores que terá de aplicar à variável de sistema «CHARS», por meio de POKE:

```
- POKE 23606,244
- POKE 23607,250
```

4 — ABECEDÁRIO 1

Vamos responder à sua pergunta. Contamos que tenha gravado no endereço 64500 os 768 bytes do novo conjunto de caracteres. Se não o fez, o que seria lamentável, terá de repetir o programa 3/1 e introduzir novamente os decimais da Fig.6/1.

Mas certamente que seguiu as nossas indicações. Introduza o programa que se segue:

PROGRAMA 5/1

```
 10 PRINT AT 0,2;"UTILIZACAO DO ABECEDARIO 1"
 20 PRINT '' "Prepare o gravador com a cassette onde gravou os 768 bytes do ABECEDARIO 1"
 30 PRINT "Prima ENTER quando estiver preparado e ligue o gravador."
 40 PAUSE 0: LOAD "" CODE 64500,768
 50 CLS: GOSUB 800
 60 PRINT AT 0,10; "ABECED£RIO 1": PRINT
 70 PRINT "Tem agora em mem%ria o ABECED£RIO 1 com acentos. Como pode verificar, o texto $ impresso com as letras acentuadas, pois s^o utilizados os caracteres que, para o efeito, foram criados."
 80 PRINT "Eis os novos caracteres:"
 90 PRINT: GOSUB 1000
100 GOSUB 900: PRINT
110 PRINT "Eis os caracteres normais:"
120 PRINT: GOSUB 1000: STOP
800 POKE 23606,244: POKE 23607,250: RETURN
900 POKE 23606,0: POKE 23607,60: RETURN
1000 FOR f = 32 TO 127: PRINT CHR$ f;: NEXT f: RETURN
```

Como funciona

Linha 40: A instrução PAUSE 0 indica ao computador que aguarde a pressão sobre qualquer tecla, antes de passar à instrução seguinte que, neste caso, é a instrução de carregamento, do gravador para o computador, dos 768 bytes do endereço 64500.

Linha 50: Apaga o écran e chama a sub–rotina da linha 800.

Linha 800: Sub–rotina que coloca na variável CHARS os bytes 244 e 250, permitindo a utilização dos caracteres guardados no endereço 64500.

Linha 60: Manda–se imprimir a palavra «ABECEDARIO 1», já com o acento agudo na letra «A»: utilizando o carácter «£» (libra).

Linha 70: Texto demonstrativo, em que são utilizados os caracteres:

```
- £ para o «Á» de «ABECEDÁRIO»
- % para o «ó» de «memória»
- $ para o «é» de «é»
- ^ para o «ã» de «são».
```

Linha 90: Chama a sub–rotina da linha 1000.

Linha 1000: Ciclo FOR–NEXT que imprime os caracteres dos códigos 32 a 127.

Linha 100: Chama a sub–rotina da linha 900.

Linha 900: Sub–rotina que repõe os valores normais na variável CHARS, passando–se aos caracteres normais do SPECTRUM.

Faça, agora, as suas próprias experiências e pratique o novo abecedário. Para o efeito, chame a linha 70 (EDIT) e modifique o texto à sua vontade, procurando aplicar o máximo possível de letras acentuadas. ENTRE de novo a linha e, por comando directo, faça «GO TO 50».

Dissemos, no início, que a construção dos acentos era uma questão de compromisso. Seremos mais claros: é evidente que a utilização dos vinte caracteres da fig. 6/1 para letras acentuadas impede o uso dos mesmos caracteres, na sua forma primitiva. Sempre que precisar de um «$», aparece-lhe um «ó»; se for um «(», aparece–lhe um «É»,

etc. O compromisso a estabelecer será para um máximo de letras acentuadas e um máximo de outros caracteres a utilizar.

Se, no seu texto, vai incluir expressões matemáticas, com parênteses e sinal «+», ou valores monetários com «$», ou ainda percentagens com «%», é óbvio que terá de prescindir de algumas letras acentuadas, escolhendo, para isso, as menos usuais, ou aquelas em que o acento possa ser dispensável.

Por outro lado, o leitor colocará as letras acentuadas nos caracteres que entender e julgar mais convenientes. A distribuição que demos na fig. 6/1 deverá ser entendida como uma simples sugestão.

5 — ABECEDÁRIO 2

Na introdução ao presente capítulo, referimos o interesse e a possibilidade de se criarem outros tipos de letra. Damos seguidamente um exemplo de um tipo de letra, dita «cheia» ou «negro», que proporcionará uma boa apresentação aos seus textos. Este novo «ABECEDÁRIO 2», cuja listagem dos respectivos decimais é dada na fig. 7/1, contém, do mesmo modo, letras acentuadas, com utilização dos mesmos caracteres.

ABECEDÁRIO 2

Lista dos caracteres, códigos e respectivos decimais

```
 !"#$%&'()*+,-./0123456789:;<=>?
@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_
`abcdefghijklmnopqrstuvwxyz{|}~©
  =32=0,0,0,0,0,0,0,0,
!=33=0,24,24,24,24,0,24,0,
"=34=0,36,36,0,0,0,0,0,
#=35=8,16,120,12,124,204,126,0,
$=36=4,8,120,204,248,192,124,0,
%=37=4,8,120,204,204,204,120,0,
&=38=120,0,120,204,204,204,120,0,
'=39=0,0,60,96,96,104,60,16,
(=40=3,254,192,252,192,192,254,0,
)=41=0,6,8,56,24,24,60,0,
*=42=48,72,120,204,248,192,124,0,
+=43=3,126,24,24,24,24,126,0,
,=44=0,0,0,0,0,24,24,48,
-=45=0,0,0,0,126,0,0,0,
.=46=0,0,0,0,0,56,56,0,
/=47=254,124,198,198,198,198,124,0,
0=48=0,124,198,222,246,198,124,0,
1=49=0,56,120,24,24,24,126,0,
2=50=0,124,198,6,124,192,254,0,
3=51=0,124,198,28,6,198,124,0,
4=52=0,24,56,88,216,254,24,0,
5=53=0,254,192,252,6,198,124,0,
6=54=0,124,192,252,198,198,124,0,
7=55=0,254,6,12,24,48,48,0,
8=56=0,124,198,124,198,198,124,0,
9=57=0,124,198,198,126,6,124,0,
:=58=0,0,0,48,0,0,48,0,
;=59=0,0,48,0,0,48,48,96,
<=60=48,72,120,12,124,204,126,0,
==61=0,0,62,62,0,62,62,0,
>=62=48,72,120,204,204,204,120,0,
?=63=0,60,126,70,12,12,0,12,
@=64=64,32,120,12,124,204,126,0,
A=65=0,124,198,198,254,198,198,0,
B=66=0,252,198,252,198,198,252,0,
C=67=0,124,198,192,192,198,124,0,
D=68=0,248,204,198,198,204,248,0,
E=69=0,254,192,252,192,192,254,0,
F=70=0,254,192,252,192,192,192,0,
G=71=0,124,198,192,222,198,124,0,
H=72=0,198,198,254,198,198,198,0,
I=73=0,126,24,24,24,24,126,0,
J=74=0,6,6,6,198,198,124,0,
K=75=0,204,216,240,216,204,198,0,
L=76=0,192,192,192,192,192,254,0,
M=77=0,198,238,254,198,198,198,0,
N=78=0,198,230,246,222,206,198,0,
O=79=0,124,198,198,198,198,124,0,
P=80=0,252,198,198,252,192,192,0,
Q=81=0,124,198,198,246,222,124,0,
R=82=0,252,198,198,252,204,198,0,
S=83=0,124,192,124,6,198,124,0,
T=84=0,252,48,48,48,48,48,0,
U=85=0,198,198,198,198,198,124,0,
V=86=0,198,198,198,198,108,56,0,
W=87=0,198,198,214,214,254,108,0,
X=88=0,199,108,56,56,108,198,0,
Y=89=0,134,204,120,48,48,48,0,
Z=90=0,254,12,24,48,96,254,0,
[=91=24,214,198,198,198,198,124,0,
\=92=24,16,204,204,204,204,120,0,
]=93=254,124,198,198,254,198,198,0,
^=94=120,0,120,12,124,204,126,0,
```

```
_=95=0,0,0,0,0,0,0,255,
`=96=3,124,198,198,254,198,198,0,
a=97=0,0,120,12,124,204,126,0,
b=98=0,96,96,124,102,102,124,0,
c=99=0,0,60,96,96,96,60,0,
d=100=0,12,12,124,204,204,124,0,
e=101=0,0,120,204,248,192,124,0,
f=102=0,28,48,120,48,48,48,0,
g=103=0,0,124,204,204,124,12,120,
h=104=0,192,192,248,204,204,204,0,
i=105=8,0,8,56,24,24,60,0,
j=106=0,12,0,12,12,12,108,56,
k=107=0,96,108,120,112,120,108,0,
l=108=0,96,96,96,96,96,56,0,
m=109=0,0,252,214,214,214,214,0,
n=110=0,0,248,204,204,204,204,0,
o=111=0,0,120,204,204,204,120,0,
p=112=0,0,248,204,204,248,192,192,
q=113=0,0,124,204,204,124,12,14,
r=114=0,0,108,112,96,96,96,0,
s=115=0,0,120,192,120,12,248,0,
t=116=0,48,120,48,48,48,28,0,
u=117=0,0,204,204,204,204,120,0,
v=118=0,0,204,204,120,120,48,0,
w=119=0,0,214,214,214,214,108,0,
x=120=0,0,204,120,48,120,204,0,
y=121=0,0,204,204,204,124,12,120,
z=122=0,0,252,24,48,96,252,0,
{=123=0,34,102,204,204,102,34,0,
!=124=8,124,198,198,198,198,124,0,
}=125=0,68,102,51,51,102,68,0,
~=126=0,54,108,0,0,0,0,0,
©=127=0,124,198,192,208,214,124,48,
```

Figura 7/1

Os 768 bytes correspondentes poderão ser colocados noutro endereço (o que possibilita a utilização simultânea dos dois abecedários). Sugerimos o endereço imediatamente abaixo do endereço 64500. Assim:

```
64500 - 768 = 63732
```

Por razões de segurança e para trabalharmos com números redondos, fixemos o novo endereço em 63700. Utilize o programa 3/1 para introduzir os decimais da fig. 7/1. Faça as seguintes alterações ao programa:

Linha 10 — LET endereco = 63700: FOR i = 33 TO 127
Linha 20 — eliminada
Linha 40 — BEEP .2,30: INPUT "Decimal? ";(f+1);" ";"do codigo ";(i);" ";n
Linha 50 — IF n < 0 OR n > 255 THEN GO TO 40
Linha 60 — POKE ((endereco +(CODE i - 32)* 8 + f),n
Linha 80 — IF i = 127 THEN PRINT "FIM DA INTRODUCAO": BEEP 1,20
Linhas 30, 70 e **90** sem alterações.

NOTA. — Para este caso, em que se vão modificar todos os caracteres dos códigos 33 a 127 (o codigo 32 é o «espaço), não é necessário, previamente, copiar o conjunto do endereço 15616 para o endereço 63700.

Uma vez introduzidos todos os decimais dos 95 caracteres, prepare a cassete onde já gravou o ABECEDÁRIO 1, deixe um espaço livre e grave os 768 bytes do endereço 63700, da forma habitual:

```
- SAVE "ABEC 2" CODE 63700,768
```

e verifique:

```
- VERIFY "" CODE
```

Controle os resultados com o programa 2/1 e determine os «bytes equivalentes» do endereço 63700 com o programa 4/1, os quais deverão ser:

```
- Byte menor = 212
- Byte maior = 247
```

Utilize de novo o programa 5/1, mas agora com as seguintes alterações, necessárias para o novo abecedário:

Linha 10 — PRINT AT 0,2; "UTILIZACAO DO ABECEDARIO 2"
Linha 20 — ....."ABECEDARIO 2"
Linha 40 — PAUSE 0: LOAD "" CODE 63700,768
Linha 60 — PRINT AT 0,11; "ABECEDÁRIO 2": PRINT
Linha 70 — (texto à sua vontade)
Linha 800 — POKE 23606,212: POKE 23607,247: RETURN: restantes linhas sem alteração

O leitor possui, agora, dois abecedários acentuados, um «normal», no endereço 64500 e outro do tipo «cheio», no endereço 63700. Sabe como utilizá-los separadamente. Falta só aprender a forma de in-

cluir e aplicar os dois abecedários num mesmo programa. Vejamos como:

PROGRAMA 6/1

```
  10 PRINT AT 0,0;"Prepare o gravador com a casset-te on
de tem gravados os dois     abecedarios"
  20 PRINT '"Prima ENTER quando estiver pre- parado e li
gue o gravador."
  30 PAUSE 0: LOAD ""CODE 63700,768: LOAD ""CODE 64500,7
68
  40 CLS : GO SUB 700
  50 PRINT AT 0,10;"ABECED`RIO 1": PRINT
  60 PRINT "Este $ o ABECED`RIO normal com  acentos"
  70 GO SUB 800: PRINT TAB 10;"ABECED`RIO 2": PRINT
  80 PRINT "Este $ o ABECED`RIO cheio com   acentos"
  90 PRINT : GO SUB 700
 100 PRINT '"Eis os caracteres normais:"
 110 PRINT : GO SUB 1000
 120 PRINT : GO SUB 800
 125 PRINT '"Eis os caracteres cheios:"
 130 PRINT : GO SUB 1000
 140 GO SUB 900: STOP
 700 POKE 23606,244: POKE 23607,250: RETURN
 800 POKE 23606,212: POKE 23607,247: RETURN
 900 POKE 23606,0: POKE 23607,60: RETURN
1000 FOR f=32 TO 127: PRINT CHR$ f;: NEXT f: RETURN
```

O programa não exige grandes explicações, salvo a linha 30, onde, como o leitor já se apercebeu, começamos por carregar (LOAD) o endereço mais baixo e logo de seguida o mais alto. Claro, como gravou primeiro o endereço 64500, terá de colocar a cassete no ponto onde gravou o endereço 63700 e depois voltar atrás. É para estas manobras que serve o contador de rotações do gravador: *aconselhamo-lo vivamente a equipar-se com tão útil acessório, se porventura não o possui.*

Analisemos outro processo de gravação e carregamento, para o caso de o leitor ter um programa de texto (ou onde aplica textos), utilizando os dois abecedários. O próprio programa incluirá instruções para gravação, quer da parte BASIC, quer dos bytes dos abecedários e instruções para «LOAD» com arranque automático. Vejamos as instruções:

```
9000 SAVE "programa" LINE 9100
9010 SAVE "ABEC 1/2" CODE 63700,1568
9020 VERIFY "": VERIFY "" CODE: STOP
9100 CLEAR 63699: LOAD "programa": LOAD "ABEC 1/2 CODE:
RUN
```


NOTA. — Estas linhas são de introduzir no final do seu programa, admitindo que não tem as linhas de 9000 em diante ocupadas. Se isso acontecer, altere os números destas linhas, como lhe convier.

Como funciona

Linha 9000 — instrução para gravar a parte BASIC, com passagem do comando à linha 9100.

Linha 9010: instrução para gravar os dois abecedários num mesmo «bloco» de 1568 bytes de comprimento. Com efeito: 64500 + 768 = 65268: 65268 — 63700 = 1568 (seriam só 2 × 768 = 1536, mas lembre-se que deixámos uma margem de segurança de 32 bytes.

Linha 9020 — instruções de verificação do BASIC e dos bytes.

Linha 9100 — CLEAR (endereço — 1) -> prepara a RAMTOP para receber os bytes do endereço 63700 : carregamento do BASIC : carregamento dos bytes dos dois abecedários : RUN -> arranque do programa na linha 1: *ATENÇÃO: a instrução RUN faz CLS e coloca a zero todas as variáveis, eliminando, também todos os registos eventualmente em memória. Se tem dúvidas sobre a conveniência da instrução RUN, use GO TO 1, ou GO TO «n», sendo «n» a linha em que o seu programa arranca.*

Deste modo, para gravar o seu programa BASIC em conjunto com os bytes dos abecedários, faça: GO TO 9000: aparece-lhe a mensagem *«start tape and press any key»*. Esteja atento, pois, logo que o BASIC esteja gravado, volta a aparecer a mesma mensagem e, sem parar o gravador, deve premir qualquer tecla.

Após a gravação dos 1568 bytes, o computador fica aguardando os VERIFY: rebobine a cassete e faça andar o gravador: a verificação é feita sem interrupção, primeiro do BASIC, depois dos bytes.

Para carregar o seu programa no computador, basta fazer LOAD "". Sem interrupção, é carregada a parte BASIC e, seguidamente, os bytes do endereço 63700. O programa arrancará automaticamente na linha respectiva.

*

Terminamos aqui o capítulo I e esperamos ter contribuído com uma série de conhecimentos que tornarão os seus programas mais atraentes e profissionalizados.

No capítulo seguinte, o leitor terá oportunidade de conhecer mais uma das inúmeras capacidades do seu SPECTRUM.

CAPÍTULO II

1 — INTRODUÇÃO

Certamente que o leitor já se encontrou na situação, algo desesperante, de precisar de mais gráficos, tendo esgotado os vinte e um UDG (gráficos definidos pelo utilizador) disponíveis no seu SPECTRUM. É natural isto acontecer, pois tanto o manual do aparelho como outra bibliografia mais avançada, pouco ou nada acrescentam ao assunto.

Na realidade, com os 768 bytes do conjunto de 96 caracteres estudados no capítulo I, o leitor pode criar *96 gráficos diferentes*, colocando-os num determinado endereço. E se precisar ainda de mais gráficos, basta criar outro conjunto, noutro endereço, à semelhança do que fizemos para os dois abecedários.

Vamos, então, ocupar-nos deste aliciante problema.

2 — GERADOR DE GRÁFICOS

O programa que se segue é um «gerador de gráficos» —> GERAGRAF, o qual permite escolher um qualquer endereço e criar séries de 96 gráficos, no lugar dos 96 caracteres normais.

Tem à sua escolha dois processos para a formação: através da introdução de decimais, ou «à vista», utilizando os cursores.

Eis o aspecto das diferentes imagens de écran:

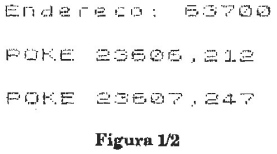


Figura 1/2

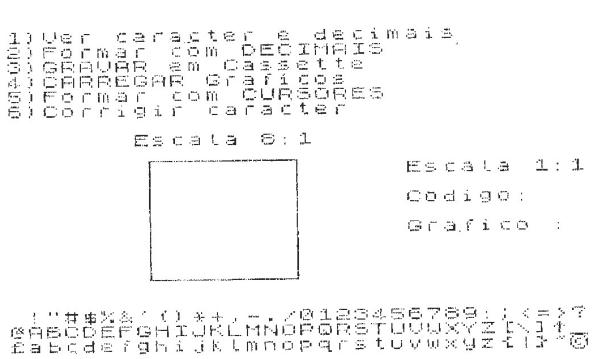


Figura 2/2

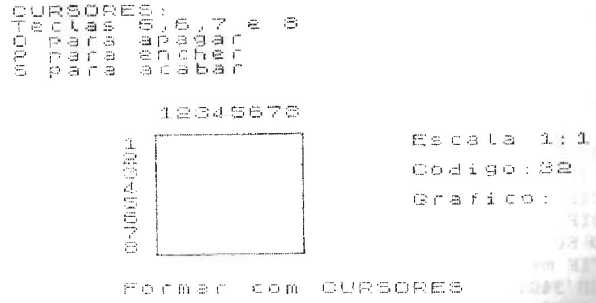


Figura 3/2

O programa 1/2 é um pouco longo, mas merece o trabalho de o estudar e introduzir, pois constituirá uma ferramenta de trabalho de extrema utilidade. Poderia ser mais simples e incluir menos instruções: perderia, certamente, qualidade e manejabilidade.

Por outro lado, é nossa preocupação habituar o leitor a dar uma boa apresentação aos seus programas e utilizar, sempre que possível, soluções elegantes.

Consideramos, ainda, que os avisos sonoros (BEEP) são de grande utilidade, chamando a atenção do utilizador para as intervenções pedidas pelo computador.

Eis a listagem do programa:

PROGRAMA 1/2

```
  5 REM COPYRIGHT R. CASQUILHO
 10 GO SUB 2600: CLS
 15 PRINT AT 11,7; FLASH 1;"AGUARDE UM MOMENTO": BEEP .3
,30
 20 FOR A=0 TO 767: POKE (n1+A),PEEK (15616+A): NEXT A
 25 GO SUB 2500: GO SUB 1500
 30 CLS : LET a=9: INK 0: PLOT 63,39: DRAW 65,0: DRAW 0,
65: DRAW -65,0: DRAW 0,-65: PRINT AT 19,0;: GO SUB 800:
FOR n=32 TO 127: PRINT CHR$ n;: NEXT n: GO SUB 900
 35 PRINT AT 7,7;"Escala 8:1";AT 9,22;"Escala 1:1";AT 11
,22;"Codigo:";AT 13,22;"Grafico :": BEEP .3,40
 40 PRINT AT 0,0;"1)Ver caracter e decimais"'"2)Formar
com "; BRIGHT 1;"DECIMAIS"; BRIGHT 0;'"3)GRAVAR em Casset
te"'"4)CARREGAR Graficos"'"5)Formar com "; BRIGHT 1;"CURS
ORES"; BRIGHT 0;'"6)Corrigir caracter": PRINT #0; BRIGHT
1;"Escolha o programa->prima 1 a 6 "
 45 LET m=CODE INKEY$-48: IF m<1 OR m>6 THEN  GO TO 45
 50 IF m=1 THEN  GO SUB 100: GO SUB 110: GO SUB 1500: GO
TO 30
 55 IF m=2 THEN  GO SUB 100: GO SUB 200: GO SUB 1500: GO
TO 30
 60 IF m=3 THEN  GO TO 2700
 65 IF m=4 THEN  GO TO 2800
 70 IF m=5 THEN  LET s$="Formar com CURSORES": GO SUB 10
0: GO SUB 300: GO SUB 1000: GO TO 500
 75 IF m=6 THEN  LET s$="Corrigir caracter": GO SUB 100:
GO SUB 340: GO SUB 1000: GO SUB 360: GO TO 500
100 BEEP .2,30: INPUT  BRIGHT 1;"Que caracter ? "; LINE
a$: IF CODE a$<32 OR CODE a$>127 THEN  GO TO 100
105 RETURN
110 LET a=9: FOR f=1 TO 8: LET c=PEEK ((n1+(CODE a$-32)*
8)+(f-1)): PRINT AT a,17; BRIGHT 1;c;AT a,6;f
115 GO SUB 2000: PRINT AT 11,29;CODE a$: GO SUB 800: PR
INT AT 13,31;CHR$ (CODE a$): GO SUB 900: RETURN
200 FOR f=0 TO 7
205 BEEP .2,30: INPUT "Decimal ? "; BRIGHT 1;(f+1); BRIG
HT 0;" "; LINE c$: IF c$="" THEN  GO TO 205
210 LET c=VAL c$: IF c<0 OR c>255 THEN  GO TO 205
215 POKE ((n1+(CODE a$-32)*8)+f),c
220 PRINT AT a,6; INK 2; BRIGHT 1;f+1;AT a,17; BRIGHT 1;
c: PRINT AT 11,29;CODE a$: GO SUB 800: PRINT AT 13,31;CH
R$ (CODE a$): GO SUB 900: GO SUB 2000: RETURN
```



```
300 DIM c(8): RETURN
340 DIM c(8): FOR q=1 TO 8: LET c(q)=PEEK ((n1+(CODE a$-
32)*8)+(q-1)): NEXT q
345 RETURN
360 LET a=9: LET b=10: FOR f=1 TO 8: LET c=PEEK ((n1+(CO
DE a$-32)*8)+(f-1)): INK 2: GO SUB 2000
365 RETURN
500 POKE 23658,0: INK 0: LET d=128: LET a=9: LET b=8:
505 LET b$=INKEY$
510 IF b$="5" AND b>8 THEN  LET b=b-1: LET d=d*2
515 IF b$="8" AND b<15 THEN  LET b=b+1: LET d=d/2
520 IF b$="6" AND a<16 THEN  LET a=a+1
525 IF b$="7" AND a>9 THEN  LET a=a-1
530 IF b$="p" AND ATTR (a,b)=56 THEN  PRINT AT a,b; INK
2;CHR$ 143: LET c(a-8)=c(a-8)+d
535 IF b$="o" AND ATTR (a,b)=58 THEN  PRINT AT a,b;" ":
LET c(a-8)=c(a-8)-d
540 POKE ((n1+(CODE a$-32)*8)+(a-9)),c(a-8): PRINT AT 11
,29;CODE a$: GO SUB 800: PRINT AT 13,31;CHR$ (CODE a$):
GO SUB 900
545 PRINT AT a,b; OVER 1; INK 8;"+": PAUSE 5: PRINT AT a
,b; OVER 1; INK 8;"+"
550 IF b$="s" THEN  CLS : GO TO 30
555 GO TO 505
800 POKE p1,b1: POKE p2,b2: RETURN
900 POKE p1,0: POKE p2,60: RETURN
1000 CLS : INK 2: PLOT 63,39: DRAW 65,0: DRAW 0,65: DRAW -
65,0: DRAW 0,-65: INK 0: FOR x=9 TO 16: PRINT AT x,6; BRI
GHT 1;x-8: NEXT x: PRINT AT 7,8; BRIGHT 1;"12345678"
1005 PRINT AT 0,0;"CURSORES:"'"Teclas "; BRIGHT 1;"5,6,7
e 8"; BRIGHT 0;'"O para apagar"'"P para encher"'"S para
acabar"
1010  PRINT  AT  9,22;"Escala  1:1";AT  11,22;"Codigo:";AT
13,22;"Grafico:";AT 20,(LEN s$-32)/2;s$: RETURN
1500 INPUT "": PRINT #0; BRIGHT 1;"          Prima uma
tecla          ": BEEP .3,30: PAUSE 0: RETURN
2000 IF c>127 THEN  PRINT AT a,8;CHR$ 143: LET c=c-128
2005 IF c>63 THEN  PRINT AT a,9;CHR$ 143: LET c=c-64
2010 IF c>31 THEN  PRINT AT a,10;CHR$ 143: LET c=c-32
2015 IF c>15 THEN  PRINT AT a,11;CHR$ 143: LET c=c-16
2020 IF c>7 THEN  PRINT AT a,12;CHR$ 143: LET c=c-8
2025 IF c>3 THEN  PRINT AT a,13;CHR$ 143: LET c=c-4
2030 IF c>1 THEN  PRINT AT a,14;CHR$ 143: LET c=c-2
2035 IF c=1 THEN  PRINT AT a,15;CHR$ 143: LET c=c-1
2040 LET a=a+1: NEXT f: RETURN
2500 LET p1=23606: LET p2=23607
```

```
2505 LET v=n1-256
2510 LET B1=v-256*INT (v/256)
2515 LET B2=INT (v/256)
2520 CLS : PRINT "Endereco: ";n1
2525 PRINT '"POKE ";p1;",";b1
2530 PRINT '"POKE ";p2;",";b2
2535 RETURN
2600 CLS : PRINT AT 0,6; BRIGHT 1;"GERADOR DE GRAFICOS"
2605 PRINT AT 9,0;"Indique o ENDERECO onde quer alojar os
GRAFICOS ou ENDERECO dos BYTES a carregar:": BEEP .2,30
2610 INPUT "ENDERECO ? ";n1: IF n1<30000 OR n1>64599 THEN
GO TO 540
2615 RETURN
2700 BEEP .3,30: INPUT "Nome>"; LINE f$: IF LEN f$>10 THE
N GO TO 0225
2705 SAVE f$CODE n1,768
2710 CLS : PRINT AT 11,11;"VERIFICAR": BEEP .3,30
2720 VERIFY f$CODE n1,768
2725 GO TO 30
2800 CLS : PRINT "Prepare a Cassette com os BYTES a car
regar e ligue o gravador": BEEP .3,30
2805 LOAD ""CODE n1
2810 GO TO 30
9000 CLEAR : SAVE "GERAGRAF" LINE 1
9010 VERIFY "GERAGRAF"
```

Como funciona

Linha 10 — Chama a sub–rotina da linha 2600, a qual nos pede a indicação do endereço onde queremos trabalhar. De notar que os valores admitidos se situam entre 30000 e 64599, isto, porque um endereço inferior a 30000 iria situar–se perto da área BASIC do próprio programa 1/2; por outro lado: 64599 + 768 = 65367, isto é, o limite inferior da área dos UDG, que não queremos perturbar. O «RETURN» da linha 2615 reconduz–nos à linha 15;

Linha 15 — Início do ciclo que copia os 96 caracteres, da ROM para o endereço escolhido;

Linha 25 — Chama a sub–rotina da linha 2500, já nossa conhecida, onde são determinados os «bytes equivalentes» do endereço escolhido. A mesma sub–rotina imprime os valores dos bytes «menor» e «maior» e o endereço, tal como é mostrado na figura 1/2. O «RETURN» da linha 2535 devolve o comando à linha 25. Chamada da sub–rotina da linha 1500, a qual imprime na base do écran a mensagem: «Prima uma tecla», após o que passa à linha 30;

Linhas 30/40 — Instruções para a formação da imagem do

«Menu principal», tal como é mostrado na figura 2/2; chamada a sub–rotina da linha 800, que coloca a variável CHARS no endereço escolhido; impressão do conjunto dos 96 caracteres; chamada da sub–rotina da linha 900, que coloca a variável CHARS nos valores normais; impressão das opções do Menu principal;

Linha 45 — Dá a «m» o valor do código da tecla premida menos 48: ou seja é, se a tecla premida for o 3 (código 51), «m» terá o valor do 51 — 48 = 3. *Tome nota desta técnica para comando de opções, que não só é elegante como fácilmente controlável.*

Linhas 50/75 — Instruções condicionais que, conforme o valor de «m», levam o comando às rotinas ou sub–rotinas relacionadas com a opção;

Linha 100 — Sub–rotina em que é pedida a entrada do carácter desejado, limitando-a aos códigos entre 32 e 127;

Linha 105 — Retorno à instrução de comando da opção 1;

Linha 110 — A variável «a» (valor da linha para a instrução PRINT AT a,n) toma o valor de 9; início do ciclo FOR–NEXT «f», em que a variável «c» toma o valor do conteúdo em decimal do endereço escolhido (n1); impressão em a,6 (9,6) do valor de «f»; (o valor de «a» é incrementado na sub–rotina da linha 2000);

Linha 115 — Chama a sub–rotina das linhas 2000/2040, a qual, dentro de uma grelha 8 × 8 e em função do conteúdo em binário dos 8 bytes do carácter seleccionado, imprime em «INK» o carácter 143; «a» é incrementado de uma unidade e o «NEXT f» fecha o ciclo iniciado na linha 110; o «RETURN» leva o comando à linha 50, com passagem à linha 1500 e regresso à linha 30;

Linhas 200/220 — OPÇÃO 2 (m = 2): Formar com DECIMAIS: esta opção permite criar um gráfico, através da introdução (POKE) de números decimais, nos endereços do carácter seleccionado (linha 215); a linha 220 imprime no écran o número de ordem do byte, o decimal e o código do carácter; chama a sub–rotina 800 e imprime o gráfico à escala 1:1; chama a sub–rotina 2000 e imprime o gráfico à escala 8:1; o «RETURN» leva o comando à linha 55, com passagem pela sub–rotina 1500 e regresso à linha 30;

Linha 300 — Sub–rotina destinada à criação de uma zona de memória para oito bytes, no quadro «DIM c»;

Linhas 340/345 — Sub–rotina que cria um quadro «DIM c», onde serão guardados os conteúdos em decimal do carácter escolhido (utilizada com a opção 6 ——> Corrigir carácter);

Linhas 360/365 — Sub–rotina utilizada na opção 6 e que imprime à escala 1:8 o carácter que queremos corrigir, através da sub–rotina 2000; o «RETURN» devolve o comando à linha 75, com passagem pela rotina das linhas 550/555;

Linhas 500/555 — Rotina para o comando do cursor (teclas 5, 6, 7 e 8) e impressão dos «bits 1» de cada um dos oito bytes do carácter; a linha 530, através da função «ATTR», imprime o carácter 143 quando

é premida a tecla «p»; a linha 535, por meio da mesma função, imprime um espaço (" ") quando é premida a tecla «o»; a linha 540 coloca o conteúdo em decimal no endereço respectivo, imprime o código do carácter e o gráfico à escala 1:1, conforme este vai sendo formado; a linha 545 imprime o sinal «+» de forma intermitente, através das funções «OVER 1» e «PAUSE 5»;

Linha 800 — Sub–rotina para colocação da variável CHARS no endereço escolhido;

Linha 900 — Sub–rotina para colocação da variável CHARS nos valores normais;

Linhas 1000/1010 — Sub–rotina utilizada pelas opções 5 e 6, para criação da imagem da figura 3/2; o «RETURN» devolve o comando às linhas 70 ou 75;

Linha 1500 — Sub–rotina destinada à impressão, na base do écran, da mensagem: «Prima uma tecla», acompanhada de um aviso sonoro; após se ter premido qualquer tecla, devolve o comando à instrução imediatamente a seguir àquela que a chamou;

Linhas 2000/2040 — Sub–rotina já explicada na linha 115;

Linhas 2500/2535 — Sub–rotina já explicada na linha 25;

Linhas 2600/2615 — Sub–rotina já explicada na linha 10;

Linhas 2700/2725 — Rotina destinada à gravação em cassete dos 768 bytes do conjunto de gráficos (96 caracteres) do endereço escolhido; é pedido o nome do conjunto, o qual é guardado na variável alfanumérica «f$»; se o nome tiver mais de dez caracteres (incluindo espaços), o comando é devolvido à linha 2700; o valor do endereço escolhido está contido na variável «n1»; após gravação, um aviso sonoro alertanos para a necessidade de verificação (VERIFY da linha 2710), após o que o programa regressa à linha 30;

Linhas 2800/2810 — Rotina destinada ao carregamento (LOAD) do conjunto de gráficos a analisar ou corrigir, localizado no endereço previamente indicado no início do programa e que fica guardado na variável «n1»; após carregamento, o programa regressa à linha 30;

Linhas 9000/9010 — Instruções destinadas à gravação do programa em cassete, com arranque automático na primeira linha após o LOAD ""; a linha 9010 faz a respectiva verificação.

Mais uma vez, o mesmo conselho: *antes de começar a introduzir o programa na máquina, estudeo cuidadosamente e esforce–se por compreender todas as instruções e a sua correlação.*

Vamos admitir que assim fez. Introduza agora o programa, prepare uma cassete limpa e grave-o com o comando directo:

```
– SAVE "GERAGRAF"
– VERIFY ""
```


Temos as nossas razões para lhe aconselhar este método: se, eventualmente, introduziu uma instrução errada, esse erro pode provocar o bloqueamento do programa (CRASH), perdendo–se todo o trabalho, pois a única solução será desligar o computador. Deste modo, com o programa previamente gravado em cassete sem «arranque automático», pode, sem receio, ensaiar a sua execução com RUN ou GO TO 1.

Se tudo estiver correcto, o écran apresentará a seguinte mensagem, com um aviso sonoro:

```
GERADOR DE GRAFICOS

Indique o ENDERECO onde quer
alojar os GRAFICOS ou ENDERECO
dos BYTES a carregar:

ENDERECO ?
```

Nesta altura, o computador aguarda o endereço a ser introduzido. Sugerimos ao leitor, como primeira experiência, a indicação do endereço 63700, o mesmo onde gravou os 768 bytes do «ABECEDÁRIO 2». Após introdução, o écran apresentará a seguinte mensagem, em «FLASH»:

```
AGUARDE UM MOMENTO
```

Esta espera corresponde ao tempo de processamento da cópia dos 768 bytes do endereço 15616 para o endereço 63700 (linha 20). No final, o écran apresentará o aspecto da figura 1/2, indicando os valores do endereço e dos bytes «menor» e «maior».

Deverá anotar estes valores, pois poderão vir a ser-lhe úteis, em ocasiões futuras.

Premindo uma tecla, o écran mostrará a imagem da figura 2/2, a qual é constituída pelo «Menu principal» que nos oferece 6 OPÇÕES, um quadrado com 65 × 65 «pixels» onde serão impressos os caractores ou gráficos à escala 8:1, uma zona intitulada Escala 1:1 onde serão impressos o código do carácter e o gráfico ou carácter, à escala 1:1. Finalmente, na base do écran, o conjunto dos 96 caracteres normais, dos códigos 32 a 127.

— EXPERIÊNCIA 1

Escolha a OPÇÃO 1 —> Ver carácter e decimais: prima a tecla 1 e ser–lhe-á perguntado qual o carácter a ver. Introduza o carácter da mesma forma como se se tratasse de uma linha de programa, isto é, premindo simplesmente a tecla respectiva (com ou sem CAPS SHIFT)

e, se for um sinal impresso a «vermelho» no teclado, usando SYMBOL SHIFT.

O carácter será impresso no interior do quadrado, à escala 8:1 e, lateralmente, à esquerda, o número de ordem do byte; à direita, o respectivo conteúdo em decimal. Na zona «Escala 1:1» aparecerá o código e o carácter em causa.

Sugerimos que introduza vários caracteres e compare os decimais obtidos com os valores dados na figura 4/1.

— EXPERIÊNCIA 2

Como segunda experiência, sugerimos a escolha da OPÇÃO 4 —> CARREGAR gráficos. Prepare a cassete onde gravou o «ABECEDÁRIO 2» e ligue o gravador. Lembre-se que a variável CHARS já está preparada para receber o endereço 63700, pois foi esse valor que indicou no início da execução.

O écran apresentará a imagem da figura 2/2, mas agora o conjunto de caracteres será o do «ABECEDÁRIO 2», como mostra a figura:

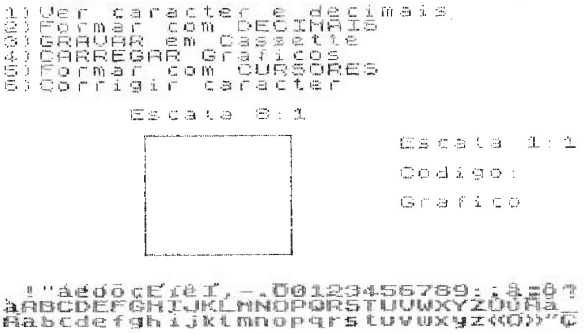


Figura 4/2

Aproveite, mais uma vez, para testar tanto o programa como a listagem da figura 7/1, à semelhança do que fez para os caracteres do abecedário normal.

— EXPERIÊNCIA 3

Vamos admitir, como aliás previmos no capítulo I, que o leitor pretende colocar letras acentuadas noutros caracteres, diferentes dos escolhidos. Pode fazê-lo por dois métodos:


```
— OPÇÃO 2 -> Formar com DECIMAIS
— OPÇÃO 6 -> Corrigir caracter
```

Sugerimos a utilização da OPÇÃO 2, pois é o método mais fácil e racional: premindo a tecla 2, o computador pergunta-nos qual o carácter a introduzir; imaginemos que o leitor precisa do carácter «$» onde está colocado o gráfico «é» (código 36). Por outro lado, pode prescindir do «Ú», que está colocado no carácter «[» (código 91):

— Introduza o carácter «$» ; em seguida os decimais correspondentes ao carácter «$» normal, retirados da listagem da figura 4/1:

```
$:36:=0,8,62,40,62,10,62,8
```

— À medida que introduzimos os decimais, o quadrado vai sendo preenchido com a forma do carácter e, lateralmente, à direita, são impressos os mesmos decimais e na zona da Escala 1:1 o código e o gráfico correspondente. No final, e após ter premido uma tecla, voltamos ao «Menu principal».

— Escolha de novo a OPÇÃO 2 para colocar o «é» no carácter «[»: introduza este carácter e, recorrendo à listagem da figura 7/1, introduza os decimais correspondentes ao «é»:

```
4,8,120,204,248,192,124,0
```

— Voltando ao «Menu principal», escolha a OPÇÃO 1 e veja o carácter «[»: aparecerá o «é» do «ABECEDÁRIO 2»; veja depois o carácter «$»: este aparecerá com o tipo normal da figura 4/1.

O leitor perguntará: «Mas como fazer para que o carácter '$' fique a 'cheio' para não destoar dos restantes?»

Vamos resolver o problema.

— EXPERIÊNCIA 4

Escolha a OPÇÃO 6 —> Corrigir caracter e introduza o carácter «$»: este aparecerá dentro do quadrado da figura 3/2, apresentando agora, na base, a indicação: «Corrigir caracter».

— Por meio dos «cursores» (teclas 5, 6, 7 e 8) e da tecla «p», encha o carácter de forma a dar-lhe o mesmo tipo «cheio» dos restantes. Pode sempre apagar com a tecla «o» e encher de novo com a tecla «p», tantas vezes quantas forem necessárias. Quando terminar, prima a tecla «s».

NOTA. — A linha 500 inicia-se com o comando «POKE 23658,0»: este comando garante a passagem automática para letras «minúsculas», únicas aceites por esta rotina. Se o comando fosse «POKE 23658,8», o computador ficaria automaticamente em letras «maiúsculas».

— Com o regresso ao «Menu principal», pode logo verificar se o seu novo carácter «$» condiz, ou não, com os restantes. Se não lhe agradar, volte à OPÇÃO 6 e corrija-o. Finalmente, peça para ver o carácter e tome nota dos novos decimais.

— EXPERIÊNCIA 5

Provavelmente o leitor aproveitou para introduzir outras alterações aos gráficos do «ABECEDÁRIO 2», quer colocando letras acentuadas noutros caracteres, quer modificando o próprio tipo das letras. Num caso ou noutro, ficou de posse de um conjunto de caracteres diferente do inicial, o qual é indispensavel preservar.

— Escolha a OPÇÃO 3 —> GRAVAR em cassete: prepare a cassete onde tem os abecedários (não destrua o «ABECEDÁRIO 2», pois pode vir ainda a ser necessário) ; indique um nome para o novo abecedário, grave e verifique.

O novo conjunto de caracteres ficará no mesmo endereço (63700) do «ABECEDÁRIO 2», o que não traz qualquer inconveniente, pois não é provável que o leitor vá utilizar os dois abecedários simultaneamente no mesmo programa.

— EXPERIÊNCIA 6

Mas admitamos que o leitor pretende colocar o seu novo abecedário noutro endereço: por exemplo, o endereço 62900 (63700 — 768 = 62932 —> arredondando para baixo = 62900).

— Tem os bytes gravados: faça BREAK ao programa e depois RUN; indique o endereço 62900 e tome nota dos respectivos bytes equivalentes; chame a OPÇÃO 4 e carregue os 768 bytes do novo abecedário; este ficará instalado no endereço 62900, como era seu desejo.

— CONCLUSÃO

Estamos convictos, se o leitor executou correctamente as seis experiências atrás indicadas, de que se apercebeu das potencialidades deste programa e deu por bem empregue o tempo de o estudar, introduzir e executar.

No parágrafo seguinte abordaremos mais uma aplicação prática do GERAGRAF: *a construção de «séries de gráficos» em diferentes endereços e a forma de os aplicar nos seus programas BASIC.*

3 — CONSTRUÇÃO DE GRÁFICOS

Tal como referimos no início deste capítulo, podemos construir tantos gráficos quantos quisermos: a única limitação reside na «memória» ocupada por cada conjunto de 96 caracteres (768 bytes). Mesmo admitindo a situação possível, mas pouco provável, de o leitor necessitar de quatro conjuntos de gráficos, teríamos:

```
4× 768 = 3072 bytes -> 3 Kbytes
```

o que lhe deixaria, ainda, cerca de 37 K de memória livre para qualquer programa BASIC, o que, em princípio, é suficiente.

Imaginando, por hipótese, que o seu programa BASIC ocupa cerca de 5 K, poderia construir e ter em memória nada menos que 47 conjuntos de gráficos, ou seja, 4512 gráficos diferentes. Vejamos como:

```
— Valor da RAMTOP = 65268 (65368 — 100 = 65268)
— 65268 — 23755 = 41513 (memória livre em bytes)
— Comprimento do programa BASIC = 5 K (5× 1024 = 5120 bytes)
— 41513 — 5120 = 36393 bytes (memória livre para gráficos)
— 36393 / 768 = 47,38 -> 47 conjuntos de 96 caracteres
— 47 × 96 = 4512 gráficos
```

Estes números são bastante elucidativos, embora correspondam a uma situação hipotética. Desçamos à terra e trabalhemos sobre um caso concreto que imaginámos para si: uma cabeça de «ROBOT», constituída por 46 gráficos. Eis a figura:

Figura 5/2

Comecemos por expor a técnica, que aconselhamos, para construção de imagens a partir de gráficos.

Muna-se de folhas de papel quadriculado, de formato A4, de preferência com 5 mm de quadrícula. Conforme o tamanho do desenho e,

consequentemente, o número de gráficos, assim terá de justapor duas ou mais folhas. As folhas deste tipo permitem executar 35 gráficos.

Trace a tinta quadrados com 4 × 4 cm (8 × 8 quadrículas); desenhe a lápis a figura que pretende, procurando, sempre que possível, fazer coincidir os traços exteriores com as linhas de divisão dos quadrados. Esta recomendação torna-se pertinente quando pretendemos obter uma imagem a várias cores, situação essa que abordaremos mais adiante.

Finalmente, encha os traços do desenho com a espessura que julgue necessária, tendo em conta que uma «fiada» de 8 pontos «pixels» dará a imagem de um traço, tal como se fizesse:

```
PLOT 95,79: DRAW 65,0
```

isto é, se marcasse o ponto de início do traço nas coordenadas $x = 95$ e $y = 79$ e desenhasse (DRAW) o traço com 65 pixels de comprimento, horizontalmente e para a direita a partir desse ponto.

Nada melhor que uma figura para concretizar. Voltemos à nossa cabeça do «ROBOT», mas agora desenhada sobre uma rede com 9 × 12 quadrados de 8 × 8:

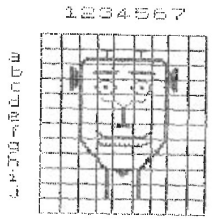


Figura 6/2

Como o leitor pode verificar, utilizámos duas folhas de papel quadriculado, embora a figura, para maior clareza, mostre mais uma «fiada» de quadrados em cada lado do rectângulo. Os gráficos encontram-se referenciados horizontalmente com as letras de «a» a «j» e, na vertical, com os números de «1» a «7».

Os traços que darão origem aos gráficos «a3» e «a5» tem a «espessura» de 1 pixel, enquanto que os traços dos gráficos «a2» ou «e2» tem 2 pixels.

Seguidamente, vamos numerar os gráficos e atribuir-lhes os caracteres onde vão ser colocados. Começaremos na linha «a» e na coluna «2» (o «a1» é um espaço em branco) e pelo carácter «!»: o carácter «espaço» (código 32) não deverá ser utilizado para alojar gráficos, pois é sempre necessário para os «quadrados em branco».

Vejamos a tabela da numeração e caracteres atribuídos:

N.º do gráfico	Coordenadas	Carácter	Código
01	a2	!	33
02	a3	"	34
03	a4	#	35
04	a5	$	36
05	a6	%	37
06	b1	&	38
07	b2	'	39
08	b3	(	40
09	b4	)	41
10	b5	*	42
11	b6	+	43
12	b7	,	44
13	c1	-	45
14	c2	.	46
15	c3	/	47
16	c4	0	48
17	c5	1	49
18	c6	2	50
19	c7	3	51
20	d2	4	52
21	d3	5	53
22	d4	6	54
23	d5	7	55
24	d6	8	56
25	e2	9	57
26	e4	:	58
27	e6	;	59
28	f2	<	60
29	f3	=	61
30	f4	>	62
31	f5	?	63
32	f6	@	64
33	g2	A	65
34	g3	B	66
35	g4	C	67
36	g5	D	68
37	g6	E	69
38	h2	F	70
39	h3	G	71
40	h5	H	72
41	h6	I	73
42	i3	J	74
43	i4	K	75
44	i5	L	76
45	j3	M	77
46	j5	N	78

A acção seguinte consiste na determinação dos «decimais» contidos em cada gráfico e em introduzi-los no programa GERAGRAF —> OPÇÃO 2. Pode também formar os gráficos com os cursores, escolhendo a OPÇÃO 5 e copiando para o écran o respectivo desenho. Sugerimos que escolha a primeira alternativa, pois esta facultar-lhe-á mais alguns preciosos conhecimentos.

Cada linha de pixels constitui um *número binário de 8 bits,* o qual tem o seu correspondente em «decimal». Lembramos ao leitor a explicação sobre o programa 1/1 e as figuras 1/1, 2/1 e 3/1 do capítulo I.

Deste modo, comecemos por determinar os oito números binários de cada gráfico, sabendo que o «INK» tem o valor de 1 e o «PAPER» tem o valor de 0.

Voltando ao nosso «ROBOT», vamos exemplificar com os gráficos «a3», «b1» e «c3». Vejamos as figuras:

GRÁFICO a3 ->

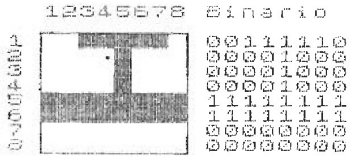


Figura 7/2

GRÁFICO b1 ->

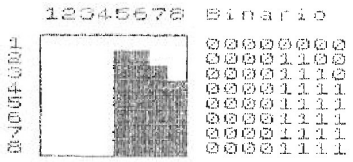


Figura 8/2

GRÁFICO c3 ->

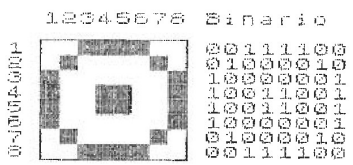


Figura 9/2

Pensamos que as figuras são suficientemente elucidativas, permitindo ao leitor determinar, sem mais explicações, os números binários de qualquer gráfico.

Por último, vamos determinar os correspondentes valores decimais de cada número binário: isto, porque a função «POKE» só aceita *valores decimais inteiros entre 0 e 255,* e tanto o nosso programa GERAGRAF como quaisquer instruções destinadas à construção de «gráficos definidos pelo utilizador» funcionam com base naquela função.

Naturalmente que o leitor pode utilizar a função «BIN» existente no seu SPECTRUM, para obter o correspondente decimal de um número binário: basta introduzir o comando directo:

```
PRINT BIN número binário
```

e a máquina indica imediatamente o decimal. Por exemplo, introduza o número binário do byte n.º 4 do gráfico «c3»:

```
PRINT BIN 10011001
```

e obterá o valor 153, que é o decimal correspondente.

No entanto, esta não nos parece a solução mais prática, sobretudo quando temos de trabalhar com 46 gráficos, ou seja, 368 números binários ($46 \times 8 = 368$).

Para estes casos, propomos a utilização do programa 2/2, intitulado «BINDEC», cuja listagem se segue e é destinado à conversão de números binários em números decimais. O programa guarda em memória: os números binários introduzidos; os decimais correspondentes; os caracteres onde pretendemos colocar os gráficos. No final da introdução, se o leitor possuir uma impressora, poderá obter uma listagem semelhante à da figura 10/2.

Eis a listagem do PROGRAMA 2/2 —> «BINDEC»:

```
  1 REM COPYRIGHT R.CASQUILHO
 10 CLS : PRINT AT 0,0;"CONVERSAO DE BINARIO EM DECIMA
L"'"    Especial para graficos"
 15 PRINT '''"Indique quantidade de graficos, respectivos
caracteres e numerosbinarios:"
 20 PRINT '''"Pode interromper quando quizer. Grave o pro
grama. Quando o car- regar de novo,este continuara nogra
fico seguinte."
 25 PRINT '''"Tome nota dos 8 decimais de cadagrafico, ou
copie para a impres-sora,ZX PRINTER ou semelhante."'''"
   PRIMA QUALQUER TECLA      "
 30 PAUSE 0: CLS : GO TO 500
 35 CLS : POKE 23609,10: LET x=0: LET y=0: LET y1=1: DIM
x$(32)
 40 BEEP .2,30: INPUT "Quantos graficos ? (max.96) ";
LINE n$: IF n$ "" THEN  GO TO 50
```

```
  45 FOR f=1 TO LEN n$: IF CODE n$(f)<49 OR CODE n$(f)>57
THEN  GO TO 40
  50 NEXT f
  55 LET n=VAL n$: IF n<1 OR n>96 THEN  GO TO 50
  60 DIM e$(n*8,8): DIM c$(n): DIM d(n*8): FOR f=1 TO n
  65 BEEP .2,30: INPUT "Caracter do grafico ? ";(f);" ";c
$(f): IF CODE c$(f)<32 OR CODE c$(f)>127 THEN  GO TO 65
   70 PRINT AT x,0; BRIGHT 1;f;"- Caracter:";c$(f);"
Codigo:";CODE c$(f)
  75 LET x=x+2
  80 BEEP .2,40: INPUT "No.Binario ? "; BRIGHT 1;(y1);"/
";(f); BRIGHT 0;"                    Se tudo 0 entre ~Z~
          Se tudo 1 entre ~U~ "; LINE a$: IF a$="" THEN
GO TO 80
  85 IF a$="Z" OR a$="z" THEN  LET a$="00000000"
  90 IF a$="U" OR a$="u" THEN  LET a$="11111111"
  95 IF LEN a$<>8 THEN  GO TO 80
 100 FOR k=1 TO 8: IF CODE a$(k)<48 OR CODE a$(k)>49 THEN
GO TO 80
 105 NEXT k
 110 LET e$(y+1)=a$: GO SUB 200
 120 PRINT "Bin : ";a$;" = Dec:";d(y+1);"   ": LET x=x+1:
LET y=y+1: LET y1=y1+1: GO TO 145
 125 IF x=20 THEN  GO TO 500
 135 IF f=n THEN  GO TO 500
 140 NEXT f
 145 IF y/8=INT (y/8) THEN  LET y1=1: GO TO 125
 150 GO TO 80
 200 LET s=0
 205 FOR c=8 TO 1 STEP -1
 210 LET d=(CODE a$(c)-48)
 215 LET s=s+d*2^(8-c)
 220 NEXT c
 225 LET d(y+1)=s: RETURN
 500 BEEP .2,30: INPUT "": PRINT #0; BRIGHT 1;"Prima: 1-
Iniciar 2-Continuar        3-Imprimir 4-Gravar 5-FIM   "
 505 LET cd=CODE INKEY$-48: IF cd<1 OR cd>5 THEN  GO TO
505
 510 IF cd=1 THEN  CLS : GO TO 35
 515 IF cd=2 AND f-1<n THEN  LET y1=1: LET x=0: FOR g=x
TO 20: PRINT AT g,0;x$: NEXT g: POKE 23609,20: NEXT f
 520 IF cd=2 AND f>=n THEN  GO TO 500
 525 IF cd=3 THEN  GO TO 800
 530 IF cd=4 THEN  CLS : SAVE "BINDEC" LINE 1: PRINT AT 1
1,5; FLASH 1;"REBOBINE E VERIFIQUE": VERIFY "": GO TO 10
 535 IF cd=5 THEN  GO TO 1e4
```



```
 790 RANDOMIZE USR 50000: PRINT #3;CHR$ 27+"E";: LET a=1
 800 LET r=6: LET b=1: FOR s=r TO r+5
 805 PRINT #3;" ";s;"-Caracter:";c$(s);" Codigo:";CODE c$
(s)
 810 PRINT #3;b;"-Bin : ";e$(a);" = dec:";d(a);" ";a: LET
a=a+1: LET b=b+1: GO TO 825
 815 IF s=r+5 THEN  PRINT a;" ";s+1: GO TO 500
 820 NEXT s
 825 IF (a-1)/8=INT ((a-1)/8) THEN  LET b=1: PRINT #3: GO
TO 815
 830 GO TO 810
9000 CLEAR : SAVE "BINDEC" LINE 1
9010 VERIFY "BINDEC": STOP
```

Como funciona

Linhas 10/25 — Página inicial, com instruções para o utilizador.

Linha 30 — O programa aguarda que seja premida qualquer tecla; limpa o écran e passa à linha 500, início da rotina para as OPÇÕES: chama-se a atenção para as seguintes particularidades: o texto é impresso nas linhas 22 e 23, através da instrução «PRINT #0»; a instrução «INPUT""» tem por finalidade apagar estas linhas sempre que o comando volte à linha 500; a linha 505 reúne as instruções que levam o comando de novo à linha 505, se não for premida qualquer tecla ou se premirmos uma tecla diferente de «1, 2, 3, 4 ou 5»: com efeito, o código do carácter «1» é 49 —> 49 — 48 = 1; o código do carácter «5» é 53 —> 53 — 48 = 5: se o valor contido em «cd» for menor que 1 ou maior que 5, o programa não passa da linha 505.

Linha 35 — Limpeza do écran; introdução do valor «10» na variável de sistema «PIP» que comanda a duração do clique da tecla premida: o valor 10 torna o som bastante mais audível: se o leitor nunca ensaiou este comando, experimente fazer «POKE 23609,30» ou «POKE 23609,50»; inicialização das variáveis «x», «y» e «y1»; criação de um «quadro para 32 caracteres»: destina-se a ser incluído num comando para apagamento de uma ou mais linhas, tal como é feito na linha 515: «FOR g = x TO 20: PRINT AT g,0;x$: NEXT g» —> apaga desde a linha 0 até à linha 20.

Linhas 40/55 — Pedido do quantitativo de gráficos: as instruções das linhas 45 e 50 impedem a entrada de caracteres cujos códigos sejam inferiores a 48 ou superiores a 57, isto é, só serão aceites algarismos de «0 a 9»: com efeito, o número introduzido fica guardado na variável «LINE n$»: o ciclo FOR-NEXT «f» cobre o comprimento (LEN) do número e a instrução condicional seguinte «varre» o conteúdo de «n$» e, se encontrar algum carácter cujo código não esteja dentro dos

limites estabelecidos (uma letra, por exemplo), o comando volta, de novo, à linha 40; atribuição à variável «n» do valor contido em «n$»; a instrução condicional seguinte impede a entrada de valores inferiores a 1 ou superiores a 96.

Linha 60 — Criação de «quadros dimensionados» onde ficarão guardados: os números binários, no quadro «e$(nx8,8)» — a dimensão do quadro será função do valor de «n», logo, se tivermos indicado 46 gráficos, o quadro ficará preparado para guardar *368 grupos de 8 caracteres, isto é, 368 números binários;* os caracteres, no quadro «c$(n)» — mais uma vez a dimensão do quadro será função do valor de «n»; os decimais resultantes da conversão dos números binários introduzidos, no quadro «d(nx8)» — se o valor de «n» for 46, o quadro ficará preparado para receber 368 números decimais; início do ciclo FOR–NEXT «f», o qual cobre os valores de 1 até «n».

Linhas 65/150 — Rotina que se inicia com o pedido de introdução do carácter (f); a instrução condicional seguinte impede a entrada de caracteres cujo código seja menor que 32 ou maior que 127; impressão da linha que reúne: o número do gráfico, o carácter indicado e o respectivo código; incremento do valor de «x» de duas unidades: esta variável comanda a zona de impressão da linha 70; pedido do número binário (y1) do gráfico (f): as indicações seguintes destinam–se a facilitar a introdução de números binários formados ou só por «zeros», ou só por «uns», ficando as linhas 85 e 90 encarregadas desta tarefa; a linha 95 impede a introdução de um número de caracteres diferente de «8»: se o comprimento da cadeia (LEN a$) for diferente de 8, regressa–se à linha 80; as linhas 100/105 impedem a introdução de caracteres cujos códigos sejam diferentes de 48 ou 49, isto é, diferentes de «zeros» ou «uns»; a linha 110 guarda na posição (y+1) do quadro «e$» o conteúdo de «a$», isto é, o número binário acabado de introduzir; passagem à sub–rotina 200/225, onde é calculado o valor do número decimal correspondente ao número binário introduzido: a linha 225 guarda esse valor na posição (y+1) do quadro «d» e o comando retorna à linha 120 que manda imprimir no écran: o número de ordem do número binário, o número binário e o número decimal; «x», «y» e «y1» são incrementados de uma unidade; o comando passa à linha 145, onde é testado se o valor de «y» é divisível por 8, isto é, se já foi introduzido o oitavo número binário do gráfico: na afirmativa, passa–se à linha 125, onde é verificado se «x» já atingiu o valor «20», isto é, se o écran já se encontra «cheio» (a impressão atingiu a linha 20): na afirmativa é chamada a rotina das OPÇÕES; na negativa, a linha 135 vê se o valor de «f» é igual ao valor de «n», isto é, se já introduzimos o último gráfico: na afirmativa é chamada a rotina das OPÇÕES; na negativa a linha 140 leva–nos ao próximo valor de «f», isto é, ao gráfico seguinte (linha 65). Voltando à linha 145: se o valor de «y» não for divisível por 8, isto significa que ainda não introduzimos os 8 números binários do gráfico em curso: o comando passa novamente à linha 80 (através da linha 150), com pedido do número binário seguinte.

NOTA. — Sugerimos ao leitor, para melhor compreensão desta rotina, a execução de um exercício de simulação (no papel), que consistirá em partir dos valores iniciais das diferentes variáveis, sujeitá–las aos sucessivos incrementos das linhas 75 e 120 e aplicar os valores obtidos às diferentes instruções: não esqueça o incremento do ciclo FOR–NEXT «f», bem como o retorno de «y1» ao valor de 1 na linha 145 e a inicialização de «x» e «y1» na linha 515.

Linhas 200/225 — Sub–rotina onde é calculado o correspondente decimal do número binário contido em «a$» e que é chamada cada vez que «y» é incrementado: para uma melhor compreensão desta sub–rotina, sugerimos a utilização do próprio computador, numa simulação do desenvolvimento do ciclo FOR–NEXT «c», para um qualquer número binário, por exemplo: a$=«10101010».

Vejamos os cálculos:

```
Primeiro ciclo
s = 0
c = 8
d = (48 - 48) = 0
s= 0 + 0×2 ^ (8 - 8) = 0

Segundo ciclo
s = 0
c = 7
d = (49 - 48) = 1
s = 0 + 1×2 ^ (8 - 7) = 2

Terceiro ciclo
s = 2
c = 6
d = (48 - 48) = 0
s = 2 + 0×2 ^ (8 - 6) = 2

Quarto ciclo
s = 2
c = 5
d = (49 - 48) = 1
s = 2 + 1×2 ^ (8-5) = 10

Quinto ciclo
s = 10
c = 4
d = (48 - 48) = 0
s = 10 + 0×2 (8-4) = 10

Sétimo ciclo
s = 42
c = 2
d = (48 - 48) = 0
s = 42 + 0×2 (8 - 2) = 42

Sexto ciclo
s = 10
c = 3
d = (49 - 48) = 1
s = 10 + 1×2 ^ (8 - 3) = 42

Oitavo ciclo
s = 42
c = 1
d = (49 - 48) = 1
s = 42 + 1×2 ^ (8 - 1) = 170
```

donde, a conversão em decimal do número binário «10101010» corresponde a «170», o que pode ser comprovado utilizando a função «BIN».

Linhas 500/505 — Já referidas na linha 30.

Linha 510 — Premida a tecla «1», o écran é limpo e o comando passa à linha 35, com início das entradas.

Linha 515 — Premida a tecla «2», e se o valor de «f-1» for menor que «n», isto é, se ainda não tivermos introduzido todos os gráficos indicados no «INPUT» da linha 40, os valores de «y1» e «x» são inicializados, o écran é limpo da linha 0 à linha 20, o clique do teclado torna-se mais audível e o ciclo é reiniciado com o novo valor de «f».

Linha 520 — Premida a tecla «2» e se o valor de «f» for maior ou igual a «n», isto é, se já tivermos introduzido todos os gráficos, o comando passa à linha 500, com a apresentação das OPÇÕES.

Linha 525 — Premida a tecla «3», o comando passa à rotina da linhas 800/830, destinada à listagem dos conteúdos dos quadros dimensionados, na impressora: repare-se na utilização do comando «PRINT #3», em lugar do comando «LPRINT»: têm ambos o mesmo resultado, mas nós preferimos o primeiro, pois basta substituir o «#3» por «#2» para que a listagem se faça no écran. Experimente o leitor. Esta rotina tem instruções semelhantes às das linhas 70 e 120, pelo que dispensam mais explicações.

Linha 530 — Premida a tecla «4», o écran é limpo e aparece a mensagem para gravação, seguida da mensagem para verificação: de notar que o arranque automático é feito para a primeira linha (LINE 1) e não com «RUN», que destruiria todos os dados em memória, bem como o conteúdo das variáveis. *Assim, se decidir interromper a introdução dos gráficos, por exemplo no número 20, ao carregar de novo o programa, este arranca na primeira linha funcional (linha 10), seguindo-se a passagem às OPÇÕES: premindo a tecla «2», o programa passa imediatamente à linha 65, pedindo a introdução do carácter para o gráfico número 21.*

Linha 535 — Premindo a tecla «5», o programa pára. Para novo arranque, use «GO TO 1» ou «GO TO 500».

Linhas 800/830 — Sub-rotina já referida na linha 525.

Linhas 9000/9010 — Instruções para gravação e verificação do programa, com «limpeza» (CLEAR) automática de todos os dados em memória e anulação do conteúdo das variáveis. O leitor só deverá utilizar esta rotina quando pretender copiar o programa (sem dados) para uma segunda cassete de segurança.

Vamos fornecer ao leitor a tabela completa dos 46 gráficos da cabeça do «ROBOT». A tabela reúne os dados de todos os gráficos, pelo que poderia trabalhar de imediato com o GERAGRAF, escolhendo um endereço e formando com decimais.

No entanto, o nosso conselho é que utilize primeiro o BINDEC; se não possuir impressora, copie para um papel os dados necessários: o número do gráfico, o respectivo carácter e os decimais obtidos. Terá assim oportunidade de praticar com um programa que, esperamos, virá a utilizar frequentemente.

TABELA DOS GRÁFICOS DA CABEÇA DO ROBOT

```
 1-Caracter:! Codigo:33
1-Bin : 00000000 = dec:0
2-Bin : 00000000 = dec:0
3-Bin : 00000000 = dec:0
4-Bin : 00000000 = dec:0
5-Bin : 00011111 = dec:31
6-Bin : 00111111 = dec:63
7-Bin : 01100000 = dec:96
8-Bin : 11000000 = dec:192

 2-Caracter:" Codigo:34
1-Bin : 01111110 = dec:62
2-Bin : 00001000 = dec:8
3-Bin : 00001000 = dec:8
4-Bin : 00001000 = dec:8
5-Bin : 11111111 = dec:255
6-Bin : 11111111 = dec:255
7-Bin : 00000000 = dec:0
8-Bin : 00000000 = dec:0

 3-Caracter:# Codigo:35
1-Bin : 00000000 = dec:0
2-Bin : 00000000 = dec:0
3-Bin : 00000000 = dec:0
4-Bin : 00000000 = dec:0
5-Bin : 11111111 = dec:255
6-Bin : 11111111 = dec:255
7-Bin : 00000000 = dec:0
8-Bin : 00000000 = dec:0

 4-Caracter:$ Codigo:36
1-Bin : 01111100 = dec:124
2-Bin : 00010000 = dec:16
3-Bin : 00010000 = dec:16
4-Bin : 00010000 = dec:16
5-Bin : 11111111 = dec:255
6-Bin : 11111111 = dec:255
7-Bin : 00000000 = dec:0
8-Bin : 00000000 = dec:0

 5-Caracter:% Codigo:37
1-Bin : 00000000 = dec:0
2-Bin : 00000000 = dec:0
3-Bin : 00000000 = dec:0
4-Bin : 00000000 = dec:0

 24-Caracter:8 Codigo:56
1-Bin : 00000011 = dec:3
2-Bin : 00000011 = dec:3
3-Bin : 00000011 = dec:3
4-Bin : 00000011 = dec:3
5-Bin : 00000011 = dec:3
6-Bin : 00000011 = dec:3
7-Bin : 00000011 = dec:3
8-Bin : 00000011 = dec:3

 25-Caracter:9 Codigo:57
1-Bin : 11000000 = dec:192
2-Bin : 11000000 = dec:192
3-Bin : 11000000 = dec:192
4-Bin : 11000000 = dec:192
5-Bin : 11000000 = dec:192
6-Bin : 11000000 = dec:192
7-Bin : 11000000 = dec:192
8-Bin : 11000000 = dec:192

 26-Caracter:: Codigo:58
1-Bin : 00011000 = dec:24
2-Bin : 00011000 = dec:24
3-Bin : 00011000 = dec:24
4-Bin : 00011000 = dec:24
5-Bin : 00011000 = dec:24
6-Bin : 00011000 = dec:24
7-Bin : 00011000 = dec:24
8-Bin : 00111100 = dec:60

 27-Caracter:; Codigo:59
1-Bin : 00000011 = dec:3
2-Bin : 00000011 = dec:3
3-Bin : 00000011 = dec:3
4-Bin : 00000011 = dec:3
5-Bin : 00000011 = dec:3
6-Bin : 00000011 = dec:3
7-Bin : 00000011 = dec:3
8-Bin : 00000011 = dec:3

 28-Caracter:< Codigo:60
1-Bin : 11000000 = dec:192
2-Bin : 11000000 = dec:192
3-Bin : 11000000 = dec:192
4-Bin : 11000000 = dec:192
```

```
5-Bin : 11111000 = dec:248
6-Bin : 11111100 = dec:252
7-Bin : 00000110 = dec:6
8-Bin : 00000011 = dec:3

 6-Caracter:& Codigo:38
1-Bin : 00000000 = dec:0
2-Bin : 00001100 = dec:12
3-Bin : 00001110 = dec:14
4-Bin : 00001111 = dec:15
5-Bin : 00001111 = dec:15
6-Bin : 00001111 = dec:15
7-Bin : 00001111 = dec:15
8-Bin : 00001111 = dec:15

 7-Caracter:' Codigo:39
1-Bin : 11000000 = dec:192
2-Bin : 11000000 = dec:192
3-Bin : 11000000 = dec:192
4-Bin : 11000001 = dec:193
5-Bin : 11000011 = dec:195
6-Bin : 11000000 = dec:192
7-Bin : 11000000 = dec:192
8-Bin : 11000000 = dec:192

 8-Caracter:( Codigo:40
1-Bin : 00000000 = dec:0
2-Bin : 00000000 = dec:0
3-Bin : 00000000 = dec:0
4-Bin : 11111111 = dec:255
5-Bin : 11111111 = dec:255
6-Bin : 00000000 = dec:0
7-Bin : 00000000 = dec:0
8-Bin : 00000000 = dec:0

 9-Caracter:) Codigo:41
1-Bin : 00000000 = dec:0
2-Bin : 00000000 = dec:0
3-Bin : 00000000 = dec:0
4-Bin : 11000011 = dec:195
5-Bin : 10000001 = dec:129
6-Bin : 00000000 = dec:0
7-Bin : 00000000 = dec:0
8-Bin : 00000000 = dec:0

 10-Caracter:* Codigo:42
1-Bin : 00000000 = dec:0
2-Bin : 00000000 = dec:0
3-Bin : 00000000 = dec:0
4-Bin : 11111111 = dec:255

5-Bin : 11000000 = dec:192
6-Bin : 11000000 = dec:192
7-Bin : 11000000 = dec:192
8-Bin : 11000000 = dec:192

 29-Caracter:= Codigo:61
1-Bin : 00000000 = dec:0
2-Bin : 00000000 = dec:0
3-Bin : 00000000 = dec:0
4-Bin : 00000000 = dec:0
5-Bin : 11000000 = dec:192
6-Bin : 01100000 = dec:96
7-Bin : 00111111 = dec:63
8-Bin : 00010101 = dec:21

 30-Caracter:> Codigo:62
1-Bin : 01111110 = dec:126
2-Bin : 11111111 = dec:255
3-Bin : 00000000 = dec:0
4-Bin : 00000000 = dec:0
5-Bin : 00000000 = dec:0
6-Bin : 00000000 = dec:0
7-Bin : 11111111 = dec:255
8-Bin : 01010101 = dec:85

 31-Caracter:? Codigo:63
1-Bin : 00000000 = dec:0
2-Bin : 00000000 = dec:0
3-Bin : 00000000 = dec:0
4-Bin : 00000000 = dec:0
5-Bin : 00000011 = dec:3
6-Bin : 00000110 = dec:6
7-Bin : 11111100 = dec:252
8-Bin : 01010000 = dec:80

 32-Caracter:@ Codigo:64
1-Bin : 00000011 = dec:3
2-Bin : 00000011 = dec:3
3-Bin : 00000011 = dec:3
4-Bin : 00000011 = dec:3
5-Bin : 00000011 = dec:3
6-Bin : 00000011 = dec:3
7-Bin : 00000011 = dec:3
8-Bin : 00000011 = dec:3

 33-Caracter:A Codigo:65
1-Bin : 11000000 = dec:192
2-Bin : 01100000 = dec:96
3-Bin : 00110000 = dec:48
4-Bin : 00110000 = dec:48
```

```
5-Bin : 11111111 = dec:255
6-Bin : 00000000 = dec:0
7-Bin : 00000000 = dec:0
8-Bin : 00000000 = dec:0

 11-Caracter:+ Codigo:43
1-Bin : 00000011 = dec:3

2-Bin : 00000011 = dec:3
3-Bin : 00000011 = dec:3
4-Bin : 10000011 = dec:131
5-Bin : 11000011 = dec:195
6-Bin : 00000011 = dec:3
7-Bin : 00000011 = dec:3
8-Bin : 00000011 = dec:3

 12-Caracter:, Codigo:44
1-Bin : 00000000 = dec:0
2-Bin : 00110000 = dec:48
3-Bin : 01110000 = dec:112
4-Bin : 11110000 = dec:240
5-Bin : 11110000 = dec:240
6-Bin : 11110000 = dec:240
7-Bin : 11110000 = dec:240
8-Bin : 11110000 = dec:240

 13-Caracter:- Codigo:45
1-Bin : 00001111 = dec:15
2-Bin : 00001111 = dec:15
3-Bin : 00001111 = dec:15
4-Bin : 00001111 = dec:15
5-Bin : 00001110 = dec:14
6-Bin : 00001100 = dec:12
7-Bin : 00000000 = dec:0
8-Bin : 00000000 = dec:0

 14-Caracter:. Codigo:46
1-Bin : 11000000 = dec:192
2-Bin : 11000000 = dec:192
3-Bin : 11000000 = dec:192
4-Bin : 11000001 = dec:193
5-Bin : 11000001 = dec:193
6-Bin : 11000000 = dec:192
7-Bin : 11000000 = dec:192
8-Bin : 11000000 = dec:192

 15-Caracter:/ Codigo:47
1-Bin : 00111100 = dec:60
2-Bin : 01000010 = dec:66
3-Bin : 10000001 = dec:129
4-Bin : 10011001 = dec:153

5-Bin : 00011000 = dec:24
6-Bin : 00001100 = dec:12
7-Bin : 00001100 = dec:12
8-Bin : 00000110 = dec:6

 34-Caracter:B Codigo:66
1-Bin : 00000000 = dec:0

2-Bin : 11000000 = dec:192
3-Bin : 01101010 = dec:106
4-Bin : 00111111 = dec:63
5-Bin : 00000000 = dec:0
6-Bin : 00000000 = dec:0
7-Bin : 00000000 = dec:0
8-Bin : 00000000 = dec:0

 35-Caracter:C Codigo:67
1-Bin : 00000000 = dec:0
2-Bin : 00000000 = dec:0
3-Bin : 10101010 = dec:170
4-Bin : 11111111 = dec:255
5-Bin : 00000000 = dec:0
6-Bin : 00000000 = dec:0
7-Bin : 00000000 = dec:0
8-Bin : 00000000 = dec:0

 36-Caracter:D Codigo:68
1-Bin : 00000000 = dec:0
2-Bin : 00000011 = dec:3
3-Bin : 10101110 = dec:174
4-Bin : 11111000 = dec:248
5-Bin : 00000000 = dec:0
6-Bin : 00000000 = dec:0
7-Bin : 00000000 = dec:0
8-Bin : 00000000 = dec:0

 37-Caracter:E Codigo:69
1-Bin : 00000011 = dec:3
2-Bin : 00000110 = dec:6
3-Bin : 00001100 = dec:12
4-Bin : 00001100 = dec:12
5-Bin : 00011000 = dec:24
6-Bin : 00110000 = dec:48
7-Bin : 00110000 = dec:48
8-Bin : 01100000 = dec:96

 38-Caracter:F Codigo:70
1-Bin : 00000011 = dec:3
2-Bin : 00000001 = dec:1
3-Bin : 00000000 = dec:0
4-Bin : 00000000 = dec:0
```

```
5-Bin : 10011001 = dec:153
6-Bin : 10000001 = dec:129
7-Bin : 01000010 = dec:66
8-Bin : 00111100 = dec:60

 16-Caracter:0 Codigo:48
1-Bin : 00000000 = dec:0
2-Bin : 00000000 = dec:0
3-Bin : 00000000 = dec:0
4-Bin : 10000001 = dec:129
5-Bin : 10000001 = dec:129
6-Bin : 00000000 = dec:0
7-Bin : 00000000 = dec:0
8-Bin : 00000000 = dec:0

 17-Caracter:1 Codigo:49
1-Bin : 00111100 = dec:60
2-Bin : 01000010 = dec:66
3-Bin : 10000001 = dec:129
4-Bin : 10011001 = dec:153
5-Bin : 10011001 = dec:153
6-Bin : 10000001 = dec:129
7-Bin : 01000010 = dec:66
8-Bin : 11000011 = dec:60

 18-Caracter:2 Codigo:50
1-Bin : 00000011 = dec:3
2-Bin : 00000011 = dec:3
3-Bin : 00000011 = dec:3
4-Bin : 10000011 = dec:131
5-Bin : 10000011 = dec:131
6-Bin : 00000000 = dec:3
7-Bin : 00000000 = dec:3
8-Bin : 00000000 = dec:3

 19-Caracter:3 Codigo:51
1-Bin : 11110000 = dec:240
2-Bin : 11110000 = dec:240
3-Bin : 11110000 = dec:240
4-Bin : 11110000 = dec:240
5-Bin : 01110000 = dec:112
6-Bin : 00110000 = dec:48
7-Bin : 00000000 = dec:0
8-Bin : 00000000 = dec:0

 20-Caracter:4 Codigo:52
1-Bin : 11000000 = dec:192
2-Bin : 11000000 = dec:192
3-Bin : 11000000 = dec:192
4-Bin : 11000000 = dec:192
```

```
5-Bin : 00000000 = dec:0
6-Bin : 00000000 = dec:0
7-Bin : 00000000 = dec:0
8-Bin : 00000000 = dec:0

 39-Caracter:G Codigo:71
1-Bin : 00000000 = dec:0
2-Bin : 10000000 = dec:128
3-Bin : 11000000 = dec:192
4-Bin : 11000000 = dec:192
5-Bin : 01100000 = dec:96
6-Bin : 00110000 = dec:48
7-Bin : 00111000 = dec:56
8-Bin : 00011100 = dec:28

 40-Caracter:H Codigo:72
1-Bin : 00000000 = dec:0
2-Bin : 00000001 = dec:1
3-Bin : 00000011 = dec:3
4-Bin : 00000011 = dec:3
5-Bin : 00000110 = dec:6
6-Bin : 00001100 = dec:12
7-Bin : 00011100 = dec:28
8-Bin : 00111000 = dec:56

 41-Caracter:I Codigo:73
1-Bin : 11000000 = dec:192
2-Bin : 10000000 = dec:128
3-Bin : 00000000 = dec:0
4-Bin : 00000000 = dec:0
5-Bin : 00000000 = dec:0
6-Bin : 00000000 = dec:0
7-Bin : 00000000 = dec:0
8-Bin : 00000000 = dec:0

 42-Caracter:J Codigo:74
1-Bin : 00001100 = dec:12
2-Bin : 00001100 = dec:12
3-Bin : 00001100 = dec:12
4-Bin : 00001100 = dec:12
5-Bin : 00001100 = dec:12
6-Bin : 00001100 = dec:12
7-Bin : 00001100 = dec:12
8-Bin : 00001100 = dec:12

 43-Caracter:K Codigo:75
1-Bin : 11111111 = dec:255
2-Bin : 01111110 = dec:126
3-Bin : 00111100 = dec:60
4-Bin : 00011000 = dec:24
```



```
5-Bin : 11000000 = dec:192
6-Bin : 11000000 = dec:192
7-Bin : 11000000 = dec:192
8-Bin : 11000000 = dec:192

 21-Caracter:5 Codigo:53
1-Bin : 00000000 = dec:0
2-Bin : 00000000 = dec:0
3-Bin : 00100000 = dec:32
4-Bin : 00011111 = dec:31
5-Bin : 00000000 = dec:0
6-Bin : 00000000 = dec:0
7-Bin : 00000000 = dec:0
8-Bin : 00000000 = dec:0

 22-Caracter:6 Codigo:54
1-Bin : 00000000 = dec:0
2-Bin : 00000000 = dec:0
3-Bin : 00000000 = dec:0
4-Bin : 10000001 = dec:129
5-Bin : 01000010 = dec:66
6-Bin : 00100100 = dec:36
7-Bin : 00011000 = dec:24
8-Bin : 00011000 = dec:24
```

```
5-Bin : 00000000 = dec:0
6-Bin : 00000000 = dec:0
7-Bin : 00000000 = dec:0
8-Bin : 00000000 = dec:0

 44-Caracter:L Codigo:76
1-Bin : 00110000 = dec:48
2-Bin : 00110000 = dec:48
3-Bin : 00110000 = dec:48
4-Bin : 00110000 = dec:48
5-Bin : 00110000 = dec:48
6-Bin : 00110000 = dec:48
7-Bin : 00110000 = dec:48
8-Bin : 00110000 = dec:48

 45-Caracter:M Codigo:77
1-Bin : 00001100 = dec:12
2-Bin : 00001100 = dec:12
3-Bin : 00001100 = dec:12
4-Bin : 00001100 = dec:12
5-Bin : 00001100 = dec:12
6-Bin : 00001100 = dec:12
7-Bin : 00001100 = dec:12
8-Bin : 00001100 = dec:12
```

Obtida a *sua tabela,* compare-a com a nossa: se não existirem diferenças, isso significa que conferiu ao seu trabalho o carácter de seriedade que ele merece. Se os erros não forem superiores a 20%, podemos considerar o resultado como aceitável: recorra à tabela e faça as necessárias correcções. Na negativa, deixamos ao seu bom critério a decisão a tomar.

Em qualquer dos casos, a fase seguinte consiste na introdução dos decimais nos caracteres escolhidos para cada gráfico: carregue o GERAGRAF no computador e indique o endereço 62100, isto porque admitimos que o leitor pretenda conservar o seu abecedário no endereço 62900 (62900 — 768 = 62132 —> arredondando = 62100), não esquecendo de anotar o valor dos «bytes equivalentes» deste endereço. No final, prepare a cassette e, com as habituais precauções, grave os 768 bytes dos gráficos «ROBOT».

Por último, vamos preparar um pequeno programa que lhe permitirá aplicar os gráficos, obtendo no écran uma imagem idêntica à da figura 5/2. Vejamos o programa:

PROGRAMA 3/2

```
  10 GOSUB 800
  20 PRINT AT 6,13;"!""#$%";AT 7,12;"&()+,";AT 8,12;"-./0
123"
  30 PRINT AT 9,13;"45678";AT 10,13;"9: ;"AT 11,13;"<=>?
«;AT 12, 13;"ABCDE"
  40 PRINT AT 13,13;"FG HI";AT 14,14;"JKL";AT 15,14;"M
N"
  50 GOSUB 900
  60 PRINT AT 19,7;"A CABECA DO ROBOT"
  70 STOP
 800 POKE 23606,148: POKE 23607,241: RETURN
 900 POKE 23606,0 z POKE 23607,60: RETURN
9000 SAVE «ROBOT" LINE 9100
9010 SAVE «ROBOT" CODE 62100,768
9020 VERIFY"": VERIFY"" CODE: STOP
9100 CLEAR 62099: LOAD"" CODE: RUN
```

O funcionamento do programa não carece de longas explicações. Limitar-nos-emos a esclarecer os seguintes aspectos:

Linha 10 — A chamada à sub-rotina 800 prepara a variável de sistema «CHARS» para o endereço 62100.

Linhas 20/40 — Os gráficos podem ser tratados como qualquer carácter, por meio das instruções «PRINT AT»: para que o nosso desenho seja impresso sensivelmente a meio do écran, vamos recorrer à figura 6/2 e imaginar que a grelha de 9 × 12 se insere na grelha de 21 × 32 do écran. Nestas circunstâncias, o gráfico «a2» será impresso na linha 6 e na coluna 13 (AT 6,13); o gráfico «a3» na mesma linha e na coluna 14 (AT 6,14): de notar que este gráfico foi colocado no carácter " (aspas), pelo que é necessário incluir mais umas «aspas» depois do carácter. A impressão dos restantes gráficos obedece à mesma regra e o leitor notará que aproveitámos o facto de existirem vários gráficos na mesma linha, para pouparmos instruções.

Linha 9100 — A instrução «CLEAR 62099» (endereço — 1) prepara a RAMTOP para receber o endereço 62100: esta instrução seria dispensável neste programa, mas a sua inclusão permite fazer «NEW» sem que os bytes dos gráficos desapareçam da memória. Depois de ter gravado o programa e os gráficos, experimente o leitor o comando «NEW»: obviamente que o programa BASIC desaparecerá. Por comando directo, faça: «PRINT PEEK 62112». Será impresso o número 31, que é o valor decimal do quinto byte do gráfico «a2»; experimente «PRINT PEEK 62116»: obterá 62, que é o decimal do primeiro byte do gráfico «a3»: deste modo se prova a permanência dos bytes em memória.

Para saber o endereço de qualquer byte de um gráfico, use a fórmula:

```
endereço = ((endereço inicial + (código - 32)×8) +
                (número do byte - 1))
```

Esclarecidos estes aspectos, introduza o programa na máquina e carregue os 768 bytes dos gráficos «ROBOT» fazendo: «GO TO 9100»; após o carregamento, o programa arrancará, imprimindo no écran a imagem da figura 5/2.

Não queremos abandonar esta matéria sem proporcionar ao leitor o conhecimento de outros processos para comando da impressão:

— Função CHR$ e ciclos FOR-NEXT:

PROGRAMA 4/2

```
  10 GOSUB 800
  20 FOR f = 13 TO 17
  30 PRINT AT 6,f;CHR$(f+20);AT 9,f;CHR$(f+39);AT 11,f;CH
R$(f+47); AT 12,f;CHR$(f+52)
  40 NEXT f
  50 FOR f = 12 TO 18
  60 PRINT AT 7,f;CHR$(f+26);AT 8,f;CHR$(f+33)
  70 NEXT f
  80 FOR f = 14 TO 16
  90 PRINT AT 14,f;CHR$(f+60)
 100 NEXT f
 110 PRINT AT 10,13;CHR$ 57+CHR$ 32+CHR$ 58+CHR$ 32+CHR$
59
 120 PRINT AT 13,13;CHR$ 70+CHR$ 71+CHR$ 32+CHR$ 72+CHR$
73
 130 PRINT AT 15,14;CHR$ 77+CHR$ 32+CHR$ 78
 140 GOSUB 900
 150 PRINT AT 19,7;"A CABECA DO ROBOT"
 160 STOP
 800 idem
 900 idem
9000 idem
9010 idem
9020 idem
9100 idem
```

Neste programa utiliza-se a função «CHR$», beneficiando da circunstância de os gráficos estarem colocados em caracteres cujos códigos são consecutivos. Aliás, esta deverá ser uma preocupação do leitor ao construir os seus gráficos: a numeração e a atribuição dos caracteres deve ser racional, isto é, começar no carácter código 33, com o número «1», seguindo para a direita até ao fim da fila, e só então passar à segunda fila.

Como pode verificar, recorremos à instrução «FOR–NEXT» para estabelecer «ciclos», cujos limites correspondem, exactamente, às posições dos gráficos dentro das 32 colunas do écran: exemplifiquemos com as linhas 20 a 40:

Linha 20 — Estabelece-se o ciclo correspondente às colunas de 13 a 17.

Linha 30 — Imprime-se na linha 6, coluna «f», o carácter (CHR$) cujo código seja igual à soma do valor de «f» + 20, isto é, como pretendemos imprimir o gráfico «a2», cujo carácter tem o código 33, será necessário somar (13 + 20) para obtermos o código desejado; como os quatro gráficos seguintes tem códigos consecutivos, a incrementação de «f» coloca automaticamente os caracteres dos códigos «34, 35, 36 e 37», nas colunas «14, 15, 16 e 17».

No caso das linhas 110 a 130, já o mesmo não sucede, pois os códigos não são consecutivos: existem «espaços» (código 32) entre os gráficos, pelo que optámos pela solução de imprimir directamente o «carácter do código indicado».

— Cadeias ALFANUMÉRICAS

PROGRAMA 5/2

```
  10 GOSUB 800: GOSUB 1000
  20 PRINT AT 6,13;a$;AT 7,12;b$;AT 8,12;c$;AT 9,13;d$;AT
10,13;e$
  30 PRINT AT 11,13;f$;AT 12,13;g$;AT 13,13;h$;AT 14,14;i
$;AT 15,14;j$
  40 GOSUB 900
  50 PRINT AT 19,7;"A CABECA DO ROBOT"
  60 STOP
 800 idem
 900 idem
1000 LET a$="!""#$%": LET b$="&()+,": LET c$="-./0123"
1010 LET d$="45678": LET e$="9: ;": LET f$="<=>? "
1020 LET g$="ABCDE": LET h$="FG HI": LET i$="JKL": LET j$
="M N"
1030 RETURN
```



```
9000 idem
9010 idem
9100 idem
```

O processo utilizado neste programa baseia-se na «criação de variáveis alfanuméricas» para cada cadeia de gráficos: como o leitor pode verificar nas linhas 1000 a 1020, criámos uma variável para cada uma das «linhas» de gráficos. Deste modo, quando mandamos imprimir «a$», o écran apresenta os carácteres «!""#$%»; se for «j$», serão impressos os caracteres «MespaçoN»: como entretanto colocámos a variável «CHARS» no endereço 62100, através da sub–rotina 800, são impressos os gráficos correspondentes.

Esta solução apresenta algumas vantagens, nomeadamente quando se pretende realizar «animação gráfica», matéria que abordaremos no parágrafo seguinte.

4 — ANIMAÇÃO GRÁFICA

Certamente que o leitor esperava que abordássemos este aliciante aspecto da problemática dos «gráficos». Não o desiludiremos, embora nos limitemos a algumas «animações», possíveis no âmbito do BASIC. Os efeitos espectaculares obtidos nos jogos comerciais só são realizáveis através de extensas e transcendentes rotinas em «código–máquina», matéria que não cabe nos objectivos deste modesto trabalho.

Mas o BASIC, quando bem dominado, consegue resultados (por vezes inesperados) bastante satisfatórios. Não será necessário saber programar *assembler,* para se utilizarem algumas rotinas em «código–máquina», que se podem «acoplar» aos programas BASIC, com o objectivo de se obterem certos efeitos: essas rotinas encontram–se disponíveis em alguns programas comerciais e em numerosas publicações nacionais e estrangeiras.

Não será dessas rotinas que nos iremos ocupar. O BASIC tem muito para dizer em termos de programação e, desde que não sejam exigidas «proezas de campeão» ou «efeitos altamente subtis», podemos conseguir resultados que darão satisfação à maioria dos utilizadores do ZX SPECTRUM.

Regressemos à realidade com uma proposta ao leitor: vamos «animar» o nosso «ROBOT», levando-o a efectuar certas acções previnmente programadas, aliás, em estrita obediência às «leis da robótica»...

Comecemos por fornecer uma tabela de alguns novos gráficos, que o leitor terá de incluir no conjunto já instalado no endereço 62100: lembramos que, dos 96 caracteres disponíveis, só utilizámos 46, pelo

que podemos ainda dispor de 50 (aliàás 49, retirando o «espaço» —> código 32).

Carregue de novo o GERAGRAF com os gráficos «ROBOT». Por meio da OPÇÃO 2, introduza os decimais da tabela seguinte:

```
 1-Caracter:O Codigo:79
1-Bin : 00000000 = dec:0
2-Bin : 00000000 = dec:0
3-Bin : 00000000 = dec:0
4-Bin : 00000000 = dec:0
5-Bin : 00000000 = dec:0
6-Bin : 00000000 = dec:0
7-Bin : 00000000 = dec:0
8-Bin : 00011111 = dec:31

 2-Caracter:P Codigo:80
1-Bin : 01111110 = dec:126
2-Bin : 11111111 = dec:255
3-Bin : 00000000 = dec:0
4-Bin : 00000000 = dec:0
5-Bin : 00000000 = dec:0
6-Bin : 00000000 = dec:0
7-Bin : 00000000 = dec:0
8-Bin : 11111111 = dec:255

 3-Caracter:Q Codigo:81
1-Bin : 00000000 = dec:0
2-Bin : 00000000 = dec:0
3-Bin : 00000000 = dec:0
4-Bin : 00000000 = dec:0
5-Bin : 00000000 = dec:0
6-Bin : 00000000 = dec:0
7-Bin : 00000000 = dec:0
8-Bin : 11111000 = dec:248

 4-Caracter:R Codigo:82
1-Bin : 00111111 = dec:63
2-Bin : 01100000 = dec:96
3-Bin : 00000000 = dec:0
4-Bin : 00000000 = dec:0
5-Bin : 00000000 = dec:0
6-Bin : 00000000 = dec:0
7-Bin : 00000000 = dec:0
8-Bin : 00000000 = dec:0

 5-Caracter:S Codigo:83
1-Bin : 11111111 = dec:255
2-Bin : 00000000 = dec:0
3-Bin : 00000000 = dec:0
4-Bin : 00000000 = dec:0
5-Bin : 00000000 = dec:0
6-Bin : 00000000 = dec:0
7-Bin : 00000000 = dec:0
8-Bin : 00000000 = dec:0

 6-Caracter:T Codigo:84
1-Bin : 11111100 = dec:252
2-Bin : 00000110 = dec:6
3-Bin : 00000000 = dec:0
4-Bin : 00000000 = dec:0
5-Bin : 00000000 = dec:0
6-Bin : 00000000 = dec:0
7-Bin : 00000000 = dec:0
8-Bin : 00000000 = dec:0

 7-Caracter:U Codigo:85
1-Bin : 11000000 = dec:192
2-Bin : 11000000 = dec:192
3-Bin : 11000000 = dec:192
4-Bin : 11000000 = dec:192
5-Bin : 11000000 = dec:192
6-Bin : 11000000 = dec:192
7-Bin : 11000000 = dec:192
8-Bin : 11000000 = dec:192

 8-Caracter:V Codigo:86
1-Bin : 00000000 = dec:0
2-Bin : 00000000 = dec:0
3-Bin : 00111100 = dec:60
4-Bin : 00011110 = dec:30
5-Bin : 00001111 = dec:15
6-Bin : 00000001 = dec:1
7-Bin : 00000000 = dec:0
8-Bin : 00000000 = dec:0

 9-Caracter:W Codigo:87
1-Bin : 00000000 = dec:0
2-Bin : 00000000 = dec:0
3-Bin : 00000000 = dec:0
4-Bin : 00000000 = dec:0
5-Bin : 10000001 = dec:129
6-Bin : 11000011 = dec:195
7-Bin : 01100110 = dec:102
8-Bin : 01100110 = dec:102

 10-Caracter:X Codigo:88
1-Bin : 00000000 = dec:0
2-Bin : 00000000 = dec:0
3-Bin : 00111100 = dec:60
4-Bin : 01111000 = dec:120
5-Bin : 11110000 = dec:240
6-Bin : 10000000 = dec:128
7-Bin : 00000000 = dec:0
8-Bin : 00000000   dec:0
```



```
 11-Caracter:Y Codigo:89
1-Bin : 00000011 = dec:3
2-Bin : 00000011 = dec:3
3-Bin : 00000011 = dec:3
4-Bin : 00000011 = dec:3
5-Bin : 00000011 = dec:3
6-Bin : 00000011 = dec:3
7-Bin : 00000011 = dec:3
8-Bin : 00000011 = dec:3
```

Grave os bytes «ROBOT» por cima da anterior gravação, pois trata-se do mesmo título, igualmente com 768 bytes, mas agora completados com 11 novos gráficos, colocados em 11 dos 46 caracteres disponíveis.

O programa seguinte, intitulado «ANIMAÇÃO 1», demonstrará uma das hipóteses de animação do nosso ROBOT. Utilizará instruções bastante simples, com criação de novas variáveis para cada novo grupo de gráficos, bem como algumas sub-rotinas que reúnem as necessárias instruções para a animação.

Vejamos o programa ANIMAÇÃO 1:

PROGRAMA 6/2 — ANIMAÇÃO 1

```
  1 REM ANIMACAO 1 - ROBOT
 10 GO SUB 800: GO SUB 1000
 15 PRINT AT 6,13;a$;AT 8,12;c$;AT 9,13;d$;AT 10,13;e$
 20 PRINT AT 13,13;h$;AT 14,14;i$;AT 15,14;j$
 25 GO SUB 900: PRINT #0;"        Prima ENTER para parar
    ": GO SUB 800
 30 GO SUB 60: GO SUB 100: PAUSE 100: GO SUB 500
 35 GO SUB 80: GO SUB 200: PAUSE 100: GO SUB 500
 40 GO TO 30
 60 PRINT AT 7,12;k$AT 11,13;f$;AT 12,13;g$: GO SUB 900:
PRINT AT 19,9;"ROBOT  CONTENTE": GO SUB 800: RETURN
 80 PRINT AT 7,12;k$;AT 11,13;l$;AT 12,13;m$: GO SUB 900
: PRINT AT 19,9;"ROBOT  ZANGADO": GO SUB 800: RETURN
100 FOR f=10 TO 40: BEEP .01,f: NEXT f: RETURN
200 FOR f=40 TO  10 STEP -1: BEEP .01,f: NEXT f: RETURN
500 IF CODE INKEY$=13 THEN  GO SUB 900: PRINT AT 0,0; BR
IGHT 1;"  PARA ARRANCAR DE NOVO -> RUN  ": STOP
505 RETURN
800 POKE 23606,148: POKE 23607,241: RETURN
900 POKE 23606,0: POKE 23607,60: RETURN
1000 LET a$="!""#$%": LET b$="&'()*+,": LET c$="-./0123":
LET d$="45678"
1010 LET e$="9 : ;": LET f$="<=>?@": LET g$="ABCDE": LET
h$="FG HI"
1020 LET i$="JKL": LET j$ "M N": LET k$ "&UVWXY,": LET l$
```

```
="<OPQ@": LET m$="ARSTE"
1030 RETURN
9000 SAVE "ANIMACAO 1" LINE 9100
9010 SAVE "ROBOT"CODE 62100,768
9020 VERIFY "": VERIFY ""CODE : STOP
9100 CLEAR 62099: LOAD ""CODE : RUN
```

Vejamos como funcionam algumas instruções:

Linhas 15/20 — São impressos os gráficos contidos nas variáveis «a$, c$, d$, e$, h$, i$ e j$»: os restantes gráficos serão impressos nas sub–rotinas 60 e 80.

Linha 30 — Chamada a sub–rotina 60: esta imprime os gráficos contidos nas variáveis «b$, f$ e g$»; chama a sub–rotina 900 e imprime a frase «ROBOT CONTENTE» (figura já nossa conhecida); chama de novo a sub–rotina 800 e retorna à linha 30, sub–rotina 100, que dá origem a uma rápida sucessão de sons, do grave para o agudo; retorno à linha 30 para uma pausa de 2 segundos (PAUSE 100); chama a sub–rotina 500, onde é testado se foi premida a tecla «ENTER»: na afirmativa, o programa pára; na negativa, retorno à linha 35.

Linha 35 — Chamada a sub–rotina 80: esta imprime, nas mesmas coordenadas anteriores, os gráficos contidos nas variáveis «k$, l$ e m$» e, após ter impresso a frase «ROBOT ZANGADO», retorno à linha 35 para a rotina 200, que dá origem a uma rápida sucessão de sons, mas agora do agudo para o grave; passagem pela sub–rotina 500 e retorno à linha 40, que devolve o comando de novo à linha 30: estamos num «ciclo sem fim», do qual só é possível sair premindo a tecla «ENTER» (código 13), ou fazendo «BREAK».

Introduza o programa e carregue os gráficos «ROBOT» com «GO TO 9100. Como o leitor verificará, o efeito não será espectacular, mas constitui uma forma de «animação gráfica», obtida através de meios extremamente simples.

Grave o programa com «GO TO 9000», pois vamos proporcionar–lhe mais duas «animações» sobre o tema «ROBOT».

Segue–se uma terceira tabela para mais 25 novos gráficos, a serem colocados nos caracteres dos códigos 90 (Z) a 114 (r):

```
 1-Caracter:Z Codigo:90
1-Bin : 00111100 = dec:60
2-Bin : 01000010 = dec:66
3-Bin : 10000001 = dec:129
4-Bin : 10110001 = dec:177
5-Bin : 10110001 = dec:177
6-Bin : 10000001 = dec:129
7-Bin : 01000010 = dec:66
8-Bin : 00111100 = dec:60
```

```
 4-Caracter:] Codigo:93
1-Bin : 00111100 = dec:60
2-Bin : 01000010 = dec:66
3-Bin : 10000001 = dec:129
4-Bin : 10001101 = dec:141
5-Bin : 10001101 = dec:141
6-Bin : 10000001 = dec:129
7-Bin : 01000010 = dec:66
8-Bin : 00111100 = dec:60
```



```
 2-Caracter:[ Codigo:91
1-Bin : 00111100 = dec:60
2-Bin : 01000010 = dec:66
3-Bin : 10000001 = dec:129
4-Bin : 10110001 = dec:177
5-Bin : 10110001 = dec:177
6-Bin : 10000001 = dec:129
7-Bin : 01000010 = dec:66
8-Bin : 00111100 = dec:60
```

```
 5-Caracter:↑ Codigo:94
1-Bin : 00000000 = dec:0
2-Bin : 00000000 = dec:0
3-Bin : 00000000 = dec:0
4-Bin : 00000000 = dec:0
5-Bin : 00000000 = dec:0
6-Bin : 00000000 = dec:0
7-Bin : 00000000 = dec:0
8-Bin : 00011111 = dec:31
```

```
 3-Caracter:\ Codigo:92
1-Bin : 00111100 = dec:60
2-Bin : 01000010 = dec:66
3-Bin : 10000001 = dec:129
4-Bin : 10001101 = dec:141
5-Bin : 10001101 = dec:141
6-Bin : 10000001 = dec:129
7-Bin : 01000010 = dec:66
8-Bin : 00111100 = dec:60
```

```
 6-Caracter:_ Codigo:95
1-Bin : 01111110 = dec:126
2-Bin : 11111111 = dec:255
3-Bin : 00000000 = dec:0
4-Bin : 00000000 = dec:0
5-Bin : 00000000 = dec:0
6-Bin : 00000000 = dec:0
7-Bin : 00000000 = dec:0
8-Bin : 11111111 = dec:255
```

```
 7-Caracter:` Codigo:96
1-Bin : 00000000 = dec:0
2-Bin : 00000000 = dec:0
3-Bin : 00000000 = dec:0
4-Bin : 00000000 = dec:0
5-Bin : 00000000 = dec:0
6-Bin : 00000000 = dec:0
7-Bin : 00000000 = dec:0
8-Bin : 11111000 = dec:248
```

```
 12-Caracter:e Codigo:101
1-Bin : 00000000 = dec:0
2-Bin : 11000011 = dec:195
3-Bin : 10000001 = dec:129
4-Bin : 00000000 = dec:0
5-Bin : 00000000 = dec:0
6-Bin : 00000000 = dec:0
7-Bin : 00000000 = dec:0
8-Bin : 00000000 = dec:0
```

```
 8-Caracter:a Codigo:97
1-Bin : 00011111 = dec:31
2-Bin : 00000000 = dec:0
3-Bin : 00000000 = dec:0
4-Bin : 00000000 = dec:0
5-Bin : 00000000 = dec:0
6-Bin : 00000000 = dec:0
7-Bin : 00000000 = dec:0
8-Bin : 00000000 = dec:0
```

```
 13-Caracter:f Codigo:102
1-Bin : 00000000 = dec:0
2-Bin : 00000000 = dec:0
3-Bin : 11000000 = dec:192
4-Bin : 11110000 = dec:240
5-Bin : 00111100 = dec:60
6-Bin : 00001111 = dec:15
7-Bin : 00000000 = dec:0
8-Bin : 00111100 = dec:60
```

```
 9-Caracter:b Codigo:98
1-Bin : 11111111 = dec:255
2-Bin : 00000000 = dec:0
3-Bin : 00000000 = dec:0
4-Bin : 00000000 = dec:0
5-Bin : 00000000 = dec:0
6-Bin : 00000000 = dec:0
7-Bin : 00000000 = dec:0
8-Bin : 00000000 = dec:0
```

```
 14-Caracter:g Codigo:103
1-Bin : 01000010 = dec:66
2-Bin : 10000001 = dec:129
3-Bin : 10000001 = dec:129
4-Bin : 10011001 = dec:153
5-Bin : 10011001 = dec:153
6-Bin : 10000001 = dec:129
7-Bin : 10000001 = dec:129
8-Bin : 01000010 = dec:66
```

```
10-Caracter:c Codigo:99
1-Bin : 11111000 = dec:248
2-Bin : 00000000 = dec:0
3-Bin : 00000000 = dec:0
4-Bin : 00000000 = dec:0
5-Bin : 00000000 = dec:0
6-Bin : 00000000 = dec:0
7-Bin : 00000000 = dec:0
8-Bin : 00000000 = dec:0

11-Caracter:d Codigo:100
1-Bin : 00000000 = dec:0
2-Bin : 00000000 = dec:0
3-Bin : 00000011 = dec:3
4-Bin : 00001111 = dec:15
5-Bin : 00111100 = dec:60
6-Bin : 11110000 = dec:240
7-Bin : 00000000 = dec:0
8-Bin : 00111100 = dec:60

15-Caracter:h Codigo:104
1-Bin : 01000010 = dec:66
2-Bin : 10000001 = dec:129
3-Bin : 10000001 = dec:129
4-Bin : 10011001 = dec:153
5-Bin : 10011001 = dec:153
6-Bin : 10000001 = dec:129
7-Bin : 10000001 = dec:129
8-Bin : 01000010 = dec:66

16-Caracter:i Codigo:105
1-Bin : 00000000 = dec:0
2-Bin : 00000000 = dec:0
3-Bin : 00000000 = dec:0
4-Bin : 00000000 = dec:0
5-Bin : 00000000 = dec:0
6-Bin : 00000000 = dec:0
7-Bin : 00000001 = dec:1
8-Bin : 00000011 = dec:3

17-Caracter:j Codigo:106
1-Bin : 01111110 = dec:126
2-Bin : 11111111 = dec:255
3-Bin : 00000000 = dec:0
4-Bin : 00000000 = dec:0
5-Bin : 00000000 = dec:0
6-Bin : 01111110 = dec:126
7-Bin : 11100111 = dec:231
8-Bin : 00000000 = dec:0

18-Caracter:k Codigo:107
1-Bin : 00000000 = dec:0
2-Bin : 00000000 = dec:0
3-Bin : 00000000 = dec:0
4-Bin : 00000000 = dec:0
5-Bin : 00000000 = dec:0
6-Bin : 00000000 = dec:0
7-Bin : 10000000 = dec:128
8-Bin : 11000000 = dec:192

19-Caracter:l Codigo:108
1-Bin : 00000110 = dec:6
2-Bin : 00000100 = dec:4
3-Bin : 00000110 = dec:6
4-Bin : 00000011 = dec:3
5-Bin : 00000001 = dec:1
6-Bin : 00000000 = dec:0
7-Bin : 00000000 = dec:0
8-Bin : 00000000 = dec:0

20-Caracter:m Codigo:109
1-Bin : 00000000 = dec:0
2-Bin : 00000000 = dec:0
3-Bin : 00000000 = dec:0
4-Bin : 00000000 = dec:0
5-Bin : 11100111 = dec:231
6-Bin : 01111110 = dec:126
7-Bin : 00000000 = dec:0
8-Bin : 00000000 = dec:0

21-Caracter:n Codigo:110
1-Bin : 01100000 = dec:96
2-Bin : 00100000 = dec:32
3-Bin : 01100000 = dec:96
4-Bin : 11000000 = dec:192
5-Bin : 10000000 = dec:128
6-Bin : 00000000 = dec:0
7-Bin : 00000000 = dec:0
8-Bin : 00000000 = dec:0

22-Caracter:o Codigo:111
1-Bin : 00111100 = dec:60
2-Bin : 00000000 = dec:0
3-Bin : 00000000 = dec:0
4-Bin : 00111111 = dec:63
5-Bin : 01000000 = dec:64
6-Bin : 00000000 = dec:0
7-Bin : 00000000 = dec:0
8-Bin : 00000000 = dec:0
```



```
23-Caracter:p Codigo:112
1-Bin : 00111100 = dec:60
2-Bin : 00000000 = dec:0
3-Bin : 00000000 = dec:0
4-Bin : 11111100 = dec:252
5-Bin : 00000010 = dec:2
6-Bin : 00000000 = dec:0
7-Bin : 00000000 = dec:0
8-Bin : 00000000 = dec:0

24-Caracter:q Codigo:113
1-Bin : 00001000 = dec:8
2-Bin : 00001000 = dec:8
3-Bin : 00001000 = dec:8
4-Bin : 00001000 = dec:8
5-Bin : 11111111 = dec:255
6-Bin : 11111111 = dec:255
7-Bin : 00000000 = dec:0
8-Bin : 00000000 = dec:0

25-Caracter:r Codigo:114
1-Bin : 00010000 = dec:16
2-Bin : 00010000 = dec:16
3-Bin : 00001000 = dec:8
4-Bin : 00001000 = dec:8
5-Bin : 11111111 = dec:255
6-Bin : 11111111 = dec:255
7-Bin : 00000000 = dec:0
8-Bin : 00000000 = dec:0
```

Mais uma vez, com o GERAGRAF e após ter carregado a última versão dos gráficos «ROBOT» (não esquecendo de indicar o endereço 62100), dê entrada aos decimais desta tabela.

Como pode verificar, a criação destes 25 novos gráficos deixa livres 13 caracteres: 46 + 11 + 25 = 82 —> 95 — 82 = 13 (naturalmente que não contámos com o carácter «espaço»).

Passemos de imediato aos programas «ANIMAÇÃO 2» e «ANIMAÇÃO 3», na convicção de que o leitor gravou cuidadosamente os gráficos onde incluiu os novos 25 que vamos aplicar nestes programas:

PROGRAMA 7/2 — ANIMAÇÃO 2

```
  1 REM ANIMACAO 2 - ROBOT
 10 GO SUB 800: GO SUB 1000
 15 PRINT AT 7,12;b$;AT 9,13;d$;AT 10,13;e$
 20 PRINT AT 11,13;m$;AT 12,13;n$;AT 13,13;h$;AT 14,14;i
$;AT 15,14;j$
 25 GO SUB 900: PRINT #0;"        Prima ENTER para parar
  ": GO SUB 800
 30 GO SUB 60: BEEP .03,40: PAUSE 30: GO SUB 500
 35 GO SUB 80: BEEP .03,40: PAUSE 30: GO SUB 500
```

```
  40 GO TO 30
  60 PRINT AT 6,13;o$;AT 8,12;k$: GO SUB 900: PRINT AT 19
,6;"ROBOT  DESCONFIADO": GO SUB 800: RETURN
  80 PRINT AT 6,13;p$;AT 8,12;l$: RETURN
 500 IF CODE INKEY$=13 THEN  GO SUB 900: PRINT AT 0,0; BR
IGHT 1;"  PARA ARRANCAR DE NOVO -> RUN  ": STOP
 505 RETURN
 800 POKE 23606,148: POKE 23607,241: RETURN
 900 POKE 23606,0: POKE 23607,60: RETURN
1000 LET a$="!""#$%": LET b$="&'()*+,": LET d$="45678"
1010 LET c$="9 : ;": LET f$="<->?@": LET g$="ABCDE": LET
h$="FG HI"
1020 LET i$="JKL": LET j$="M N": LET k$="-.Z0[23": LET l$
="-.\0]23": LET m$="<^_`@": LET n$="AabcE"
1030 LET o$="!q#$%": LET p$="!""#r%"
1040 RETURN
9000 SAVE "ANIMACAO 2" LINE 9100
9010 SAVE "ROBOT"CODE 62100,768
9020 VERIFY "": VERIFY ""CODE : STOP
9100 CLEAR 62099: LOAD ""CODE : RUN
```

PROGRAMA 8/2 — ANIMAÇÃO 3

```
   1 REM ANIMACAO 3 - ROBOT
  10 GO SUB 800: GO SUB 1000
  15 PRINT AT 6,13;a$;AT 7,12;b$;AT 8,12;c$;AT 9,13;d$;AT
10,13;e$
  20 PRINT AT 11,13;f$;AT 12,13;g$;AT 13,13;h$;AT 14,14;i
$;AT 15,14;j$
  25 GO SUB 900: INPUT "": PRINT #0;"     Prima ENTER par
a parar     ": GO SUB 800
  30 GO SUB 80: GO SUB 100: PAUSE 100: GO SUB 500: GO SUB
60: GO SUB 200: PAUSE 100: GO SUB 500: GO TO 15
  60 PRINT AT 7,12;k$;AT 8,12;l$;AT 9,13;o$;AT 11,13;m$;A
T 12,13;n$: GO SUB 900: PRINT AT 19,8;"ROBOT   PERPLEXO":
GO SUB 800: RETURN
  80 GO SUB 900: PRINT AT 19,8;"ROBOT EXPECTANTE": GO SUB
800: RETURN
 100 FOR f=1 TO 50: BEEP .005,10: NEXT f: RETURN
 200 FOR f=5 TO 50: BEEP .005,f: NEXT f: RETURN
 500 IF CODE INKEY$=13 THEN  GO SUB 900: PRINT AT 0,0; BR
IGHT 1;"  PARA ARRANCAR DE NOVO -> RUN  ": STOP
 505 RETURN
```



```
 800 POKE 23606,148: POKE 23607,241: RETURN
 900 POKE 23606,0: POKE 23607,60: RETURN
1000 LET a$="!""#$%": LET b$="&'()*+,": LET c$="-./0123":
LET d$="45678"
1010 LET e$="9 : ;": LET f$="<^_`@": LET g$="AabcE": LET
h$="FG HI"
1020 LET i$="JKL": LET j$="M N": LET k$="&UdefY": LET l$
="-.g0h23": LET m$="<ijk@": LET n$="AlmnE"
1030 LET o$="4o6p8": LET p$="!q#$%": LET q$="!""#r%"
1035 RETURN
9000 SAVE "ANIMACAO 3" LINE 9100
9010 SAVE "ROBOT"CODE 62100,768
9020 VERIFY "": VERIFY ""CODE : STOP
9100 CLEAR 62099: LOAD ""CODE : RUN
```

Pensamos que estes dois últimos programas dispensam quaisquer explicações: pela sua simplicidade, são suficientemente explícitos para que o leitor, com os conhecimentos já adquiridos, possa compreender o seu funcionamento.

Terminamos aqui o capítulo II, com a esperança de termos contribuído para a solução dos dois principais problemas que perturbam os utilizadores do ZX SPECTRUM: *a alegada limitação do número de gráficos disponíveis e o «mito» do código-máquina.* Acredite, leitor: nada temos contra o código-máquina, aliás vamos aplicá-lo em alguns programas dos próximos capítulos, única forma de se obterem os resultados desejados.

Como dizíamos no início deste parágrafo: O BASIC TEM MUITO PARA DAR, QUANDO BEM DOMINADO E APLICADO COM IMAGINAÇÃO.

CAPÍTULO III

EM FALTA

1 — INTRODUÇÃO

Neste capítulo vamos ocupar-nos da programação de «caracteres gigantes» a que chamaremos, abreviadamente, «CATGIG», possibilitando a impressão no écran de caracteres de qualquer dimensão, a partir dos abecedários ou gráficos em processamento.

Quando dizemos «qualquer dimensão», torna-se evidente que estaremos sempre condicionados aos limites do écran e, se tivermos uma «frase» ou um conjunto de gráficos com muitos caracteres, teremos de limitar a dimensão, para que a imagem «não saia» do écran.

Admitimos que o leitor já tenha encontrado e ensaiado algumas rotinas em código-máquina (existem publicações com estas rotinas), que permitem construir letras de dimensão superior ao normal, imprimindo-as numa determinada zona do écran. A rotina que vamos utilizar, com 300 bytes de comprimento, é comandada por uma sub-rotina BASIC e permite dimensionar os caracteres, colocá-los em qualquer linha, centrar a «frase» ou o conjunto, dar maior ou menor espaço entre caracteres e iniciar a impressão em qualquer coluna.

O leitor terá ocasião de apreciar os esplêndidos efeitos que esta rotina nos proporciona e aprender a aplicá-los nos seus programas.

1 — A ROTINA CATGIG

A rotina «CATGIG» é uma rotina em código-máquina, com 300 bytes, preparada para o endereço 45000.

Quando dizemos «preparada», isto significa que o seu endereço não pode ser alterado pelos mesmos métodos e facilidade que utilizámos nos abecedários e nos gráficos.

Para alterar (relocatar) o endereço desta rotina teremos de recorrer a um programa especial, o qual será facultado ao leitor na próxima secção. Por agora, todos os ensaios e experiências que iremos sugerir ao leitor serão feitos no endereço 45000.

Comecemos por fornecer ao leitor a listagem dos 300 bytes convertidos em números decimais, sob a forma de instruções «DATA». Estes decimais estarão incluídos no programa 1/3, destinado ao seu carregamento automático no endereço 45000.

LISTAGEM DOS DECIMAIS DA ROTINA CATGIG

```
1000 DATA 33,15,91,126,35,34,0,91,111,60,200,38,0,41,41,
41,237,75,54,92,9,
1005 DATA 62,8,50,4,91,58,11,91,50,9,91,58,10,91,50,8,91
,62,9,50,5,
1010 DATA 91,126,35,34,2,91,7,50,6,91,58,5,91,61,32,50,5
8,4,91,61,32
1015 DATA 24,58,14,91,71,58,12,91,79,58,10,91,129,5,32,2
52,50,10,91,42,0,
1020 DATA 91,195,203,175,50,4,91,58,13,91,71,58,9,91,12
8,50,9,91,42,2,91,
1025 DATA 195,232,175,50,5,91,58,12,91,71,58,9,91,50,7,9
1,58,13,91,79,197,
1030 DATA 205,108,176,193,58,7,91,60,50,7,91,13,32,241,5
8,8,91,60,50,8,91,
1035 DATA 5,32,221,58,6,91,195,248,175,128,64,32,16,8,4,
2,1,58,142,92,238,
1040 DATA 255,71,58,141,92,160,71,58,8,91,230,248,111,5
8,7,91,254,192,208,31,31,
1045 DATA 31,230,31,103,203,28,203,29,203,28,203,29,203,
28,203,29,62,88,180,103,58,
1050 DATA 142,92,166,176,119,58,7,91,71,230,7,246,64,103
,120,31,31,31,230,24,180,
1055 DATA 103,120,23,23,230,224,111,58,8,91,71,31,31,31,
230,31,181,111,235,33,100,
1060 DATA 176,120,230,7,79,6,0,9,70,26,33,6,91,203,70,40
,3,176,18,201,47,
1065 DATA 176,47,18,201,0,0,0,0,0,0,0,0,0,0,0,0,0,0,23,2
20,10,
1070 DATA 206,11,231,80,26,23,0
```

Antes de passarmos ao programa destinado ao processamento destes decimais, falemos um pouco da instrução «DATA», cujo significado é «dados». O BASIC utiliza esta instrução para a leitura ou ar-

mazenamento de dados, que tanto podem ser valores numéricos como os da listagem acima, como cadeias alfanuméricas:

```
1000 DATA "ABCDE","A1B2C3","$#%&"
1010 DATA "ROBOT","PROGRAMA", data"
```

Os dados contidos nas instruções «DATA» são lidos pela instrução «READ», através de uma variável numérica ou alfanumérica, normalmente no interior de um ciclo FOR-NEXT. O conteúdo da variável é aplicado directamente ao programa ou guardado em quadros «DIM».

Esta sucinta explicação (que o leitor provavelmente já conhece), tem por objectivo «abrir o diálogo» para um método de programação que passaremos a utilizar com bastante frequência, nos programas que se seguem.

PROGRAMA 1/3

```
  1 REM PROCESSAMENTO CATGIG          COPYRIGHT R.CASQUIL
HO 1986
  10 CLS : PRINT AT 0,13;"CATGIG"'''"Rotina em Codigo M
aquina paraexecucao de Letras Gigantes."
  20 PRINT '"A rotina ficara no endereco :  45000 e tem
300 bytes de compri-mento."''"Se for necessario colocar a
ro- tina noutro endereco,utilizar o programa ~RELOCG
IG~."
  25 PRINT '"Prima qualquer tecla para inici-ar o process
amento do codigo:": PAUSE 0
  30 PRINT '''TAB 11; FLASH 1;"UM MOMENTO"
  40 LET end=45000: LET t=0
  45 RESTORE 1000: FOR i=end TO end+300
  50 READ d: POKE i,d: LET t=t+d
  55 IF i=45300 AND t=23497 THEN  GO TO 2000
  60 NEXT i
  70 PRINT '"Ha erro nas DATA's .Verifique.": BEEP .5,0:
PAUSE 200: CLS : LIST 1000: STOP
1000 DATA 33,15,91,126,35,340,91,111,60,200,38,0,41,4 1,4
1,237,75,54,92,9,62,8,50,4,91,58,11,91,50,9,91,58,10,91,5
0,8,91,62,9,50,5,91,126,35,34,2,91,7,50,6,91,58,5,91,61,3
2,50,58,4,91,61,32
1005 DATA 24,58,14,91,71,58,12,91,79,58,10,91,129,5,32,25
2,50,10,91,42,0,91,195,203,175,50,4,91,58,13,91,71,58,9,9
1,128,50,9,91,42,2,91,195,232,175,50,5,91,58,12,91,71,58,
9,91,50,7,91,58,13
1010 DATA 91,79,197,205,108,176,1 93,58,7,91,60,50,7,91,
```



```
13,32,241,58,8,91,60,50,8,91,5,32,221,58,6,91,195,248,175
,128,64,32,16,8,4,2,1,58,142,92,238,255,71,58,141
1015 DATA 92,160,71,58,8,91,230,248,11 1,58,7,91,254,192,
208,31,31,31,230,31,103,203,28,203,29,203,28,203,29,203,2
8,203,29,62,88,180,103,58,142,92,166,176,119
1020 DATA 58,7,91,71,230,7,246,64,103,12 0,31,31,31,230,2
4,180,103,120,23,23,230,224,111,58,8,91,71,31,31,31,230,3
1,181,111,235,33,100,176,120,230,7,79,6,0,9
1025 DATA 70,26,33,6,91,203,70,40,3,176,18,201,47,176,47,
18,201,0,0,0,0,0,0,0,0,0,0,0,0,0,0,23,220,10,206,11,231,8
0,26,23,0
2000 CLS : PRINT "Processo terminado sem erros."'"Tem ago
ra em memoria o codigo  maquina da rotina ~CATGIG~"'"Vam
os proceder a sua gravacao emcassette:"
2010 PRINT '"Prepare o gravador com uma cas- sette limpa:
prima ENTER:"'"SAVE ""CATGIG"" CODE 45000,300": PAUSE 0
2020 SAVE "CATGIG"CODE 45000,300
2030 PRINT '"Rebobine para verificar:"
2040 VERIFY ""CODE 45000,300
2050 STOP
9000 SAVE "CATGIG" LINE 10
9010 VERIFY "": STOP
```

Como funciona

Linhas 10/25 — Impressão no écran de mensagens destinadas ao utilizador, com uma «PAUSE 0», fazendo o programa aguardar pela pressão sobre uma tecla.

Linhas 30/60 — O processamento dos 300 decimais demora alguns segundos: a mensagem «UM MOMENTO» destina-se a informar o utilizador de que o programa não parou, sendo a espera provisível; colocação do valor «45000» na variável «end»; colocação da variável «t» a 0; a instrução «RESTORE 1000» tem por objectivo instruir o computador quanto ao número de linha onde se encontra a primeira instrução «DATA»: neste caso seria dispensável a indicação do número de linha, visto existir um único grupo de dados que são lidos consecutivamente — o computador vai automaticamente para a primeira linha de «DATA»; estabelecimento do ciclo FOR–NEXT «i»: tem o mesmo significado que «FOR i = 45000 TO 45300»; leitura dos valores em «DATA» pela variável «d»; introdução (POKE) do valor contido em «d» no endereço «i»: no primeiro ciclo, é introduzido «33» no endereço 45000; no ciclo seguinte é introduzido «15» no endereço 45001, etc., até «i» ter atingido o valor 45300; incremento do valor de «t» com o valor do decimal acabado de introduzir: esta variável destina-se a guardar o somatório do valor dos 300 decimais em «DATA»; instrução condicio-

nal que verifica se «i» atingiu o valor de 45300 e, simultaneamente, se «t» atingiu o valor de 23497 (total dos 300 decimais): na negativa, o comando passa à linha 70, sendo impressa a mensagem «Há erro nos DATAs. Verifique.», com um aviso sonoro e uma pausa de 4 segundos, no fim da qual o programa pára, ficando listada a linha 1000: isto proporciona ao leitor a imediata verificação dos valores; na afirmativa, o comando passa à linha 2000; o ciclo é fechado pela linha 60.

Linhas 1000/1025 — Conjunto de instruções «DATA» contendo os 300 bytes da rotina «CATGIG» convertidos em decimais. O leitor notará que, no programa, utilizámos menos linhas de «DATA» do que na listagem: fizemo-lo deliberadamente, a fim de termos oportunidade de lhe dar um conselho: *quanto menor for o número de dados contidos numa linha de «DATA», mais fácil se torna a verificação e mais rápida será a correcção.* Deste modo, fica ao seu critério usar a solução do programa (não hesite), ou a solução da listagem inicial.

Linhas 2000/2040 — Impressão da mensagem informando que o processamento se efectuou sem erros; instruções para preparar e executar a gravação dos 300 bytes da rotina «CATGIG» no endereço 45000; finalmente, a verificação.

Linhas 9000/9010 — Gravação do programa, com arranque na linha 10 e verificação.

Introduza o programa com a maior atenção, faça GO TO 10 e grave a rotina em código-máquina «CATGIG» com os cuidados habituais: grave depois o programa BASIC noutra cassete: na eventualidade de avaria da primeira, tem sempre a hipótese de voltar a processar a rotina.

É altura de corresponder à expectativa do leitor, proporcionando-lhe uma série de programas para aplicação da «CATGIG»:

PROGRAMA 2/3

```
  1 REM ENSAIO 1 CATGIG
  5 PAPER 6: INK 1: BORDER 6: CLS : GO TO 50
 10 LET w=23306: POKE w,(256-e*1*LEN g$)/2
 15 POKE w+1,y: POKE w+2,l: POKE w+3,a: POKE w+4,e
 20 LET w=w+4: FOR f=1 TO LEN g$: POKE w+f,CODE g$(f):
NEXT f
 25 POKE w+f,255: RANDOMIZE USR 45000
 30 RETURN
 50 LET g$="ROTINA CATGIG": LET y=70: LET a=4: LET l=2:
LET e=9: GO SUB 10
 60 GO SUB 1000
```



```
 70 CLS : PRINT  BRIGHT 1;" Utilizacao  da rotina ~CATG
IG~"
 80 PRINT '"Vamos ensaiar a rotina em CodigoMaquina ~CA
TGIG~,  atribuindo  di-ferentes  valores  as  variaveis
de comando, que sao as seguintes:"
 90 PRINT '"~y~- posicao em altura"'"~a~- altura das l
etras"'"~l~- largura das letras"'"~e~- espaco entre let
ras"
 100 PRINT '"A posicao em largura e' dada au-tomaticament
e."'TAB 6;"A palavra e'~ROBOT~."
 110 BEEP .2,30: INPUT "Valor para a posicao em altura
(~y~ entre 0 e 150) ";y
 115 IF y<0 OR y>150 THEN  GO TO 120
 120 BEEP .2,30: INPUT "Valor para a altura das letras
(~a~ entre 1 e 20) ";a
 125 IF a<1 OR a>20 THEN  GO TO 120
 130 BEEP .2,30: INPUT "Valor para a largura das letras
(~l~ entre 1 e 6) ";l
 135 IF l<1 OR l>6 THEN  GO TO 130
 140 BEEP .2,30: INPUT "Valor para o espaco entre letras
(~e~ entre 8 e 16) ";e
 145 IF e<8 OR e>16 THEN  GO TO 140
 150 CLS : LET g$="ROBOT": GO SUB 10
 160 GO SUB 1000
 170 CLS : PRINT "Indicou os seguintes valores:";AT 8,0;"
Posicao ~y~:";y'"Altura  ~a~:";a'"Largura ~l~:";l'"Es
paco  ~e~:";e
 180 PRINT #0;TAB 7; BRIGHT 1;"Novo ensaio ? (s/n)": BEEP
.3,20
 190 IF INKEY$<>"s" AND INKEY$<>"S" AND INKEY$<>"n" AND
INKEY$<>"N" THEN  GO TO 190
 200 IF INKEY$="s" OR INKEY$="S" THEN  CLS : GO TO 110
 210 STOP
1000 PRINT #0; INVERSE 1;"        Prima qualquer tecla
     ": BEEP .05,30: BEEP .05,40: PAUSE 0: RETURN
9000 SAVE "ENSAIO 1" LINE 9100
9010 SAVE "CATGIG"CODE 45000,300
9020 VERIFY "": VERIFY ""CODE : STOP
9100 CLEAR 44999: LOAD "CATGIG"CODE 45000,300: RUN
```

O programa 2/3, tal como qualquer outro programa onde seja aplicada a rotina «CATGIG», possui uma pequena sub-rotina em BASIC, destinada ao «tratamento» dos valores que irão ser colocados nas quatro variáveis de comando da impressão.

A referida sub-rotina nada tem de transcendente, mas só será le-

gível para quem conheça minimamente o código–máquina: duvidamos que seja o caso do leitor, pois não estaria connosco neste momento. Se, como imaginamos, do código–máquina só conhece os efeitos, estaria o leitor perdendo o seu tempo e paciência, com fastidiosas e pouco convincentes explicações.

Deste modo, vamos passar à parte que, verdadeiramente, pode interessar os nossos leitores: como aplicar e «controlar» a rotina «CATGIG» nos seus próprios programas:

Linha 5 — Definição dos atributos do «papel», «tinta» e «bordo»: se o leitor só possui «preto e branco», altere as instruções em conformidade; passagem à linha 50.

Linha 50 — Colocação da «cadeia de caracteres» na variável «g$» e definição dos valores a atribuir às quatro variáveis de comando da impressão dos caracteres. Mais adiante desenvolveremos este tópico.

Linha 60 — Chamada da sub–rotina 1000, que imprime na base do écran a mensagem «Prima qualquer tecla», acompanhada de um aviso sonoro; «PAUSE 0» aguardando a pressão sobre uma tecla.

Linhas 70/100 — Impressão da página com indicações para o utilizador, terminando com a frase «A palavra é ROBOT»: isto significa que o leitor fará os seus ensaios com este grupo de caracteres, embora mais tarde possa alterá–los como entender.

Linhas 110/145 — Entrada dos valores para as quatro variáveis do comando da impressão.

Linha 150 — Limpeza do écran; colocação da cadeia de caracteres «ROBOT» na variável «g$»; chamada à sub–rotina 10.

Linha 160 — Chamada à sub–rotina 1000.

Linha 170 — Limpeza do écran e impressão dos valores atribuídos às quatro variáveis, permitindo a sua anotação para uso futuro.

Linhas 180/210 — Rotina que imprime na base do écran a frase «Novo ensaio ? (s/n)»: repare o leitor nas instruções condicionais da linha 190, bloqueando o programa até que as teclas «s» ou «n» sejam premidas (tanto minúsculas como maiúsculas): se premirmos «s», o écran é limpo e o comando volta à linha 110 para novo ensaio; premindo «n», o programa pára.

Linha 1000 — Já referida na linha 60.

Linhas 9000/9100 — Instruções já bem conhecidas do leitor, para gravação do BASIC e da rotina «CATGIG», acompanhadas das indispensáveis verificações; a linha 9100 prepara a «RAMTOP» para receber o endereço 45000 e carrega automaticamente os bytes do código–máquina, depois do BASIC. Arranque por «RUN».

Voltemos então à linha 50: a variável «g$» contém o conjunto de caracteres que queremos imprimir no écran: no caso, é a frase «ROTINA CATGIG». Analisemos as quatro variáveis de comando:

— *Variável «y»:* o valor a atribuir a esta variável está directamente relacionado com a rêede de pixels do écran. Reportemo–nos à p. 70 do *Manual do ZX SPECTRUM:* verificará o leitor que existem 22 linhas com 8 pixels cada, o que perfaz um total de 176 pixels, numerados de 0 a 175 em intervalos de 8. Aliás, se o leitor já praticou com as instruções «PLOT» e DRAW», está suficientemente familiarizado com o problema. Mas a rotina «CATGIG» introduz, aqui, um elemento de perturbação: coloca o «pixel 0» no topo do écran e o «pixel 175» na base. Desta forma, se quisermos que a impressão se efectue, por exemplo, do «pixel 105» para baixo, teremos de indicar o valor «70» (175 — 105 = 70), como acontece na linha 50 do programa.

— *Variável «a»:* esta variável define a «altura» dos caracteres e o seu valor é multiplicado por «8»: assim, se «a» valer «1», o carácter terá «8» pixels» de altura, ou seja, a altura normal; se «a» contiver o valor «4», o carácter terá (8 × 4) «32» pixels», isto é, «4 vezes a altura normal»: na linha 50 foi dado o valor «4» a «a».

— *Variável «l»:* esta variável define a «largura» do carácter, pelo mesmo princípio da anterior.

— *Variável «e»:* esta variável define o «espaço» (intervalo) entre caracteres. Funciona dentro dos mesmos princípios; deste modo, o valor «8» corresponde ao espaçamento normal, enquanto o valor «16» corresponderia ao dobro do espaçamento.

Uma vez definida a cadeia de caracteres (colocada em «g$») e os valores para as variáveis, a sub–rotina 10 encarrega–se de imprimir o conteúdo de «g$», na posição e com as dimensões por nós indicadas. Resta dizer que a posição «x» (início da impressão) é calculada de modo a colocar a cadeia de caracteres a meio do écran e em função da largura e do espaçamento indicados.

Mais adiante indicaremos uma variante à sub–rotina 10, permitindo colocar o início da cadeia em qualquer ponto do écran.

Chegámos à altura de o leitor introduzir o programa na máquina, procedendo por duas fases:

a) Introduzir o programa e gravá–lo com o comando directo SAVE «ENSAIO 1» ;

b) Preparar a cassete onde tem gravada a rotina «CATGIG»: fazer «GO TO 9100» e carregar os bytes da rotina;

isto, porque, se existir algum «gato» nas instruções, o programa pode entrar em «crash» e, como sabe, a única forma de nos libertarmos de tal situação consiste em desligar o computador, perdendo–se todo o trabalho. Se isso acontecer após ter carregado o código–máquina, carregue de novo o BASIC e corrija os erros.

Tudo correu bem: o programa arrancou e apresenta uma frase que enche o écran: é a primeira demonstração da rotina «CATGIG». Premida uma tecla, temos as instruções para o utilizador e o início dos pedidos para os valores das variáveis. Sugerimos ao leitor que comece os seus ensaios com os seguintes valores:

	Ensaio 1	Ensaio 2	Ensaio 3	Ensaio 4	Ensaio 5	Ensaio 6
y =	80	0	80	20	60	0
a =	1	20	4	4	10	20
l =	1	1	2	4	6	6
e =	8	10	10	10	9	9

Como o leitor pode verificar, no Ensaio 1 os caracteres são impressos no tamanho normal, visto que «a» = 1 e «l» = 1. No segundo ensaio, os caracteres aparecem com a máxima altura e mínima largura (160 × 8 pixels). Os Ensaios 3 e 4 proporcionam-nos versões equilibradas, enquanto os dois últimos mostram o efeito do aumento da altura com a máxima largura.

Aliás, numa cadeia de quatro caracteres como é o caso de «ROBOT», o valor de «l» está limitado a 6, razão pela qual condicionámos as entradas a este valor na linha 135. Do mesmo modo, limitámos a 20 o valor de «a» (linha 125), a fim de evitar que a impressão fosse além da linha 22.

Como último ensaio, propomos ao leitor que altere os limites superiores das linhas 115, 125 e 135 e os limites inferior e superior da linha 145: atribua às variáveis os valores que entender e veja os resultados. Altere o limite inferior da linha 125 para «0»: introduza o valor 0 na largura e quaisquer outros nas restantes variáveis: como seria previsível, a rotina não reconhece caracteres com «0 pixels de largura».

Conserve o programa na máquina, pois o seguinte tem um mínimo de alterações, que o leitor poderá introduzir de imediato:

PROGRAMA 3/3

```
   1 REM ENSAIO CATGIG COM                ABECEDARIO 2
   5 PAPER 6: INK 1: BORDER 6: CLS : GO TO 50
  10 LET w=23306: POKE w,(256-e*l*LEN g$)/2
  15 POKE w+1,y: POKE w+2,l: POKE w+3,a: POKE w+4,e
  20 LET w=w+4: FOR f=1 TO LEN g$: POKE w+f,CODE g$(f): N
EXT f
  25 POKE w+f,255: RANDOMIZE USR 45000
  30 RETURN
  50 GO SUB 800: LET g$="ROTINA CATGIG": LET y=10: LET a=
6: LET l=2: LET e=9: GO SUB 10: LET g$="com": LET y=80:
```



```
ET a=2: LET l=4: LET e=8: GO SUB 10
  55 LET g$="ABECED£RIO 2": LET y=124: LET a=4: LET l=2:
ET e=10: GO SUB 10
  60 GO SUB 1000
  70 CLS : PRINT  BRIGHT 1;" Utiliza'^o  da rotina ~CATGI
G~"
  80 PRINT '"Vamos ensaiar a rotina em C%digoM#quina ~CA
TGIG~, atribuindo di-ferentes valores @s vari#veis decoma
ndo, que s^o as seguintes:"
  90 PRINT '"~y~- posi'^o em altura"'"~a~- altura das l
etras"'"~l~- largura das letras"'"~e~- espa'o entre let
ras"
 100 PRINT '"A posi'^o  em largura $    dada au-tomaticam
ente."'TAB 6;"A palavra $ ~ROBOT~."
 110 BEEP .2,30: INPUT "Valor para a posi'^o em altura  =
~y~ entre 0 e 150 ";y
 115 IF y<0 OR y>150 THEN  GO TO 120
 120 BEEP .2,30: INPUT "Valor para a altura das letras  =
~a~ entre 1 e 20 ";a
 125 IF a<1 OR a>20 THEN  GO TO 120
 130 BEEP .2,30: INPUT "Valor para a largura das letras =
~l~ entre 1 e 6 ";l
 135 IF l<1 OR l>6 THEN  GO TO 130
 140 BEEP .2,30: INPUT "Valor para o espa'o entre letras
~e~ entre 8 e 16 ";e
 145 IF e<8 OR e>16 THEN  GO TO 140
 150 CLS : LET g$="ROBOT": GO SUB 10
 160 GO SUB 1000
 170 CLS : PRINT "Indicou os seguintes valores:";AT
8,0;"P osi'^o ~y~:";y'"Altura  ~a~:";a'"Largura ~l~:";l
'"Espa'o  ~e~:";e
 180 GO SUB 900: PRINT #0;TAB 7; BRIGHT 1;"Novo ensaio ?
(s/n)": BEEP .3,20
 190 IF INKEY$<>"s" AND INKEY$<>"S" AND INKEY$<>"n" AND
INKEY$<>"N" THEN  GO TO 190
 200 IF INKEY$="s" OR INKEY$="S" THEN  CLS : GO SUB 800:
GO TO 110
 210 STOP
 800 POKE 23606,212: POKE 23607,247: RETURN
 900 POKE 23606,0: POKE 23607,60: RETURN
1000 PRINT #0; INVERSE 1;"         Prima qualquer tecla
    ": BEEP .05,30: BEEP .05,40: PAUSE 0: RETURN
9000 SAVE "ENSAIO 2" LINE 9100
9010 SAVE "CATGIG"CODE 45000,300: SAVE "ABEC 2"CODE 637
00,768
9020 VERIFY "": VERIFY ""CODE : VERIFY ""CODE : STOP
```

```
9100 CLEAR 44999: LOAD "CATGIG"CODE 45000,300
9110 LOAD "ABEC 2"CODE 63700,768: RUN
```

O programa 3/3 destina-se a permitir os mesmos ensaios do anterior, mas desta vez com os caracteres do ABECEDÁRIO 2: esperamos que o leitor tenha seguido o nosso conselho e conservado este abecedário gravado no endereço 63700. Sendo assim, introduza as alterações. Chamamos a sua atenção para os seguintes pontos:

Linhas 50/55 — Chamada da sub-rotina 800, que coloca a variável «CHARS» no endereço 63700; a página de apresentação tem agora três cadeias e foram definidos valores diferentes para as variáveis de comando: repare-se que cada cadeia é definida separadamente, com uma chamada à sub-rotina 10.

Linhas 70/100 — O mesmo texto do programa anterior, mas agora com os caracteres dos acentos.

Linhas 110/145 — Introdução dos caracteres dos acentos e eliminação dos parênteses, dado estes corresponderem a acentos.

Linha 170 — Introdução dos acentos.

Linhas 800/900 — Sub-rotinas já nossas conhecidas.

Linhas 9000/9110 — Introdução na linha 9010 da instrução para gravação dos bytes do ABECEDÁRIO 2 no endereço 63700; na linha 9020, introdução de mais uma instrução para verificação dos bytes do abecedário; criação da nova linha 9110, para carregamento dos bytes «ABEC 2».

O leitor pode efectuar, agora, ensaios semelhantes aos sugeridos por nós para o programa anterior. *Mas por que não criar as suas próprias cadeias de caracteres e modificar a linha 150?*

No final, não se esqueça de gravar o programa com «GO TO 9000», numa zona bem localizada da sua cassete.

Passemos ao último programa desta série de ensaios da rotina «CATGIG»: os resultados não deixarão de surpreendê-lo e, certamente, diverti-lo.

PROGRAMA 4/3

```
  1 REM ENSAIO 3 CATGIG COM     GRAFICOS ROBOT     ABECED
ARIO 2
  3 GO SUB 2000
  5 PAPER 7: INK 0: BORDER 7: CLS : GO TO 50
 10 LET w=23306: POKE w,(256-e*l*LEN g$)/2
 15 POKE w+1,y: POKE w+2,l: POKE w+3,a: POKE w+4,e
 20 LET w=w+4: FOR f=1 TO LEN g$: POKE w+f,CODE g$(f): N
EXT f
 25 POKE w+f,255: RANDOMIZE USR 45000
 30 RETURN
 50 GO SUB 800: LET g$="ROTINA CATGIG": LET y=0: LET a=
0: LET l=2: LET e=9: GO SUB 10: LET g$="com": LET y=60: L
ET a=2: LET l=4: LET e=8: GO SUB 10
 55 LET g$="ABECED`RIO 2": LET y=90: LET a=4: LET l=2: L
ET e=10: GO SUB 10: LET g$="e": LET y=124: GO SUB 10: LET
g$="Gr#ficos ROBOT": LET y=160: LET a=2: LET l=2: LET
e=8: GO SUB 10
 60 GO SUB 1000
 70 CLS : PRINT  BRIGHT 1;" Utiliza'^o  da rotina ~CATG
IG~"
 80 PRINT '''"Vamos ensaiar a rotina em C%digoM#quina ~CA
TGIG~,em conjunto comos gr#ficos ROBOT. Dadas  as ca-ract
er)sticas  da  figura, ser^ofixados os seguintes valores
:"
 85 PRINT '''"~y~= 23"'''"~e~= 8"
 90 GO SUB 700: PRINT AT 10,22;a$;AT 11,21;b$;AT 12,21;c
$;AT 13,22;d$;AT 14,22;e$;AT 15,22;f$;AT 16,22;h$;AT 17,2
2;i$;AT 18,23;j$;AT 19,23;k$: GO SUB 800
 100 PRINT AT 18,0;"A posi'^o em largura"'"$     automati
ca."
 120 BEEP .2,30: INPUT "Valor da altura dos gr#ficos    -
~a~ entre 1 e 2 ";aa
 125 IF aa<1 OR aa>2 THEN  GO TO 120
 130 BEEP .2,30: INPUT "Valor da largura dos gr#ficos   -
~l~ entre 1 e 5 ";ll
 135 IF ll<1 OR ll>5 THEN  GO TO 130
 150 CLS : GO TO 500
 160 GO SUB 1000
 170 CLS : PRINT "Indicou os seguintes valores:";AT 8,0;"
Posi'^o  ~y~:23"'"Altura   ~a~:";aa'"Largura  ~l~:";ll''
"Espa'o  ~e~:8"
 180 GO SUB 900: PRINT #0;TAB 7; BRIGHT 1;"Novo ensaio ?
(s/n)": BEEP .3,20
 190 IF INKEY$<>"s" AND INKEY$<>"S" AND INKEY$<>"n" AND
INKEY$<>"N" THEN  GO TO 190
 200 IF INKEY$="s" OR INKEY$="S" THEN  CLS : GO SUB 800:
GO TO 120
 210 STOP
 500 GO SUB 700: LET y=23: LET a=aa: LET l=ll: LET e=8:
LET g$=a$: GO SUB 10: LET g$=b$: LET y=y+8*aa: GO SUB 10
 510 LET g$=c$: LET y=y+8*aa: GO SUB 10: LET g$=d$: LET
```

```
y=y+8*aa: GO SUB 10
 520 LET g$=e$: LET y=y+8*aa: GO SUB 10: LET g$=f$: LET y
=y+8*aa: GO SUB 10
 530 LET g$=h$: LET y=y+8*aa: GO SUB 10: LET g$=i$: LET y
=y+8*aa: GO SUB 10
 540 LET g$=j$: LET y=y+8*aa: GO SUB 10: LET g$=k$: LET y
=y+8*aa: GO SUB 10
 550 GO SUB 800: LET g$="A CABE A DO ROBOT": LET y=0: LET
a =2: LET l=1: LET e=8: GO SUB 10
 600 GO SUB 1000
 650 GO TO 170
 700 POKE 23606,148: POKE 23607,241: RETURN
 800 POKE 23606,212: POKE 23607,247: RETURN
 900 POKE 23606,0: POKE 23607,60: RETURN
1000 PRINT #0; INVERSE 1;"      Prima qualquer tecla    "
: BEEP .05,30: BEEP .05,40: PAUSE 0: RETURN
2000 LET a$="!""#$%": LET b$="&'()*+,": LET c$="-./0123"
2010 LET d$="45678": LET e$="9 : ;": LET f$="<=>?@"
2020 LET h$="ABCDE": LET i$="FG HI": LET j$="JKL": LET k$
="M N"
2030 RETURN
9000 SAVE "ENSAIO 3" LINE 9100
9010 SAVE "CATGIG"CODE 45000,300: SAVE "ROBOT"CODE 621
00,768: SAVE "ABEC 2"CODE 63700,768
9020 VERIFY "": FOR f=1 TO 3: VERIFY ""CODE : NEXT f: S
TOP
9100 CLEAR 44999: LOAD "CATGIG"CODE 45000,300
9110 LOAD "ROBOT"CODE 62100,768
9120 LOAD "ABEC 2" CODE 63700,768
9130 RUN
```

Eis algumas das imagens proporcionadas por este programa:

Figura 1/3

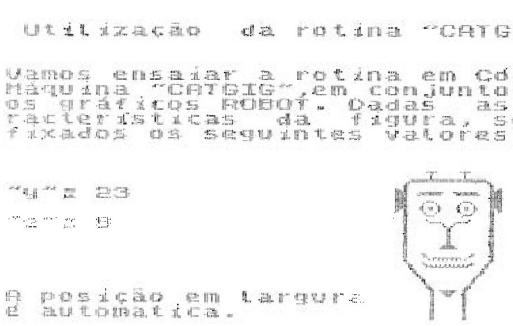


Figura 2/3

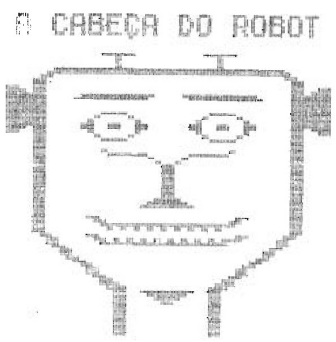


Figura 3/3

As imagens são, sem dúvida, sugestivas — mas tenha o leitor paciência: vamos primeiro analisar o programa:

Linha 3 — Chamada à sub-rotina 2000, onde são criadas as variáveis que conterão as cadeias dos gráficos.

Linhas 50/55 — A página de apresentação tem, agora, cinco cadeias de caracteres do ABECEDÁRIO 2.

Linhas 70/80 — Texto modificado em consonância com a finalidade do programa.

Linha 85 — Os valores das variáveis «y» e «e» são previamente fixados: «y = 23», para que o início da impressão vertical se localize no «pixel 152» (175 — 23 = 152), deixando 24 pixels no topo do écran para

a impressão do título; «e = 8»: tratando-se de uma *cadeia de gráficos*, o espaçamento tem de obedecer à própria construção do desenho.

Linha 90 — Chamada à sub-rotina 700, que coloca a variável «CHARS» no endereço 62100, onde estão alojados os gráficos «ROBOT»; impressão da nossa conhecida «cabeça», pelo processo já estudado no programa 5/2.

Linhas 120/135 — Entrada dos valores a atribuir às duas variáveis que definem a altura e a largura dos caracteres: de notar que estas variáveis são, agora, «aa» e «ll», por razões que mais adiante se tornarão evidentes.

Linha 150 — Chamada à rotina 500, através da qual são definidas as cadeias a colocar em «g$» e os valores das variáveis de comando da impressão.

Linha 170 — Informação sobre os valores introduzidos, mas em que as únicas variáveis são «aa» e «ll»: «y» e «e» já têm valores fixados, como vimos na linha 85.

Linhas 500/550 — Nova chamada à sub-rotina 700; iniciação dos valores das variáveis, em que «a» e «l» têm, respectivamente, os valores de «aa» e «ll»; dado a «g$» o conteúdo de «a$» («!""#$%»), o qual constitui a primeira linha de gráficos do nosso «ROBOT»; chamada à sub-rotina 10 para impressão; dado a «g$» o conteúdo de «b$»; incremento do valor de «y» de «8 × aa»: se o leitor indicou o valor 1 para a altura dos caracteres, o incremento será de 23 + 8 = 31, isto é, a impressão da segunda fila de gráficos será feita, exactamente, «8 pixels» abaixo da primeira; se «aa=2», o incremento seria de 23 + 16 = 39, em concordância com a altura dos caracteres, agora com «16 pixels» (não esqueça o leitor que a rotina «CATGIG» inverte a ordenação vertical da rede de pixels).

As restantes linhas funcionam de forma idêntica — «g$» vai tomando o conteúdo das sucessivas filas de gráficos: «y» vai sendo incrementado de «8 × aa» e é chamada a sub-rotina 10.

Linha 700 — Já referida.

Linhas 2000/2030 — Sub-rotina idêntica à do programa 5/2.

Linhas 9000/9020 — Introdução na linha 9010 da instruçao para gravação dos bytes dos gráficos «ROBOT» no endereço 62100; na linha 9020, criação de um ciclo FOR-NEXT «f» para efectuar as três verificações de bytes.

Linhas 9100/9130 — Carregamento dos três conjuntos de bytes e arranque por «RUN». Também aqui poderíamos usar um ciclo: «FOR f = 1 TO 3: LOAD "" CODE: NEXT f»: optámos pela solução mais complicada, a fim de tornar as instruções mais perceptíveis.

É altura de o leitor proceder à introdução do programa e preparar as cassetes onde colocou a rotina «CATGIG», os gráficos «ROBOT» e o «ABEC 2». Proceda como para os anteriores programas: grave primeiro o BASIC com um comando directo. Depois faça «GO TO 9100» e carregue, sucessivamente, os três conjuntos de bytes. O programa arrancará, apresentando a imagem da figura 1/3.

Premindo uma tecla, dê início aos seus ensaios, para os quais sugerimos a adopção dos seguintes valores:

	Ensaio 1	Ensaio 2	Ensaio 3	Ensaio 4	Ensaio 5	Ensaio 6
a =	1	2	2	2	1	2
l =	1	1	2	4	5	5

Considerando as dimensões originais do desenho, fomos obrigados a limitar os valores a atribuir a «a» e a «l»: com efeito, a largura (l) tem 56 pixels (7 gráficos × 8 pixels) e a altura (a) ocupa 80 pixels (10 gráficos × 8 pixels), pelo que não é possível aumentar a altura para mais do dobro e a largura mais de quatro vezes a original. O programa admite o valor «5» para a largura, mas a imagem fica com *menos 12 pixels de cada lado: 56 × 5 = 280, donde mais 24 pixels que os 256 consentidos.*

A figura 3/3 foi obtida com «a=2» e «l=3», enquanto a «cabeça» da figura 2/3 teria, obviamente, «a=1» e «l=1».

Sugerimos ao leitor que faça os seus próprios ensaios, retirando a função às linhas 125 e 135, bastando, para o efeito, introduzir uma instrução «REM» no início das linhas. Deste modo, pode atribuir às variáveis «aa» e «ll» os valores que entender.

Vamos terminar esta secção fornecendo ao leitor uma pequena alteração a introduzir na sub-rotina 10, que lhe facultará o controlo da posição «x» —> início da impressão horizontal:

```
10 LET w = 23306: POKE w,x
```

Cria-se, assim, mais uma variável (x), à qual é necessário atribuir um valor. Esse valor está directamente relacionado com a numeração horizontal da rede de pixels do écran. Introduza mais as seguintes alterações ao programa:

```
100 BEEP .2,30: INPUT «Valor para a posicao em largura = (~x» entre 0 e 255) «;x
```

e crie a linha:

```
105 IF x <0 OR x>255 THEN GO TO 100
```

Torna-se evidente que o valor de «x» terá de ter em conta a dimensão da cadeia de caracteres e a largura e o espaçamento desejados: se

a cadeia «ROBOT» for colocada com «x=120» e «l=4», a impressão *sairá do écran, pois este só comporta 256 pixels:*

```
- ROBOT = 40 pixels (5 caracteres x 8 pixels)
- 40 x 4 = 160 (l = 4)
- 120 + 160 = 280 (x = 120)
```

Deixamos ao critério do leitor a utilização da sub–rotina na sua forma inicial, ou com esta modificação. Mas o leitor tem ainda outra hipótese, a qual lhe permitirá incluir as duas soluções no mesmo programa: construa duas sub–rotinas, uma para a impressão automática a meio do écran, outra para a introdução da variável «x»:

```
 10 LET w = 23306: POKE w, (256 - e x l x LEN g$) / 2:
POKE w + 1,y: POKE w + 2,l: POKE w + 3,a: POKE w + 4,e:
LET w = w + 4: FOR f = 1 TO LEN g$: POKE w + f, CODE
g$(f): NEXT f: POKE w, 255: RANDOMIZE USR 45000: RETURN
 20 LET w = 23306: POKE w,x: POKE w + 1,y: POKE w + 2,l:
POKE w + 3,a: POKE w + 4,e: LET w = w + 4: FOR f = 1 TO
LEN g$: POKE w + f, CODE g$(f): NEXT f: POKE w,255: RANDO-
MIZE USR 45000: RETURN
```

Consoante os casos, chamará uma ou a outra sub–rotina: no caso da sub–rotina 20, terá de definir previamente o valor de «x». O leitor notou, certamente, que colocámos todas as instruções na mesma linha: esta construção traz a vantagem de diminuir o comprimento do programa BASIC e, por outro lado, torna o mais rápido o processamento.

3 — RELOCATAR CATGIG

O endereço 45000, para o qual está preparada a rotina «CATGIG», serviu perfeitamente para os programas de demonstração e ensaio que o leitor estudou e executou na secção anterior.

Admitamos que tem um extenso programa BASIC com cerca de 25 K de comprimento (25 600 bytes), no qual pretende aplicar a rotina «CATGIG»: não pode fazê–lo no endereço 45000, como se demonstra seguidamente:

```
- Endereço inicial do BASIC: 23755
- Comprimento do programa: 25600 bytes
- Endereço final do BASIC: 49360 (23755 + 25600 = 493
55 -> 49360)
```


Se o leitor carregar o seu programa na máquina e seguidamente fizer «CLEAR 49999», como se fosse para carregar a rotina «CATGIG», receberá a mensagem «M RAMTOP no good 0 : 1», significando que *«o topo da RAM não está em condições para receber esse endereço»*. O computador teria toda a razão, pois o programa colocou a «RAMTOP» em 49360.

Se, em vez de utilizar a instrução «CLEAR», o leitor carregasse directamente a rotina fazendo «LOAD "" CODE 45000» e depois listasse o programa, teria o desgosto de verificar que grande número de linhas tinha sido apagado, *tantas quantas ocupavam os últimos 4360 bytes do programa: 49360 — 45000 = 4360.*

Mas, se tivesse feito «CLEAR 49359», receberia a mensagem «0 OK,0 : 1», significando que o computador *aceitaria carregar qualquer rotina ou grupo de bytes a partir do endereço 49360, sem perturbar o programa BASIC.*

Poderíamos daqui concluir que a nossa rotina «CATGIG» deveria ser colocada num endereço superior a 49360. Mas não nos precipitemos: o futuro endereço para a rotina «CATGIG» terá de ser calculado tendo em conta determinados eventos previsíveis e mensuráveis — e outros, prováveis, mas de difícil avaliação. Vejamos:

a) Com a aplicação da rotina «CATGIG», certamente que o leitor vai criar mais uma série de linhas e, mesmo, acrescentar instruções a linhas já existentes: no final, o programa ficará com um comprimento superior ao inicial, digamos, por hipótese, com mais 2 K (2048 bytes), isto é, 27650 (25600 + 2048 = 27648 —> arredondando: 27650);

b) Natural será que o leitor queira utilizar o método para criação de gráficos, aprendido no capítulo II: um conjunto de caracteres ocupa 768 bytes;

c) Por que não dar uma «nota profissional» às suas páginas de texto, utilizando dois tipos de letras acentuadas?: dois conjuntos de caracteres ocupam 1536 bytes.

Considerando verdadeiros estes pressupostos, poderíamos dizer que o endereço aconselhado para a colocação da rotina seria, arredondando, 51500. Com efeito:

```
49360 + 2048 = 51408 -> arredondando: 51450
```

Continuando o nosso raciocínio e dentro dos mesmos princípios, o leitor poderia, agora, construir uma «tabela» de endereços para o seu programa e grupos de bytes:

	End. inicial	Comprimento	End. final
- Programa BASIC:	23755	27650	51450
- Rotina «CATGIG»:	51500	300	51800

	End. inicial	Comprimento	End. final
– Bytes «GRÁFICOS»:	62100	768	62868
– Bytes «ABEC 2»:	63700	768	64468
– Bytes «ABEC 1»:	64500	768	65268

Tudo isto parece correcto, mas um observador mais atento diria que a tabela carecia de «lógica», pois, além de não aproveitar toda a área disponível para o BASIC, se quiséssemos carregar os bytes num único grupo (o que não só é possível como desejável), teríamos um enorme espaço inútil entre o início do «CATGIG» e o final do «ABEC 1». Com efeito:

– Endereço final do «ABEC 1»: 65268.
– Endereço inicial do «CATGIG»: 51500.
– *Diferença em bytes: 13768.*
– Comprimento útil dos bytes: 2604 (300 + (3× 768).
– *Espaço inútil em bytes: 11164 (10,9 K).*

Torna–se, portanto, necessário que o leitor reveja os seus cálculos, com a preocupação de eliminar os «vazios» e deixar o máximo possível de espaço para o BASIC. Assim, vamos raciocinar numa outra base, isto é, partiremos do endereço 65368 (início dos «UDG») e calcular os restantes endereços «a descer»:

– Endereço de início dos «UDGs»: 65368
– Comprimento de «CATGIG»: 300 bytes
– *Início de «CATGIG»: 65068*
– Comprimento de «GRÁFICOS»: 768 bytes
– *Início de «GRÁFICOS»: 64300*
– Comprimento de «ABEC 1»: 768 bytes
– *Início de «ABEC 1»: 63532*
– Comprimento de «ABEC 2»: 768 bytes
– *Início do «ABEC 2»: 62764*
– Diferença entre início de «ABEC 2» e início de «UDG»: 2604 (65368 – 62764 = 2604)
– *Área livre para BASIC: 62764 – 23755 = 39009 (38 K)*

Como o seu programa ocupa 27650 bytes, teria *uma área de expansão de 11359 bytes (39009 — 27650 =11359),* a qual tanto poderia ser utilizada para BASIC como para alojar mais um conjunto de gráficos ou um terceiro abecedário, ou ainda diversas rotinas em código–máquina.

Só temos uma observação a fazer relativamente à sua tabela: não é cómodo trabalhar com grupos de bytes cujos endereços não estejam

arredondados, nem seria prudente colocar os grupos de bytes em endereços «encostados», isto é, o último byte de um grupo ser imediatamente seguido pelo primeiro byte útil de outro grupo.

Com o «desperdício» de alguns bytes, propomos ao leitor a seguinte tabela final, com os endereços «arredondados»:

	End. inicial	Comprimento	End. final
– Bytes «UDG»:	65368	168	65536
Programa BASIC:	23755	27650	51450
Rotina «CATGIG»:	*65060*	*300*	*65360*
Bytes «GRÁFICOS»:	*64290*	*768*	*6058*
Bytes «ABEC 2»:	*63520*	*768*	*64288*
Bytes «ABEC 1»:	*62750*	*768*	*63518*

Com estes arredondamentos desperdiçámos «14» bytes:

```
65368 – 62750 = 2618
2618 – 2604 = 14
```

o que não tem qualquer significado na área de expansão de BASIC, a qual passará a ter 11345 bytes:

```
62750 – 23755 = 38995
38995 – 27650 = 11345
```

O leitor terá compreendido que todo este exercício não se destinou, somente, a determinar o novo endereço para alojar a rotina «CATGIG», mas, e muito especialmente, a proporcionar–lhe um método de trabalho para cálculo de endereços onde alojar qualquer grupo de bytes. Como exercício complementar, sugerimos que o leitor calcule novos endereços, agora numa ordem diferente, isto é, a rotina «CATGIG» no topo e os «GRÁFICOS» na base.

Retomando o tema «RELOCATAR CATGIG», segue–se a listagem do programa «RELOCTGIG», especialmente concebido para deslocar a rotina «CATGIG» do endereço 45000 para qualquer outro endereço.

Não entraremos em pormenores sobre a sub–rotina 500, destinada a introduzir alterações ao código–máquina que permitirão à rotina trabalhar noutro endereço: invocamos, para isso, as mesmas razões dadas a respeito da sub–rotina 10 do programa 2/3.

PROGRAMA 5/3

```
  1 REM RELOCATAR CATGIG          COPYRIGHT R.CASQUILHO
1986
  3 LET end3=45000: CLS : LET g$="RELOCATAR CATGIG": LET
y=80: LET a=3: LET l=2: LET e=8: GO SUB 10
  5 GO TO 20
 10 LET w=23306: POKE w,(256-e*l*LEN g$)/2: POKE w+1,y:
POKE w+2,l: POKE w+3,a: POKE w+4,e: LET w=w+4: FOR f=1 TO
LEN g$: POKE w+f,CODE g$(f): NEXT f: POKE w+f,255: RANDOM
IZE USR end3: RETURN
 20 PRINT AT 18,5; BRIGHT 1;"Indicar novo endereco"
 30 BEEP .3,30: INPUT "Novo Endereco ? ";end2
 40 LET end1=45000: LET end3=end2
 45 CLS : PRINT AT 11,12; FLASH 1;"AGUARDAR"
 50 FOR f=1 TO 300
 60 POKE end2,PEEK end1
 70 LET end1=end1+1: LET end2=end2+1
 80 NEXT f
 85 GO SUB 500: LET w$="CATGIG"
 90 CLS : LET g$="ROTINA": LET y=40: LET a=2: LET l=2: G
O SUB 10: LET g$=w$: LET y=65: LET l=3: GO SUB 10: LET g$
="RELOCATADA": LET y=90: LET l=2: GO SUB 10
100 PRINT AT 17,0; BRIGHT 1;" Gravar ";w$;" no endereco:
";end3
110 PRINT '""Prepare a cassettee o gravador:        Prim
a ENTER": PAUSE 0
120 SAVE w$CODE end3,300
130 CLS : PRINT AT 11,5; FLASH 1;"Rebobinar e verificar"
: BEEP .5,20
140 VERIFY w$CODE end3,300
150 STOP
500 DIM p(5): RESTORE 505: FOR f=1 TO 5
505 DATA 3,32,164,48,156
510 READ x: LET p(f)=end3+x: NEXT f
515 DIM q(5): RESTORE 520: FOR f=1 TO 5
520 DATA 86,106,127,154,251
525 READ y: LET q(f)=end3+y: NEXT f
530 DIM u(5): FOR f=1 TO 5
535 LET u(f)=p(f)-256
540 NEXT f
545 DIM v(5): DIM z(5): FOR f=1 TO 5
550 LET v(f)=u(f)-256*INT (u(f)/256)
555 LET z(f)=1+(INT (u(f)/256))
560 NEXT f
565 FOR f=1 TO 5
570 POKE q(f),v(f): POKE q(f)+1,z(f)
575 NEXT f
580 RETURN
9000 CLEAR : SAVE "RELOCTGIG" LINE 9100
9010 SAVE "CATGIG"CODE 45000,300
9020 VERIFY "": VERIFY ""CODE : STOP
9100 CLEAR 44999: LOAD ""CODE : RUN
```


Como funciona

Linha 3 — Atribuído à variável «end3» o valor 45000; limpeza do écran e impressão da cadeia contida em «g$», com chamada da sub–rotina 10.

Linha 5 — Passagem do comando à linha 20.

Linha 10 — Sub–rotina já nossa conhecida.

Linha 20 — Mensagem avisando para o pedido do novo endereço.

Linha 30 — Pedido para entrada do novo endereço, cujo valor é dado à variável «end2».

Linha 40 — Criação da variável «end1», à qual é dado o valor 45000; dado o valor do novo endereço, contido em «end2», à variável «end3».

Linha 45 — Limpeza do écran; impressão da mensagem «AGUARDAR», a qual permanece durante 9 segundos, tempo necessário para a cópia dos bytes para o novo endereço e processamento das alterações introduzidas pela sub–rotina 500.

Linhas 50/80 — Rotina que processa a cópia dos bytes do endereço 45000 (colocado em «end1»), para o novo endereço (colocado em «end3»): esta rotina serve para copiar qualquer grupo de bytes de um endereço para outro, bastando atribuir os valores desejados às variáveis «end1» e «end3» e dar ao ciclo FOR–NEXT «f» o comprimento do grupo de bytes.

Linha 85 — Passagem à sub–rotina 500; atribuição da cadeia «CATGIG» à variável «w$»: esta variável irá ser utilizada nas linhas 90,120 e 140.

Linha 90 — Após o processamento pela sub–rotina 500, o comando passa à linha 90, que imprime a mensagem «ROTINA RELOCATADA».

Linhas 100/140 — Indicações e instruções para gravação da rotina no novo endereço.

Linha 150 — Após gravação e verificação, o programa pára: para novo arranque usar «RUN» ou GO TO 1.

Linhas 500/580 — Sub–rotina já referida na linha 85.

Linhas 9000/9100 — Instruções para gravação do programa BASIC, dos bytes da rotina «CATGIG» no endereço 45000 e respecti-

vas verificações; a linha 9100 prepara a «RAMTOP» para receber o endereço 45000 e carrega os bytes com «RUN».

Como habitualmente, introduza o programa e grave o BASIC com o comando directo SAVE «RELOCTGIG». Prepare a cassete onde gravou a rotina «CATGIG», faça «GO TO 9100» e carregue os bytes: o programa arrancará com a mensagem da linha 3 e o pedido de introdução do novo endereço: indique 65060, conforme a tabela.

Após o carregamento, grave a rotina com o novo endereço na sua cassete de bytes e grave o programa com a *rotina original*, noutra cassete: repare que a rotina original do endereço 45000 não foi destruída durante o processamento para o novo endereço: isto só aconteceria se o leitor tivesse indicado, como novo endereço, por hipótese, 44900 ou 45100. Nesse caso haveria uma sobreposição de endereços e, naturalmente, a instrução da linha 9010 gravaria 300 bytes do endereço 45000, mas estes bytes já não corresponderiam aos originais.

4 — JUNTAR BYTES

O processo utilizado no programa 4/3, onde são aplicados três grupos de bytes, é válido, funciona, mas carece de profissionalismo e elegância. O leitor tem agora à sua disposição os meios de suprir tal carência: calculou e realizou uma tabela com todos os endereços dos diferentes grupos de bytes a aplicar nos seus programa BASIC.

A rotina «CATGIG» funciona já no endereço 65060: temos a confirmação deste facto pela impressão da mensagem da linha 90 do programa 5/3.

Resta–nos «relocatar» os bytes de «GRÁFICOS», «ABEC 1» e «ABEC 2» nos novos endereços da tabela e proceder à sua *«junção»*, isto é, criar um único grupo de bytes com início no endereço 62750 e com o comprimento de 2610 bytes (65360 — 62750 = 2610), isto é, *uma rotina de bytes que engloba, neste caso, desde o «ABEC 1» até à rotina «CATGIG» e que se carrega e grava num único bloco.*

Existem diferentes métodos para relocatar e juntar grupos de bytes.

Vamos indicar–lhe um programa capaz de realizar estas operações, intitulado «RELOCJUNTA» e que servirá ao leitor sempre que precise de relocatar uma ou mais rotinas ou grupos de bytes e, se forem mais de um grupo, gravá–los num só bloco.

Para as rotinas em código–máquina «não relocatáveis», como é o caso da «CATGIG», estas terão primeiro de ser tratadas em programas específicos e só depois integradas no bloco «já relocatado».

PROGRAMA 6/3

```
  1 REM RELOCATAR E JUNTAR          COPYRIGHT R.CASQUIL
HO 1986
 10 CLS : PRINT "Indique numero de rotinas e gru-pos de
bytes a relocatar."''"Seguidamente,o endereco inicial,o
novo endereco e o comprimento de cada grupo de bytes."''
"COMECE PELO GRUPO CUJO NOVO EN- DERECO SEJA O MAIS BAIX
O:"
 20 BEEP .2,30: INPUT "Numero de grupos ? ";n
 30 DIM a(n): DIM b(n): DIM c(n): FOR f=1 TO n
 40 BEEP .2,30: INPUT "Endereco inicial do grupo ";(f);"
";a(f): INPUT "Novo endereco do grupo ";(f);" ";b(f): INP
UT "Comprimento do grupo ";(f);" ";c(f)
 50 NEXT f
 55 GO SUB 500
 60 CLS : PRINT AT 5,12; FLASH 1;"AGUARDAR": PRINT
 70 FOR f=1 TO n
 80 FOR g=0 TO c(f)-1
 90 POKE (b(f)+g),PEEK (a(f)+g)
100 NEXT g
110 PRINT '"Grupo ";f;" relocatado: End.:";b(f): BEEP .3
,40: BEEP .1,20
120 NEXT f
130 LET tot=(b(n)+c(n))-b(1)
140 GO SUB 700
150 GO TO 800
500 CLS : PRINT "Prepare o gravador e a cassette onde te
m os bytes a relocatar."''"Carregue cada grupo seguindo
as indicacoes do computador:"
510 FOR f=1 TO n
520 PRINT '"Vai carregar o grupo ";f;'"do endereco: ";a
(f)'"com ";c(f);" bytes"
530 PRINT '"Prima ENTER quando estiver pre- parado e li
que o gravador:"
540 PAUSE 0: LOAD ""CODE a(f),c(f)
550 CLS : NEXT f: RETURN
700 CLS : PRINT "Prepare o gravador e a cassette onde va
i gravar os bytes, juntosnum so grupo com ";tot;" bytes"'
"a partir do endereco ";b(1)
710 PRINT '"Depois da verificacao serao in- dicados os
bytes equivalentes decada endereco."''"Indique nome para
os bytes:"
720 BEEP .2,30: INPUT "Nome ? ";w$: IF LEN w$>10 THEN P
RINT '"Nome com mais de 10 caracteres: CORRIJA": BEEP .3,
10: PAUSE 100: GO TO 720
```

```
 730 SAVE w$CODE b(1),tot
 740 CLS : PRINT AT 11,10;"VERIFICAR"''"Rebobine a casset
te e ligue o            gravador"
 750 VERIFY ""CODE
 760 RETURN
 800 CLS : PRINT "Tome nota dos bytes equivalentesdos
enderecos iniciais de cada  grupo de bytes:"
 810 DIM d(n): DIM e(n): FOR f=1 TO n
 820 LET d(f)=b(f)-256*INT (b(f)/256)
 830 LET e(f)=(INT (b(f)/256))-1
 840 NEXT f
 850 FOR f=1 TO n
 860 PRINT 'b(f)'"POKE 23606,";d(f);" : POKE 23607,";e(f
)
 870 NEXT f
 880 STOP
9000 CLEAR : SAVE "RELOCJUNTA" LINE 1
9010 VERIFY "": STOP
```

Que faz o programa

O programa «RELOCJUNTA» começa por apresentar-nos uma página de instruções, das quais salientamos a mais importante: «COMECE PELO GRUPO CUJO ENDEREÇO SEJA O MAIS BAIXO». Isto significa que, se o leitor tiver três grupos de bytes, cujos endereços «novos» já determinou (na sua tabela), deverá começar pela indicação dos dados do «ABEC 1», pois é este grupo que *vai ter o endereço mais baixo.*

A entrada dos restantes grupos deverá, logicamente, obedecer ao valor numérico dos respectivos endereços. Entretanto, o computador já tinha inquirido sobre o quantitativo de grupos a relocatar: a resposta (no nosso caso) seria «3», valor que ficará atribuído à variável «n».

As perguntas seguintes respeitam a cada grupo de bytes, e cobrem o endereço inicial (onde os bytes se encontram gravados), o novo endereço e o comprimento (conforme tabela). O quadro seguinte é destinado a facilitar, ao leitor, a entrada dos dados:

	End. inicial «a(n)»	Comprimento «c(n)»	Novo End. «b(n)»
Grupo 1:	64500	768	62750
Grupo 2:	63700	768	63520
Grupo 3:	62100	768	64290

Os quadros DIM «a(n)», «b(n)» e «c(n)» conterão, respectivamente, os valores do endereço inicial, novo endereço e comprimento. Os quadros ficam dimensionados em função do quantitativo de grupos, isto é, do valor contido em «n». Assim, «a(1)» terá o valor 64500 e «b(1)» o valor 62750. Lembramos que as instruções «RUN» e «CLEAR» apagam quaisquer valores contidos em quadros «DIM» ou variáveis, pelo que, se o programa parar, use «GO TO 10» para novo arranque (rotina 20/50).

Após a entrada dos dados relativos a cada grupo, o programa pede o carregamento dos bytes respectivos, pela ordem de entrada, com a mensagem: *«Vai carregar o grupo (f) do endereço (a(f)) com (c(f)) bytes»,* visto o carregamento estar inserido num ciclo FOR–NEXT (f), cujo comprimento é «n». O leitor deverá ter preparada a cassete onde gravou os abecedários e os gráficos «ROBOT», para o solicitado carregamento (sub–rotina 500/550).

Terminado o carregamento, aparece a mensagem «AGUARDAR», entrando em funcionamento a rotina 70/120. Esta rotina é interessante, pois vai «relocatar», separadamente, cada grupo de bytes, imprimindo, no final de cada operação, uma mensagem: «o grupo (f) foi relocatado no endereço (b(f)», com um aviso sonoro. Este aparente preciosismo torna–se importante, quando o utilizador espera, olhando para o écran, onde só a palavra «AGUARDAR» se apresenta: constitui um «relatório periódico de actividade» que a máquina, gentilmente, fornece ao utilizador.

Chamamos a atenção do leitor para a linha 130, cujas instruções calculam o total de bytes do «bloco» a gravar. Note–se que o total não é o somatório dos comprimentos de cada grupo de bytes, mas, antes, a soma do último novo endereço com o respectivo comprimento, menos o valor do primeiro novo endereço. Concretizemos:

```
Último novo endereço: 64290 «b(n)» -> «b(3)»
Comprimento: +768 «c(n)» -> «c(3)»
Resultado: 65058
Primeiro novo endereço: -62750 «b(1)»
TOTAL: 2308 «tot»
```

Deste modo, o total contém os 4 bytes «desperdiçados» entre os endereços 62750 e 65058 (ver tabela), o que não aconteceria se o total fosse a *soma dos comprimentos dos três grupos: 2304 (3 ¥768).* O valor 2308 ficará contido na variável «tot», que irá ser utilizada na sub–rotina 700.

O programa passa à sub–rotina 700/760, onde se encontram as instruções destinadas à gravação dos grupos de bytes, mas agora num «bloco» de «tot» bytes, a partir do endereço «b(1)»: no nosso caso, será um «bloco» de 2308 bytes colocados no endereço 62750. Repare–se na instrução condicional da linha 720, impedindo a entrada de «nomes» com mais de dez caracteres e na mensagem que, nessa eventualidade, é impressa.

Segue–se uma rotina muito importante para o leitor (800/880), que entra em funcionamento após a verificação. Trata–se do cálculo dos «bytes equivalentes» do *endereço a partir do qual estão os bytes de cada grupo* e da impressão desse endereço, seguida das instruções para aplicação:

```
Endereço 62750 -> POKE 23606,30: POKE 23607,244
Endereço 63520 -> POKE 23606,32:.POKE 23607,247
Endereço 64290 -> POKE 23606,34: POKE 23607,250
```

Serão estes os decimais a aplicar à variável «CHARS» para se obterem os caracteres colocados nos endereços respectivos.

Feita a gravação do seu «bloco» de bytes, resta introduzir a rotina «CATGIG», já relocatada para o endereço 65060. Esta operação será efectuada por comandos directos, sem o auxílio de qualquer programa. Prepare a cassete onde tem gravada a rotina «CATGIG» e rebobine a cassete onde acabou de gravar os 2308 bytes do endereço 62750: fica, assim, preparado para as seguintes acções:

a) Limpe a memória do computador usando o comando directo «PRINT USR 0» ou «RANDOMIZE USR 0» (tem o mesmo efeito que desligar e voltar a ligar a máquina).

b) Dê a instrução «CLEAR 62749» (endereço — 1) e LOAD "" CODE 62750,2308: carregue o bloco;

c) Coloque no gravador a cassete da rotina «CATGIG» e faça LOAD "" CODE 65060,300: carregue a rotina. Tem agora no computador o «bloco» formado pelos bytes «ABEC 1», «ABEC 2» e «GRÁFICOS», mais a rotina «CATGIG»;

d) Use outra cassete (ou a mesma mas numa zona bem definida) e faça SAVE «ROTINA 1» CODE 62750,2610 («ROTINA 1» foi o nome por nós imaginado, mas, obviamente, o leitor escolherá aquele que entender) e grave os bytes.

Vamos terminar este capítulo, fornecendo ao leitor as indicações necessárias para aplicar a «ROTINA 1» em qualquer programa. Para o efeito, vamos socorrer–nos do programa 4/3 — «ENSAIO 3» e introduzir–lhe as alterações decorrentes dessa aplicação, mais algumas por nós imaginadas, com o objectivo de tirar partido da existência de dois abecedários.

Carregue o programa 4/3 com o comando MERGE "": como sabe, este comando é habitualmente usado para «misturar» grupos de linhas, mas, neste caso, vai servir–nos para carregar unicamente a parte BASIC do programa, impedindo a entrada dos bytes que se lhe seguem: rotina «CATGIG», gráficos «ROBOT» e «ABEC 2». Pare o gravador quando começarem os bytes e aguarde o aparecimento da mensagem «0 OK : 1».

«Liste» o programa e introduza–lhe as alterações que constam no programa 7/3, cuja listagem se segue. As alterações dispensam qualquer explicação complementar, dizendo respeito ao *carregamento da «ROTINA 1», gravação do programa em conjunto com a mesma rotina, alteração dos «POKE» na variável «CHARS» com criação de uma nova sub–rotina (850) para o endereço 64290 — «GRÁFICOS» e pequenas modificações que o leitor facilmente compreenderá.*

Contamos que o leitor retire pleno proveito de tudo o que lhe transmitimos sobre a rotina «CATGIG» e manipulação de endereços, conferindo aos seus programas o tal «toque» de profissionalismo que eles merecem.

PROGRAMA 7/3

```
   1 REM ENSAIO 4 CATGIG COM      GRAFICOS ROBOT, ABECEDA
RIO 1          E ABECEDARIO 2
   3 GO SUB 2000
   5 PAPER 7: INK 0: BORDER 7: CLS : GO TO 50
  10 LET w=23306: POKE w,(256-e*l*LEN g$)/2
  15 POKE w+1,y: POKE w+2,l: POKE w+3,a: POKE w+4,e
  20 LET w=w+4: FOR f=1 TO LEN g$: POKE w+f,CODE g$(f): N
EXT f
  25 POKE w+f,255: RANDOMIZE USR 65060: REM Alteracao do
endereco
  30 RETURN
  50 GO SUB 800: LET g$="ROTINA CATGIG": LET y=0: LET a=6
: LET l=2: LET e=9: GO SUB 10: LET g$="com": LET y=60: LE
T a=2: LET l=4: LET e=8: GO SUB 10
  55 LET g$="2 ABECED`RIOS": LET y=90: LET a=4: LET l=2:
LET e=9: GO SUB 10: LET g$="e": LET y=124: GO SUB 10: LET
g$="Gr#ficos ROBOT": LET y=160: LET a=2: LET l=2: LET
e=8: GO SUB 10
  60 GO SUB 1000
  70 CLS : GO SUB 800: PRINT  BRIGHT 1;" Utiliza'^o  da r
otina ~CATGIG~"
  80 GO SUB 700: PRINT '''"Vamos ensaiar a rotina ~CATGIG
~,em conjunto com os 2 abeced#riose os gr#ficos ROBO
T.Dadas as ca~racter)sticas  da  figura, ser^ofixados os
s eguintes valores :"
  85 GO SUB 800: PRINT '''"~y~= 23"''"~e~= 8"
  90 GO SUB 850: PRINT AT 10,22;a$;AT 11,21;b$;AT 12,21;c
$;AT 13,22;d$;AT 14,22;e$;AT 15,22;f$;AT 16,22;h$;AT 17,2
2;i$;AT 18,23;j$;AT 19,23;k$: GO SUB 800
 100 GO SUB 700: PRINT AT 18,0;"A posi'^o em largura"'"$
automatica."
```

```
 120 GO SUB 800: BEEP .2,30: INPUT "Valor da altura dos g
r#ficos     = ~a~ entre 1 e 2 ";aa
 125 IF aa<1 OR aa>2 THEN  GO TO 120
 130 BEEP .2,30: INPUT "Valor da largura dos gr#ficos    =
~l~ entre 1 e 5 ";ll
 135 IF ll<1 OR ll>5 THEN  GO TO 130
 150 CLS : GO TO 500
 160 GO SUB 1000
 170 CLS : GO SUB 800: PRINT "Indicou os seguintes valore
s:";AT 8,0;"Posi'^o ~y~:23"'"Altura  ~a~:";aa'"Largura
~l~:";ll'"Espa'o  ~e~:8"
 180 GO SUB 900: PRINT #0;TAB 7; BRIGHT 1;"Novo ensaio ?
(s/n)": BEEP .3,20
 190 IF INKEY$<>"s" AND INKEY$<>"S" AND INKEY$<>"n" AND I
NKEY$<>"N" THEN  GO TO 190
 200 IF INKEY$="s" OR INKEY$="S" THEN  CLS : GO SUB 800:
GO TO 120
 210 STOP
 499 REM Utilizacao da nova sub-rotina 850 para GRAFICOS
 500 GO SUB 850: LET y=23: LET a=aa: LET l=ll: LET e=8: L
ET g$=a$: GO SUB 10: LET g$=b$: LET y=y+8*aa: GO SUB 10
 510 LET g$=c$: LET y=y+8*aa: GO SUB 10: LET g$=d$: LET y
=y+8*aa: GO SUB 10
 520 LET g$=e$: LET y=y+8*aa: GO SUB 10: LET g$=f$: LET y
=y+8*aa: GO SUB 10
 530 LET g$=h$: LET y=y+8*aa: GO SUB 10: LET g$=i$: LET y
=y+8*aa: GO SUB 10
 540 LET g$=j$: LET y=y+8*aa: GO SUB 10: LET g$=k$: LET y
=y+8*aa: GO SUB 10
 550 GO SUB 800: LET g$="A CABE A DO ROBOT": LET y=0: LET
a =2: LET l=1: LET e=8: GO SUB 10
 600 GO SUB 1000
 650 GO TO 170
 700 POKE 23606,30: POKE 23607,244: RETURN : REM "ABEC 1
"
 800 POKE 23606,32: POKE 23607,247: RETURN : REM "ABEC 2
"
 850 POKE 23606,34: POKE 23607,250: RETURN : REM "GRAFICO
S"
 900 POKE 23606,0: POKE 23607,60: RETURN
1000 GO SUB 900: PRINT #0; INVERSE 1;"      Prima qualque
r tecla     ": BEEP .05,30: BEEP .05,40: PAUSE 0: RETURN
2000 LET a$="!""#$%": LET b$="&UdefY,": LET c$="-.g0h23":
REM Alteracao de caracteres.
2010 LET d$="4o6p8": LET e$="9 : ;": LET f$="<=>?@": REM
Alteracao de caracteres.
```



```
2020 LET h$="ABCDE": LET i$="FG HI": LET j$="JKL": LET k$
 "M N"
2030 RETURN
8888 REM ALTERACOES
9000 SAVE "ENSAIO 4" LINE 9100
9010 SAVE "ROTINA 1"CODE 62750,2610
9020 VERIFY "": VERIFY ""CODE : STOP
9100 CLEAR 62749: LOAD "ROTINA 1"CODE 62750,2610: RUN
```

CAPÍTULO IV

EM FALTA

1 — INTRODUÇÃO

A «manipulação de endereços», quando perfeitamente dominada, é uma técnica poderosa, permitindo executar programas de grande nível, capazes de proporcionar resultados (em qualidade e rapidez), só ultrapassáveis pelo recurso ao código–máquina.

Nos anteriores capítulos o leitor teve ocasião de compreender que os «endereços» não são exclusivamente utilizados nas rotinas em código–máquina, antes se mas tornam num meio eficaz de controlo através da linguagem «BASIC».

Este capítulo vai ser dedicado a um programa de desenho, baseado na *utilização de «endereços» para armazenar os valores das coordenadas «X» e «Y» da rede de pixels.* Esta técnica não será inédita, mas o programa «DESEEND» desenvolve-a, facultando ao leitor uma «ferramenta de trabalho» de grande utilidade, para a realização de desenhos a serem incluídos nos seus programas.

2 — O PROGRAMA «DESEENDE»

O programa é totalmente construído em BASIC, tendo existido a habitual preocupação de o dotar de instruções capazes de proporcionar uma fácil e agradável utilização, através de frequentes avisos sonoros, frases e textos oportunamente colocados, facultando permanente informação sobre o andamento das operações.

São utilizadas oito teclas para a movimentação do «pixel», a que chamaremos, também, «cursor» ou «ponto». As teclas foram seleccionadas tendo em conta uma rápida memorização por parte de qualquer utilizador: vejamos como se apresentam as teclas de cursor:

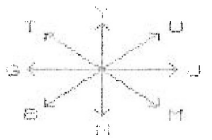


Figura 1/4

As teclas «G» e «J» movem o ponto na horizontal; as teclas «Y» e «N» na vertical e as teclas «T», «M», «B» e «V» movem o ponto na diagonal. Esta disposição em forma de «rosa–dos–ventos», e no lado direito do teclado, torna–se facilmente memorizável e permite libertar a fila superior (teclas de 1 a 0) para outros comandos.

Vejamos a página inicial, onde são apresentadas as funções das diferentes teclas de comando:

```
FUNCOES DAS TECLAS

TRACOS HORIZ/VERTIC.....:G J Y N
TRACOS DIAGONAIS.......:T M B U

DESLIGAR CURSOR.........PREMIR 0
LIGAR CURSOR............PREMIR 1
APAGAR PONTOS...........PREMIR 2
VER DESENHO.............PREMIR 3
GRAVAR DESENHO..........PREMIR 4
GRAVAR SCREEN...........PREMIR 5
CARREGAR DESENHO........PREMIR 6
COPY/IMPRESSORA.........PREMIR 7
PARAR PROGRAMA..........PREMIR 8

INICIAR  DESENHO.......PRIMA I
CARREGAR DESENHO.......PRIMA C

PRIMA "I" OU "C"
```

Figura 2/4

Combinando as diferentes teclas do cursor, é possível obter linhas rectas para a realização de figuras ortogonais, ou linhas sinuosas, para a obtenção de mapas ou cartas geográficas, como pode ver-se na figura 3/4.

A figura 4/4 é um exemplo de desenho de máquinas, representando um «rolete». Obviamente, os valores das «cotas» foram impressos fora do programa, através de instruções «PRINT AT».

A figura 5/4 é constituída por um cubo e um paralelepípedo, em perspectiva, e uma «fantasia», composta por linhas horizontais, verticais e oblíquas.

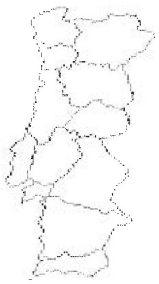


Figura 3/4

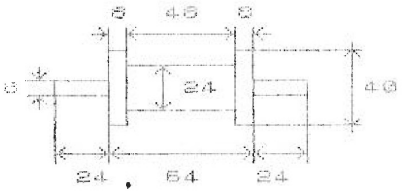


Figura 4/4

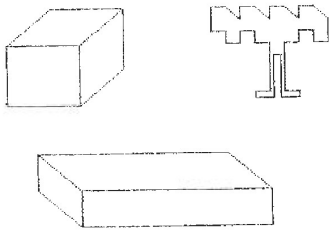


Figura 5/4

Todas estes desenhos foram executados com o programa «DESEEND», recorrendo-se a um método de trabalho que aconselhamos o leitor a seguir, para a obtenção dos resultados pretendidos. A base do trabalho encontra-se na p. 76 do *Manual do ZX SPECTRUM*, constituída pela já nossa conhecida rede de 22 linhas por 32 colunas, dividida em 256 × 176 pixels.

Sugerimos ao leitor que construa a sua própria rede em formato maior, de preferência com quadrados de 8 mm de lado. Na base, trace mais duas filas e, do lado direito, prolongue as linhas para mais duas colunas. Reserve essas zonas para a inscrição da numeração «pixel», tendo o cuidado de escrever a numeração das coordenadas «Y» na posição «normal» e corrigir uma pequena «gralha» tipográfica: a coordenada «X 169» deverá ser «167».

Qualquer figura que o leitor deseje executar no «DESEEND» deverá ser primeiro desenhada na rede, na posição definitiva em relação às margens ou a outras figuras existentes ou a desenhar. Como regra, e sempre que possível, o leitor deverá aproveitar as linhas da rede para constituírem «linhas do desenho»: torna-se assim mais fácil referenciar as «coordenadas de início».

Vamos fazer um pequeno exercício com o cubo da figura 5/4. Aliás, este servirá para a primeira explicação sobre «como desenhar com o DESEEND». Mas, antes, duas recomendações: faça vários exemplares da rede (ou fotocópias) e nunca utilize o «original»; faça os desenhos a lápis: pode corrigir sem inutilizar o papel.

— Inicie o desenho nas coordenadas «47,127», isto é, no ponto de cruzamento da coordenada horizontal «X 47» com a coordenada vertical «Y 127»;

— Trace: uma diagonal até às coordenadas «71,151», uma horizontal até às coordenadas «111,151» (daqui em diante omitiremos a palavra «coordenadas»), uma vertical até «111,111», uma diagonal até «87,87», uma horizontal até «47,87», uma vertical até ao ponto de início, isto é, «47,127», uma horizontal até «87,127» e uma diagonal até «111,151»; volte a «87,127» e trace, por último, uma diagonal até «111,151».

A sequência pela qual indicámos a traçagem não é arbitrária: está directamente relacionada com a utilização do programa e, como o leitor terá ocasião de verificar, é uma das soluções que permite a maior economia de movimentos do cursor.

Temos, portanto, desenhada uma perspectiva de um cubo com «40 pontos de aresta»: com efeito, a «distância» que vai de «X 71» até «X 111» tem 40 pontos, o mesmo acontecendo entre «Y 111» e «Y 151».

Os restantes desenhos da figura 5/4 foram executados pelo mesmo processo: o paralelepípedo foi iniciado em «63,55» e a «fantasia» em «159,135». Voltaremos com mais pormenor a estes dois desenhos, na secção seguinte.

Passemos à figura 3/4, representando o mapa de Portugal. Para a execução deste tipo de desenhos, sugerimos ao leitor a adopção do seguinte método: decalque a figura para uma folha de acetato (previamente desenhada, ou obtida a partir de um livro, mas sempre com dimensões compatíveis com as da rede). Corte o acetato com margens

suficientes e coloque-o sobre a rede, procurando a melhor localização: no caso presente, escolhemos como «ponto de referência» o cabo de São Vicente e colocámo-lo nas coordenadas «111,15», de modo a que o mapa se situasse sensivelmente a meio do écran.

A fase seguinte será tratada no próximo parágrafo. Chegámos à altura de apresentar o programa; eis a listagem:

PROGRAMA 1/4

```
  1 REM COPYRIGHT          DESENHO EM ENDERECOS          R.
CASQUILHO 1986
  5 POKE 23658,8: POKE 23693,56: BORDER 7: LET end1=3000
0: LET end2=45000: CLS
 10 PRINT AT 0,7; BRIGHT 1;"FUNCOES DAS TECLAS": PRINT A
T 3,0;"TRACOS HORIZ/VERTIC......G J Y N"'"TRACOS DIAGONAI
S.........T M B U"
 20 PRINT '"DESLIGAR CURSOR.........PREMIR 0"'"LIGAR CUR
SOR............PREMIR 1"'"APAGAR PONTOS...........PREMIR
2"
 30 PRINT "VER DESENHO.............PREMIR 3"'"GRAVAR DES
ENHO..........PREMIR 4"'"GRAVAR SCREEN...........PREMIR
5"
 40 PRINT "CARREGAR DESENHO.........PREMIR 6"'"COPY/IMPRE
SSORA.........PREMIR 7"'"PARAR PROGRAMA..........PREMIR
8"
 50 PRINT '" INICIAR DESENHO.......PRIMA I"'" CARREGAR
DESENHO.......PRIMA C"'''TAB 8; BRIGHT 1;"PRIMA ~I~ ou ~C
~"
 60 LET z$=INKEY$: IF z$<>"I" AND z$<>"C" THEN BEEP .03
40: PAUSE 20: GO TO 60
 70 IF z$="I" THEN CLS : GO TO 100
 80 CLS : LET a$="Cursor ligado.   ": GO TO 2000
100 BEEP .2,30: INPUT "QUANTOS DESENHOS ? ";W
105 FOR Q=1 TO W
110 BEEP .3,20: INPUT "COORDENADA X (0 a 255) ";a: BEEP
3,30: INPUT "COORDENADA Y (0 a 175) ";b
120 LET r=0: LET C=0: LET a$="Cursor ligado.   "
130 LET D=CODE INKEY$
140 LET a=a+(D=85)+(D=74)+(D=77)-(D=84)-(D=71)-(D=66)
150 LET b=b+(D=84)+(D=89)+(D=85)-(D=66)-(D=78)-(D=77)
160 LET a=a-(a>255)+(a<0)
170 LET b=b-(b>175)+(b<0)
180 IF D=48 THEN LET r=1: LET a$="Cursor desligado!": B
EEP .2,15
190 IF D=49 THEN LET r=0: LET a$="Cursor ligado.   ": B
EEP .2,15
200 IF D=50 THEN LET r=2: LET a$="Apagar pontos:   ": B
EEP .2,15
210 IF D=51 THEN GO SUB 700: GO TO 130
220 IF D=52 THEN CLS : GO TO 3000
230 IF D=53 THEN GO TO 1500
240 IF D=54 THEN CLS : GO TO 2000
250 IF D=55 THEN COPY : GO TO 130
260 IF D=56 THEN GO TO 3100
300 IF r=0 THEN PLOT a,b: POKE end1+C,a: POKE end2+C,b
310 PRINT #0;AT 0,3;"X=";a;" ";"Y=";b;" ";a$;" ";TAB 10;
"Desenho No.";Q
320 IF D=85 OR D=84 OR D=77 OR D=71 OR D=66 OR D=89 OR D
=78 OR D=74 THEN IF r=0 THEN LET C=C+1
330 IF D=85 OR D=74 OR D=77 OR D=84 OR D=71 OR D=66 OR D
=89 OR D=78 THEN IF r=2 THEN PLOT a,b: BEEP .003,25: PL
OT OVER 1;a,b: LET C=C-1
340 IF C>=15000 THEN LET C=15000: INPUT "": PRINT #0; F
LASH 1;"MEMORIA ESGOTADA-GRAVE DESENHO": BEEP .5,10: PAU
SE 100
350 GO TO 130
700 CLS : FOR f=0 TO C-1
710 PLOT PEEK (end1+f),PEEK (end2+f)
720 NEXT f
730 BEEP .3,30: RETURN
1000 PRINT #0;"   CORRIGIR->1 CONTINUAR->2   ": BEEP .2
,20
1010 LET z=CODE INKEY$-48: IF z<1 OR z>2 THEN GO TO 10
10
1020 IF z=1 THEN INPUT "": GO TO 130
1030 RETURN
1500 BEEP .3,20: INPUT "Nome do desenho ";n$: IF n$="" TH
EN PRINT #0;"TEM DE TER UM NOME -> REPITA": BEEP .5,10:
PAUSE 50: GO TO 1500
1510 IF LEN n$>10 THEN INPUT "": PRINT #0;"NOME MUITO LO
NGO -> REPITA": BEEP .5,10: PAUSE 50: GO TO 1500
1520 SAVE n$SCREEN$ : BEEP .3,30
1530 GO TO 130
2000 PRINT AT 0,8;"CARREGAR DESENHO"'''"Vai carregar um d
esenho ja gra-vado em cassette, por meio desteProgra
ma."
2010 PRINT '"Depois de carregados os dois mo-dulos, pode
modificar ou comple-tar o desenho,utilizando as mes-mas
teclas de CURSOR."
2020 PRINT '"As coordenadas do ultimo pixeldo desenho
carregado aparecem nofundo do ecran."'''"Indique o numero
```

```
de BYTES, e os dois enderecos do desenho: ligueo gravado
r:"
2030 BEEP .3,20: INPUT "Bytes ? ";n: BEEP .3,20: INPUT "E
ndereco 1 ";end1: BEEP .3,20: INPUT "Endereco 2 ";end2
2040 LOAD ""CODE end1,n: LOAD ""CODE end2,n
2050 LET C=n: GO SUB 700
2060 LET a=PEEK ((end1+n)-1): LET b=PEEK ((end2+n)-1):
LET W=1: LET Q=1: LET r=0: GO SUB 1000: GO TO 5
3000 PRINT AT 0,0;"GRAVAR BYTES DO DESENHO No.";Q: PRINT
AT 3,0;"Vai gravar os dois modulos, dos enderecos :"'"En
dereco 1:";end1;" com ";C;" bytes"'"Endereco 2:";end2;" c
om ";C;" bytes"
3010 PRINT '"Indique o Titulo do desenho.Pre-pare uma cas
sette limpa e ponhao gravador a funcionar.
 Quandoaparecer pela segunda vez a or-dem: ~Start tape
then press anykey~,deixe o gravador a funcio-nar e pri
ma qualquer tecla."
3020 PRINT '"Conserve a cassette com o titulodo desenho,
o numero de bytes e os enderecos."
3030 BEEP .3,20: INPUT "Nome do desenho? ";n$: IF n$="" T
HEN PRINT #0;"TEM DE TER UM NOME -> REPITA": BEEP .5,10:
PAUSE 50: GO TO 3030
3035 IF LEN n$>10 THEN INPUT "": PRINT #0;"NOME MUITO LO
NGO -> REPITA": BEEP .5,10: PAUSE 50: GO TO 3030
3040 SAVE n$CODE end1,C: SAVE n$CODE end2,C
3050 CLS : PRINT AT 0,0;"Tome nota dos enderecos e do nu-
mero de bytes:";'' BRIGHT 1;end1;",";C'end2;",";C''; BRIG
HT 0;"Gravou o desenho No.";Q: BEEP .3,20
3060 PRINT AT 10,0;"REVER DESENHO:";TAB 24;"PRIMA R"'"DES
ENHO SEGUINTE";TAB 24;"PRIMA S"'"ACABAR";TAB 24;"PRIMA
A"
3070 LET z=CODE INKEY$: IF z<>82 AND z<>83 AND z<>65 THEN
GO TO 3070
3080 IF z=82 THEN GO SUB 700: GO SUB 1000: GO TO 3050
3090 IF z=83 THEN LET end1=end1+C: LET end2=end2+C: LET
C=0: CLS : NEXT Q
3100 STOP
9000 CLEAR : SAVE "DESENHO" LINE 1
9010 VERIFY "": STOP
```

Como funciona

Linha 5 — Colocação do valor «8» na variável de sistema 23658: este comando garante a passagem automática para letras maiúsculas (ver nota à EXPERIÊNCIA 4 do GERAGRAF); colocação do valor «56» na variável de sistema «ATTR p»; este comando tem o mesmo efeito que: «PAPER 7: INK 0: BRIGTH 0: FLASH 0». No Apêndice 1 o leitor encontrará uma tabela e explicações para a utilização desta variável de sistema; atribuição do valor «30000» à variável «end1» e do valor «45000» à variável «end2».

Linhas 10/80 — Impressão da página inicial, tal como está representada na figura 2/4; o programa aguarda (linha 60) que sejam premidas as teclas «I» ou «C»: se premirmos «I», o écran é limpo e o comando passa à linha 100; se premirmos «C», o écran é limpo, a cadeia «Cursor ligado» é atribuída à variável «a$» e o comando passa para a rotina 2000.

Linha 100 — O programa indaga sobre o número de desenhos que pretendemos executar: o quantitativo fica guardado na variável «W».

Linha 105 — Estabelecimento do ciclo FOR–NEXT «Q», cujo comprimento é o do valor contido em «W».

Linha 110 — Instrução múltipla, com duas entradas, uma para a coordenada inicial «X», outra para a coordenada inicial «Y», cujos valores ficarão atribuídos, respectivamente, às variáveis «a» e «b».

Linha 120 — Inicialização das variáveis «r» e «C» com o valor «0»; colocação da cadeia «Cursor ligado» na variável «a$»: isto já tinha sido feito na linha 80, mas torna–se necessário repeti–lo aquando da passagem pela rotina 3000.

Linhas 130/350 — Rotina que controla a execução do desenho, bem como as diferentes teclas das «funções»; a variável «D» conterá o código da tecla premida (linha 130); o movimento do «ponto» é comandado pelas instruções das linhas 140 e 150: reportemo–nos à figura 1/4 e lembremo–nos de que as variáveis «a» e «b» contêm os valores iniciais das coordenadas «X» e «Y». Retomaremos esta explicação mais adiante, com o pormenor que se impõe.

Linhas 700/730 — Sub–rotina que imprime no écran o desenho executado, fazendo «PLOT» sobre os valores retirados (PEEK) dos endereços «end1» e «end2», os quais correspondem às coordenadas «a» e «b» e neles foram colocados (POKE) pelas instruções da linha 300.

Linhas 1000/1030 — Sub–rotina destinada a permitir a correção de um desenho acabado de carregar.

Linhas 1500/1530 — Rotina que nos faculta a gravação do desenho sob a forma de «SCREEN$».

Linhas 2000/2060 — Rotina destinada ao carregamento dos bytes de qualquer desenho, previamente gravado através do programa: após carregamento dos dois módulos (endereços «end1» e «end2»), o desenho é impresso pela chamada à sub–rotina 700, sendo atribuído à variável «C» o valor contido em «n», isto é, o número de bytes que indicámos no «INPUT» da linha 2030; na linha 2060 são atribuídos a «a» e a «b» os valores contidos em «end1+n» e «end2+n», isto é, nos endereços indicados nos «INPUT» da linha 2030, somados do

valor contido em «n»: deste modo, *«a» toma o valor da última coordenada «X» e «b» o valor da última coordenada «Y»* do desenho carregado; reinicializadas as variáveis «W», «Q» e «r»; passagem à sub–rotina 1000, o que nos conduz à página inicial, caso não queiramos corrigir o desenho.

Linhas 3000/3100 — Rotina destinada à gravação dos dois módulos do desenho: são indicados o número do desenho, os endereços iniciais e o número de bytes; as linhas 3030 e 3035 contêm instruções que impedem a entrada de «um nome vazio» ou de um nome com mais de dez caracteres; a linha 3040 grava os dois módulos, cada um com o comprimento «C»; a linha 3050 volta a dar–nos os elementos que devemos anotar, passando–se à linha 3060, que imprime no écran as três opções à nossa disposição: «REVER DESENHO», com passagem pelas sub–rotinas 700 e 1000, o que permite efectuar correcções e voltar a gravar; «DESENHO SEGUINTE» (linha 3090) onde são criados os endereços para o desenho que se segue: «end1» e «end2» tomarão os valores anteriores somados do comprimento «C»; deste modo, o novo desenho iniciar–se-á nos endereços imediatamente seguintes; a variável «C» é reinicializada a «0» e o «NEXT Q» leva–nos à linha 110, com pedido de novas coordenadas.

Linhas 9000/9010 — Instruções reservadas para a cópia do programa: a instrução «CLEAR» coloca as variáveis a «zero» e o programa arranca automaticamente na primeira linha funcional.

Retomemos as instruções das linhas 130/150, relembrando os códigos dos caracteres utilizados para as diferentes «funções»:

Teclas de cursor

Carácter	Código	Carácter	Código
G	71	T	84
J	74	M	77
Y	89	B	66
N	78	U	85

Teclas de funções

Carácter	Código	Carácter	Código
0	48	5	53
1	49	6	54
2	50	7	55
3	51	8	56
4	52	—	—

Repare que a rotina 130/150 é «um ciclo fechado», por meio da linha 350: o programa encontra–se «bloqueado», aguardando que seja premida uma das teclas cujos códigos estão autorizados. Vejamos, primeiro, como funcionam as teclas de cursor e como o «ponto» é impresso no écran.

As variáveis «a» e «b» contêm os valores das coordenadas horizontal «X» e vertical «Y», indicadas nos «INPUT» da linha 110; as linhas 140 e 150 funcionam como instruções condicionais, permitindo que os valores contidos em «a» e «b» *sejam aumentados ou diminuídos, consoante a tecla premida.*

Para maior clareza, sugerimos ao leitor que introduza já o programa, faça a sua gravação de segurança e arranque com «GO TO 1». Após a impressão da página inicial, prima «I» e indique os valores «47» para «X» e «127» para «Y». Siga metodicamente as nossas instruções e explicações:

a) Os valores «47» e «127» estão atribuídos às variáveis «a» e «b»: se premirmos a tecla «U» (código 85), os dois valores vão aumentando, pois o código 85 encontra–se do lado «positivo» em ambas as instruções das linhas 140 e 150.

a.1) Lembremo–nos de que a variável «r» tem o valor «0» e desçamos à linha 300: se *«r = 0», então faça «pontos» nas coordenadas «a» e «b»:* como «a» e «b» vão aumentando do mesmo valor, os «pontos» vão sendo impressos no écran, desenhando *uma diagonal a 45 graus, para a direita e para cima, enquanto a tecla «U» estiver premida* (ver figura 1/4).

a.2) Simultaneamente, os sucessivos valores de «a» e «b» são impressos por efeito das instruções da linha 310, o mesmo se passando com a mensagem «Cursor ligado»: deste modo torna–se possível controlar, permanentemente, as coordenadas do último «ponto» impresso.

b) O leitor manteve a tecla «U» premida, até as coordenadas atingirem «71,151»; vamos, agora, prolongar o traço para a direita, na horizontal, isto é, *vamos aumentar o valor de «a» mantendo o valor de «b»:* a linha 140 permite fazê–lo, pois possui o código 74 do lado «positivo»: se o leitor premir a tecla «J», cujo código é 74 e só existe no lado «positivo» da linha 140, o valor de «a» é aumentado e o valor de «b» não sofre qualquer alteração.

b.1) A linha 300 desenha um traço horizontal para a direita e a linha 310 imprime os sucessivos valores da coordenada horizontal «X». Mantenha a tecla «J» premida até as coordenadas tomarem o valor «111,151».

c) O leitor quer, agora, fazer descer o ponto, na vertical: como «descer» é «negativo», torna–se necessário encontrar a tecla que diminua o valor de «b», sem alterar o valor de «a»: a linha 150 possui o código 78 do lado «negativo», o qual corresponde à tecla «N». Deste mo-

do, premindo «N», o valor de «b» vai diminuindo e o valor de «a» não é alterado.

c.1) A linha 300 desenha um traço vertical descendente e a linha 310 imprime os sucessivos valores da coordenada «Y». Mantenha a tecla «N» premida até as coordenadas tomarem o valor «111,111».

NOTA. — Como o leitor já se apercebeu, estamos a repetir, agora com o programa a funcionar, o exercício que sugerimos executar sobre a «rede»: desenhar o cubo da figura 5/4. Prossigamos com as explicações.

d) Precisamos, agora, de uma diagonal, para baixo e para a esquerda: a tecla indicada é a «B», pois tanto a linha 140 como a linha 150 possuem, do lado «negativo», o código 66.

d.1) Premindo a tecla «B», a linha 300 traça a diagonal pretendida: mantenha a pressão, até as coordenadas tomarem o valor «87,87».

e) É a vez de desenharmos uma linha horizontal para a esquerda, diminuindo o valor de «a» e mantendo o valor de «b». No lado «negativo» da linha 140 encontramos o código 71, correspondendo à tecla «G».

e.1) Premindo «G», a linha 300 executa o traço pretendido, até as coordenadas tomarem o valor «47,87».

f) Segue-se uma linha vertical para cima: a linha 150 possui, do lado «positivo», o código 89, correspondente à tecla «Y», o que nos permite aumentar o valor de «b», sem afectar o valor de «a».

f.1) Prima «Y», até as coordenadas tomarem o valor «47,127», isto é, o ponto de início do desenho.

g) Vamos executar a linha horizontal para a direita: será com a tecla «J», já anteriormente utilizada. Mantenha-a premida, até as coordenadas tomarem o valor «87,127».

h) Seguidamente, uma diagonal para a direita e para cima, premindo a tecla «U», até as coordenadas tomarem o valor «111,151».

i) Resta ao leitor desenhar a vertical que une as coordenadas «87,87» às coordenadas «87,127». Para o efeito, o programa possui a função «DESLIGAR CURSOR», que se obtém premindo a tecla «0», cujo código é 48.

i.1) A instrução condicional da linha 180 coloca a variável «r» com o valor «1», quando premida a tecla «0»; simultaneamente, coloca a cadeia «Cursor desligado !» na variável «a$». Como «r» não tem os valores «0» ou «2», as instruções condicionais das linhas 300, 320 e 330 impedem o «PLOT».

i.2) Premindo a tecla «0» ouve-se um aviso e é impressa a mensagem «Cursor desligado !». Nesta altura, o leitor pode mover o cursor em qualquer sentido, sem desenhar no écran: isto permite *procurar as coordenadas onde se irá iniciar o próximo traço: como o cursor estava em «111,151», utilize a tecla «B» e desça «em diagonal», até «87,127».*

i.3) Para ligar o cursor, isto é, para repor o valor «0» na variável «r», temos as instruções da linha 190: *se premir a tecla «1», cujo código é 49, «r» passará a «zero» e «a$» conterá a cadeia «Cursor ligado», com novo aviso sonoro.* Pode, agora, o leitor desenhar a linha vertical, premindo «N», até as coordenadas tomarem o valor «87,87».

Se o leitor seguiu atentamente as nossas instruções e explicações, tem, agora, desenhado no écran um cubo em perspectiva, e certamente compreendeu o funcionamento do «cursor» e a forma como é executada a impressão dos «pontos» no écran.

Prossigamos as nossas explicações, falando das linhas 160 e 170. As instruções nelas contidas, aparentemente «herméticas», são de grande importância num programa de desenho: impedem que o «ponto» saia do écran, com a paragem do programa. Vejamos como funcionam:

— Como o leitor tem presente, as coordenadas horizontais vão de «0» a «255» para a direita e de «0» a «175» para cima. Na linha 160 «diz-se» que o valor de «a» não será aumentado *se for maior que 255, nem será diminuído, se for menor que 0,* isto é, o ponto «pára» quando chega aos extremos laterais da «rede», mesmo que o leitor insista em premir as teclas «G» ou «J».

— O mesmo se passa com a linha 170, mas agora no sentido vertical. Premindo «Y», o ponto pára quando a coordenada «Y» atinge 175; premindo «N», o ponto pára quando a coordenada «Y» atinge 0.

Regressemos à linha 300 para abordarmos a parte mais interessante do programa: a colocação nos endereços «end1» e «end2» dos valores das coordenadas, contidos em «a» e «b».

A análise deste processo terá de ser feita em conjunto com as instruções da linha 320:

— A linha 320 é uma instrução condicional múltipla, que incrementa o valor de «C» sempre que seja premida uma das teclas de cursor. Com efeito, *se o valor contido em «D» coincidir com um dos códigos das teclas de cursor e se o valor contido em «r» for «zero», o valor de «c» é incrementado de uma unidade.*

— Deste modo, no início do desenho, «C» contém o valor «0» (linha 120): quando partimos das coordenadas «47,127» e premimos «U» até às coordenadas «71,151», aumentamos os valores de «a» e «b» de 24 pontos; o valor de «C» foi incrementado da mesma quantidade, passando a conter o valor «24».

— «end1» contém o valor «30000» e «end2» o valor «45000». Vejamos como são introduzidas as coordenadas nestes endereços, através das instruções «POKE end1+c,a: POKE end2+c,b»:

Movimento	Valor de «C»	end1	Coord. «a»	end2	Coord. «b»
0	0	30000	47	45000	127
1	1	30001	48	45001	128
2	2	30002	49	45002	129
.	.	.	.	.	.
10	10	30010	57	45010	137
11	11	30011	58	45011	138
.	.	.	.	.	.
23	23	30023	70	45023	150
24	24	30024	71	45024	151

— Assim, no final da execução do traço diagonal, desde «47,127» até «71,151», o endereço 30024 conterá o valor «71», o endereço 45024 conterá o valor «151» e «C» conterá o valor «24». Nos traços seguintes, o valor de «C» continua a ser incrementado e somado a «end1» e «end2» e as instruções «POKE» introduzem em «end1+C» e «end2+C» os valores das coordenadas respectivas. Vejamos o quadro representando o desenho completo:

Movimento	Inicio	Fim	Valor de «C»	end1	end2
1	47,127	71,151	24	30024	45024
2	71,151	111,151	64	30064	45064
3	111,151	111,111	104	30104	45104
4	111,111	87,87	128	30128	45128
5	87,87	47,87	168	30168	45168
6	47,87	47,127	208	30208	45208
7	47,127	87,127	248	30248	45248
8	87,127	111,151	272	30272	45272
9	87,127	87,87	312	30312	45312

— Da observação deste quadro podemos concluir que o desenho do cubo «ocupou» 312 bytes, tendo as coordenadas finais (87,87) sido colocadas, respectivamente, em 30312 e 45312.

Antes de passarmos à análise da linha 330, a qual contém as instruções destinadas a «APAGAR PONTOS», vejamos o que sucede quando iniciamos o próximo desenho:

— Após a gravação em cassete dos bytes do desenho n.º 1 usando a rotina 3000/3100, a opção «S — DESENHO SEGUINTE» coloca «end1» no valor «end1+c = 30312», «end2» em «end2+c = 45312» (linha

3090) e o conteúdo de «C» novamente em «zero». Deste modo, os endereços iniciais do próximo desenho serão 30312 e 45312, ficando o primeiro par de coordenadas alojado em 30313 e 45313: «C» valendo «0», o processo repete–se nos mesmos moldes.

Vejamos então como se processa o «Apagar pontos:», através da linha 330 — esta é uma instrução condicional múltipla que, quando «afirmativa», isto é, *quando o valor de «r» for «2» e for premida uma das teclas de cursor, imprime um «ponto» nas coordenadas «a,b» e, novamente, um outro «ponto» nas mesmas coordenadas, mas agora com a função «OVER 1»: daqui resulta o «apagamento» desse «ponto»*. Vamos exemplificar:

— Admitamos que o leitor, durante a execução do primeiro traço em diagonal por meio da tecla «U», se distraiu e ultrapassou as coordenadas «71,151», por exemplo, até «73,153»; nada mais simples de resolver: prima a tecla «2» e verá aparecer a mensagem «Apagar pontos:» ao lado dos valores das coordenadas. Agora muita atenção: como o traço «errado» foi executado com a tecla «U», terá de ser *apagado no sentido inverso, isto é, com a tecla «B»*.

— Prima cuidadosamente a tecla «B», olhando para os valores das coordenadas, e pare quando atingir os valores desejados, isto é, «71,151»; o apagamento é acompanhado de um clique para melhor percepção dos movimentos; nesta altura ligue de novo o cursor, premindo «1» e, para se certificar do resultado da correcção, prima a tecla «3».

— Entra em acção a linha 210: o comando passa à sub–rotina 700, a qual limpa o écran e desenha novamente o traço, agora corrigido. Retorno à linha 130 e o programa fica preparado para que o leitor continue o seu desenho. De notar que, durante a correcção, o valor de «C» foi diminuído de «2» pontos (LET C=C—1), pelo que retoma o valor de «24».

Resta–nos falar da linha 340: trata–se de uma instrução condicional para protecção do programa, impedindo que o valor de «C» ultrapasse «15000». Embora pouco provável de ocorrer, tal situação acarretaria a destruição das coordenadas verticais colocadas em «end2», pois 30000 + 15000 = 45000, isto é, quando «end1+C» atingisse 45000, a próxima coordenada horizontal iria ser colocada em 45001, substituindo o valor da coordenada vertical já instalada nesse endereço.

3 — GRAVAR E CARREGAR DESENHOS

Vamos partir do princípio de que o leitor seguiu os nossos conselhos e, nesta altura, tem um cubo desenhado no écran. A fase seguinte será a gravação do «desenho» para cassete ou, mais correctamente, gravar os dois «módulos» dos endereços 30000 e 45000, cada um com 312 bytes de comprimento. Prima «4»: entra em acção a linha 220 (o código do carácter «4» é 52) e o comando passa à rotina 3000 com a impressão da seguinte página:

```
GRAVAR BYTES DO DESENHO No.1

Vai gravar os dois modulos, dos
enderecos :

Endereco 1:30000 com 312 bytes
Endereco 2:45000 com 312 bytes

Indique o Titulo do desenho.Pre-
pare uma cassette limpa e ponha
o gravador a funcionar. Quando
aparecer pela segunda vez a or-
dem: "start tape then press any
key",deixe o gravador a funcio-
nar e prima qualquer tecla.

Conserve a cassette com o titulo
do desenho, o numero de bytes e
os enderecos.
```

Siga as indicações do texto e grave os dois módulos com as habituais precauções, estando atento ao aparecimento da segunda mensagem, para premir uma tecla. Feita a gravação, o programa apresenta-lhe outra página de texto:

```
Tome nota dos enderecos e do nu-
mero de bytes:

30000,312
45000,312

Gravou o desenho No.1


REVER DESENHO:              PRIMA R
DESENHO SEGUINTE            PRIMA S
ACABAR                      PRIMA A
```

Tome nota dos elementos indicados e prossiga, escolhendo a opção que mais lhe convier: se vai executar um segundo desenho, prima «S»; se quer rever o seu trabalho com possibilidade de correcção e nova gravação, prima «R»; se terminou os desenhos e deseja passar à fase seguinte, prima «A»: o programa pára (linha 3100). Pensamos que escolherá esta última opção, dando-nos, assim, oportunidade para abordarmos o «CARREGAR DESENHOS».

Tem, portanto, o seu desenho gravado em cassete, numa zona devidamente referenciada, e tomou nota dos endereços e do número de bytes.

Esta gravação vai servir-nos para o ensaio de carregamento. Para o efeito, pare o programa premindo «A» e faça «RUN»: esta acção colocará as variáveis a «zero», mas não destruirá o conteúdo dos endereços onde tem, ainda, os bytes do desenho acabado de gravar. Se o leitor quiser fazer um ensaio cem por cento concludente (é esta a nossa sugestão), desligue a máquina, volte a ligá-la e carregue de novo o «DESEEND». Deste modo, não restarão dúvidas no seu espírito de que a máquina está «limpa», sem qualquer «desenho» em memória. Prima «C» e terá no écran a seguinte página:

```
CARREGAR DESENHO

Vai carregar um desenho ja gra-
vado em cassette, por meio deste
programa.

Depois de carregados os dois mo-
dulos, pode modificar ou comple-
tar o desenho,utilizando as mes-
mas teclas de CURSOR.

As coordenadas do ultimo pixel
do desenho carregado aparecem no
fundo do ecran.

Indique o numero de BYTES, e os
dois enderecos do desenho! ligue
o gravador.
```

Responda às perguntas do computador, prepare a cassete onde gravou os bytes do «cubo» e ligue o gravador: os dois módulos serão carregados consecutivamente e, no final, o «cubo» será impresso no local onde o leitor o desenhou. Tem, depois, duas opções, «CORRIGIR —> 1»: premindo «1» entra em acção a linha 1020 e o comando passa à rotina 130, sendo impressas, na base do écran, as coordenadas do último «ponto» desenhado e a mensagem «Cursor ligado», podendo o leitor alterar ou corrigir o desenho, como entender; «CONTINUAR —> 2»: premindo «2», o programa «retorna» à linha 5, isto é, à página inicial.

Resta-nos falar das três últimas opções facultadas pelo programa. Vamos vê-las.

a) GRAVAR SCREEN

O leitor pode gravar em cassete a «imagem» do écran, recorrendo à função «SCREEN$». Esta solução, utilizada em muitos programas comerciais, permite colocar no écran uma imagem alusiva ao programa em curso de carregamento, a qual, na maioria dos casos, é «limpa» aquando da entrada em funcionamento.

Tem, portanto, uma utilização específica, razão pela qual a incluímos no programa.

Terminado o desenho, prepare uma cassete com uma boa zona livre (vai gravar 6912 bytes) e prima «5»: indique um nome (rotina 1500/1530) e ligue o gravador. No final, terá gravado 6912 bytes, dos quais 6144 do «Ficheiro de imagem» e 768 do «Ficheiro de atributos», a partir do endereço 16384. Conserve esta gravação, pois na secção seguinte, «APLICAÇÕES DOS DESENHOS», explicaremos como tirar partido desta solução.

b) COPY/IMPRESSORA

Se o leitor possuir uma impressora *ZX* ou semelhante, pode copiar a imagem do écran para o papel: prima «7» (linha 250 em acção), tendo previamente instalado e ligado a impressora.

c) PARAR PROGRAMA

Pode parar o programa em qualquer momento da execução dos desenhos: prima «8» e o programa pára, por acção da linha 260, a qual leva o comando à linha 3100.

4 — APLICAÇÕES DOS DESENHOS

4.1 — Desenhos em endereços

Esta é a aplicação para a qual o programa foi concebido, sendo a que se torna mais útil para o leitor, pois permite-lhe introduzir inúmeros desenhos nos seus programas, sem ter de recorrer às fastidiosas instruções «PLOT» e «DRAW».

No caso de desenhos complexos, como mapas ou cartas geográficas, esta solução torna-se particularmente interessante, pois o desenho é «copiado» directamente para os endereços, a partir de uma folha de acetato colocada sobre o écran. O leitor já imaginou o trabalho que representaria determinar as coordenadas de cada «curva» do mapa de Portugal?

A aplicação de um «desenho em endereços» num programa BASIC é extremamente simples, bastando construir e incluir no programa uma sub-rotina semelhante à utilizada no DESEEND (sub-rotina 700): basta colocar os diferentes endereços e comprimentos nas variáveis de comando e chamar a sub-rotina. Vamos exemplificar, utilizando a gravação do «cubo» e construindo um pequeno programa.

Entretanto, se o leitor estiver preocupado com o diminuto «espaço» disponível para o seu programa BASIC (30000 — 23755 = 6245), podemos, desde já, tranquilizá-lo, pois os endereços utilizados no DESEEND podem ser alterados, seja em função do programa de aplicação, seja em conformidade com o número de desenhos e o seu comprimento. Desenvolveremos este assunto mais adiante. Vejamos o programa de aplicação:

```
10 POKE 23693,56: BORDER 7: CLS
100 LET C = 312: LET end1 = 30000: LET end2 = 45000
110 GOSUB 1000
120 PRINT AT 0,0;"CUBO EM PERSPECTIVA"
120 STOP
1000 FOR f = 1 TO C-1
1010 PLOT PEEK (end1+f), PEEK (end2+f)
1020 NEXT f
1030 RETURN
```

O funcionamento do programa dispensa qualquer explicação. Introduza-o na máquina e, por comando directo, carregue os bytes dos dois módulos do «cubo»:

```
LOAD"" CODE 30000,312: LOAD"" CODE 45000,312
```

Faça «RUN»: o cubo será desenhado no écran, exactamente na posição em que foi gravado, e o programa pára, com o título «CUBO EM PERSPECTIVA» impresso na linha 0. Vamos completar o programa com as instruções que permitirão gravar o BASIC em conjunto com os bytes dos dois módulos, para arranque automático com «RUN»:

```
9000 SAVE "CUBO" LINE 9100
9010 SAVE "CUBO 1" CODE 30000,312: SAVE "CUBO 2" CODE 450
00,312 9020 VERIFY"": VERIFY"" CODE: VERIFY"" CODE: STOP
9100 LOAD"" CODE 30000: LOAD"" CODE 45000: RUN
```

Como o leitor pode verificar, a linha 100 introduz as variáveis de comando, atribuindo lhes os valores correspondentes ao desenho em causa. A sub-rotina 1000 funciona para quaisquer valores de «C», de

«end1» e «end2», podendo ser colocada em qualquer ponto do seu programa. A posição das instruções de comando, neste caso as linhas 100 e 110, será função do programa no qual se vão aplicar os desenhos e do momento em que estes devem aparecer no écran.

Vamos admitir que o leitor foi além das nossas sugestões e utilizou o DESEEND para construir e gravar os restantes desenhos da figura 5/4. Se o fez, conhece os respectivos endereços e comprimentos. Na negativa, indicamos os que nos serviram para a sua execução, propondo-lhe um programa experimental, que desenhará a figura que o leitor quiser ver. Mas, antes, uma pequena tabela de endereços e comprimentos:

Desenho	end1	end2	Comprimento	end1	end2
CUBO	30000	45000	312	30312	45312
FANTASIA	30312	45312	384	30696	45696
PARALELEPÍPEDO	30696	45696	456	31152	46152

Esta tabela preliminar vai permitir-nos «relocatar e juntar» os endereços recorrendo ao programa «RELOCJUNTA», estudado no capítulo III. Vamos colocar o endereço inicial numa zona confortável, deixando bastante espaço para o BASIC. Como os nossos desenhos ocupam 2304 bytes (1152 × 2), poderíamos colocar a «RAMTOP» em 60440. Vejamos porquê.

Muito possivelmente, o leitor juntou ao seu programa a ROTINA 1, composta pela rotina «CATGIG», por um conjunto de «GRÁFICOS» e os dois abecedários «ABEC1» e «ABEC2». Como esta rotina já tem o seu endereço determinado: 62750, com 2610 bytes de comprimento, teremos de baixar a «RAMTOP» de 2304 bytes (comprimento dos três desenhos), ou seja (provisoriamente):

```
62750 - 2304 = 60446 -> 60440 arredondando
```

Resta-nos construir a nossa tabela definitiva dos *velhos* e novos endereços arredondados, para boa utilização do «RELOCJUNTA», tendo em conta que são três desenhos com dois endereços cada: seis grupos de bytes a relocatar. Para maior facilidade, chamaremos «GRUPOS» aos desenhos, numerando-os de «1 a 3» em duas alíneas:

	Endereço inicial	Comprim.	Endereço final	Endereço velho inicial
ROTINA 1	*62750*	*2610*	*65360*	—
GRUPO 3.2	62290	456	62746	45696
GRUPO 3.1	61830	456	62286	30696

	Endereço inicial	Comprim.	Endereço final	Endereço velho inicial
GRUPO 2.2	61440	384	61824	45312
GRUPO 2.1	61050	384	61434	30312
GRUPO 1.2	60730	312	61042	45000
GRUPO 1.1	60410	312	60722	30000

Temos, assim, um grupo de 2336 bytes colocados a partir do endereço 60410, o que nos deixa um espaço para BASIC de 35,8 K, com um «desperdício» de 30 bytes, relativamente aos valores anteriormente estimados. Será altura de carregarmos no computador o RELOCJUNTA, prepararmos a cassete onde temos gravados os bytes dos três desenhos e iniciar o trabalho.

Gravado o grupo de 2336 bytes com início no endereço 60410, façamos uma primeira experiência com o seguinte programa:

```
 10 POKE 23693,56: BORDER 7: CLS
100 LET C = 312: LET end1 = 60410: LET end2 = 60730
105 GOSUB 1000
110 LET C = 384: LET end1 = 61050: LET end2 = 61440
115 GOSUB 1000
120 LET C = 456: LET end1 = 61830: LET end2 = 62290
125 GOSUB 1000
130 STOP
1000 FOR f = 0 TO C-1
1010 PLOT PEEK (end1+f), PEEK (end2+f)
1020 NEXT f
1030 RETURN
```

As linhas 100, 110 e 120 atribuem às variáveis de comando os valores retirados da tabela e a chamada à sub-rotina 1000, pelas linhas 105, 115 e 125, colocam, no écran, as três figuras, em sequência. Introduza o programa e, por comando directo, carregue os 2336 bytes e arranque com «RUN»:

```
LOAD "" CODE 60410,2336: RUN
```

O programa seguinte utilizará os bytes «3 DESENHOS» e a «ROTINA 1», terminando, assim, a alínea «Desenhos em endereços».

Após a introdução do programa no computador, grave o BASIC com um comando directo e carregue, primeiro, os bytes «3 DESENHOS»; em seguida os bytes «ROTINA 1»:

```
CLEAR 60409: LOAD "" CODE 60410,2336: LOAD "" CODE 62
750,2610
```

e arranque com «RUN». Se tudo estiver correcto e quiser gravar o programa, faça «GO TO 9000»: gravará o BASIC seguido da nova «ROTINA 2», com 4950 bytes, colocados no endereço 60410:

```
(62750 + 2610) - 60410 = 4950
```

PROGRAMA 2/4

```
  1 REM ENSAIO 5 COM 3 DESENHOS     CATGIG,GRAFICOS ROBOT
E DOIS              ABECEDARIOS
  3  GO SUB 2000
  5 POKE 23693,56: BORDER 7: CLS
  8 GO TO 50
 10 LET w=23306: POKE w,x: POKE w+1,y: POKE w+2,l: POKE
w+3,a: POKE w+4,e: LET w=w+4: FOR f=1 TO LEN g$: POKE w+f
,CODE g$(f): NEXT f: POKE w+f,255: RANDOMIZE USR 65060: R
ETURN
 20 LET w=23306: POKE w,(256-e*l*LEN g$)/2: POKE w+1,y:
POKE w+2,l: POKE w+3,a: POKE w+4,e: LET w=w+4: FOR f=1 TO
LEN g$: POKE w+f,CODE g$(f): NEXT f: POKE w+f,255: RANDOM
IZE USR 65060: RETURN
 50 GO SUB 800: LET g$="3 DESENHOS": LET y=0: LET a=6: L
ET l=2: LET e=10: GO SUB 20
 55 LET g$="Rotina CATGIG": LET y=60: LET a=4: LET l=2:
LET e=8: GO SUB 20: LET g$="2 ABECED£RIOS": LET y=110: LE
T a=2: LET l=2: LET e=8: GO SUB 20
 60 LET g$="Gr#ficos Robot": LET y=140: LET a=3: LET l=2
: LET e=8: GO SUB 20
 70 GO SUB 1000
 80 CLS : GO SUB 800: LET g$="OP©/ES": LET y=0: LET a=2
: LET l=4: LET e=8: GO SUB 20
 90 GO SUB 700: PRINT AT 5,3;"1 - CUBO";AT 7,3;"2 FANTAS
IA";AT 9,3;"3 - PARALELEP+PEDO";AT 11,3;"4 - ROBOT";AT 13
,3;"5 - OS 3 DESENHOS";AT 15,3;"6 - ACABAR": BEEP .3,30
 100 GO SUB 800: LET g$="Prima o n\mero da op'^o desejada
": LET y=150: LET a=2: LET l=1: LET e=8: GO SUB 20: BEEP
.2,30: BEEP .2,40
 110 LET k=CODE INKEY$-48: IF k<1 OR k>6 THEN  GO TO 110
 120 CLS : GO TO (200*(k=1))+(300*(k=2))+(400*(k=3))+(500
*(k=4))+(600*(k=5))+(150*(k=6))
 150 LET g$="O programa parou": LET y=80: LET a=3: LET l=
1: LET e=9: GO SUB 20: GO SUB 900: STOP
 200 LET C=312: LET end1=60410: LET end2=60730: GO SUB 30
00
 210 GO SUB 800: LET g$="CUBO": LET x 47: LET y=0: LET a=
```



```
2: LET l=2: LET e=10: GO SUB 10
 220 LET g$="em": LET x=127: LET y=53: LET a=2: LET l=3:
LET e=8: GO SUB 10: LET g$="PERSPECTIVA": LET x=16: LET y
=111: LET a=4: LET l=2: LET e=10: GO SUB 10
 230 GO SUB 1000: GO TO 80
 300 LET C=384: LET end1=61050: LET end2=61440: GO SUB 30
00
 310 GO SUB 800: LET g$="FANTASIA": LET x=8: LET y=40: LE
T a=4: LET l=2: LET e=9: GO SUB 10
 320 GO SUB 1000: GO TO 80
 400 LET C=456: LET end1=61830: LET end2=62290: GO SUB 30
00
 410 GO SUB 800: LET g$="PARALELEP+PEDO": LET y=90: LET a
=2: LET l=2: LET e=9: GO SUB 20
 420 GO SUB 1000: GO TO 80
 500 GO SUB 850: LET y=8: LET a=2: LET l=2: LET e=8
 505 LET g$=a$: LET y=y+8*a: GO SUB 20: LET g$=b$: LET y=
y+8*a: GO SUB 20
 510 LET g$=c$: LET y=y+8*a: GO SUB 20: LET g$=d$: LET y=
y+8*a: GO SUB 20
 515 LET g$=e$: LET y=y+8*a: GO SUB 20: LET g$=f$: LET y=
y+8*a: GO SUB 20
 520 LET g$=h$: LET y=y+8*a: GO SUB 20: LET g$=i$: LET y=
y+8*a: GO SUB 20
 525 LET g$=j$: LET y=y+8*a: GO SUB 20: LET g$=k$: LET y=
y+8*a: GO SUB 20
 530 GO SUB 800: LET g$="A cabe'a do ROBOT": LET y=0: LET
a=2: LET l=1: LET e=8: GO SUB 20
 535 GO SUB 1000: GO TO 80
 600 POKE 23693,79: BORDER 1: CLS
 610 LET C=456: LET end1=61830: LET end2=62290: GO SUB 30
00
 620 LET C=384: LET end1=61050: LET end2=61440: GO SUB 30
00
 630 LET C=312: LET end1=60410: LET end2=60730: GO SUB 30
00
 640 LET g$="OS 3 DESENHOS": LET y=0: LET a=2: LET l=2: L
ET e=8: GO SUB 20
 650 GO SUB 1000: POKE 23693,56: BORDER 7: GO TO 80
 699 GO SUB 900: STOP
 700 POKE 23606,30: POKE 23607,244: RETURN
 800 POKE 23606,32: POKE 23607,247: RETURN
 850 POKE 23606,34: POKE 23607,250: RETURN
 900 POKE 23606,0: POKE 23607,60: RETURN
1000 GO SUB 900: PRINT #0; INVERSE 1;"      Prima qualque
 tecla      ": BEEP .05,30: BEEP .05,40: PAUSE 0: RETURN
```

```
2000 LET a$="!""#$%": LET b$="&UdefY,": LET c$="-.g0h23"
2010 LET d$="4o6p8": LET e$="9 : ;": LET f$="<=>?@"
2020 LET h$="ABCDE": LET i$="FG HI": LET j$="JKL": LET k$
="M N"
2030 RETURN
3000 FOR f=0 TO C-1
3010 PLOT PEEK (end1+f),PEEK (end2+f)
3020 NEXT f: RETURN
9000 CLEAR : SAVE "ENSAIO 5" LINE 9100
9010 SAVE "ROTINA 2"CODE 60410,4950
9020 VERIFY "": VERIFY ""CODE : STOP
9100 CLEAR 60409: LOAD ""CODE : RUN
```

4.2 — Desenhos em SCREEN$

Vamos utilizar a gravação da imagem do écran, que o leitor conservou. Como já tínhamos referido, estas imagens podem ser colocadas no início do carregamento de um programa, permanecendo até que este entre em funcionamento.

Admitamos que o leitor tem um programa BASIC, acoplado, ou não, a rotinas e deseja incluir–lhe o seu «SCREEN$»: siga o método seguinte:

a) Construa um «cabeçalho»: trata–se de um «minúsculo» programa a colocar no início da cassete, cuja finalidade é instruir o computador para o carregamento dos bytes do «SCREEN$» e do BASIC que se lhe segue:

```
  10 PAPER 7: INK 7: BORDER 7: CLS 20 PRINT AT 0,0;: LOAD
"" SCREEN$
  30 LOAD""
```

Introduza estas linhas na máquina, prepare uma cassete limpa e grave *logo no início da fita,* com o comando directo:

```
SAVE "nome" LINE 10
```

deixe correr um pouco de fita para limpar eventuais gravações anteriores (no caso de não ter utilizado uma cassete limpa) e aproveite o «VERIFY» para colocar a fita na posição exacta em que deverá receber a próxima gravação: o espaço entre o «cabeçalho» e o «SCREEN$» deve ser o menor possível.

b) Coloque no gravador a cassete onde tem o «SCREEN$» : faça LOAD "" SCREEN$ e carregue os 6912 bytes. No final, terá a imagem gravada com o DESEEND. Coloque no gravador a cassete onde gravou o «cabeçalho» e faça:

```
SAVE "nome" SCREEN$
```

c) Pare o gravador logo que termine a gravação e retire a cassete. Coloque no gravador aquela em tem o seu programa, faça «NEW» e carregue-o como habitualmente. Retire a cassete e volte a colocar aquela onde já tem o «cabeçalho» e o «SCREEN$»:

d) Grave o seu programa com GO TO 9000, imediatamente a seguir ao «SCREEN$». É altura de vermos os resultados; rebobine a cassete, faça LOAD "" e ligue o gravador:

— O «cabeçalho» coloca os atributos todos em «branco»: deste modo, a habitual mensagem com o nome do programa ficará «invisível», não perturbando a imagem do «SCREEN$»;

— O «SCREEN$» é carregado e a linha 30 permite o carregamento do seu programa, sem qualquer interrupção.

Este método é válido para qualquer «SCREEN$» que o leitor queira utilizar como «introdução» aos seus programas. Para que os resultados sejam satisfatórios, convém que os diferentes módulos tenham o mínimo de «soluções de continuidade», isto é, que as respectivas gravações se sucedam, praticamente sem «vazios».

Damos por terminado o capítulo IV e, com ele, consideramos alcançado o objectivo deste trabalho: dotar o leitor com os meios que lhe permitirão produzir programas de qualidade, dar–lhe a conhecer algumas das muitas capacidades do *ZX Spectrum* e proporcionar–lhe o estudo e compreensão do funcionamento dos diferentes programas de aplicação e demonstração.

Resta esperar que o seu esforço, capacidade e imaginação conduzam aos resultados que o leitor, e nós, ambicionamos.

APÊNDICE I

A variável de sistema «ATR P», colocada no endereço 23693, permite a introdução de valores decimais de 0 a 255, por meio de instruções «POKE», e comandar os atributos permanentes de forma simples e económica. Substitui as instruções «PAPER», «INK», «BRIGHT» e «FLASH», para toda a gama possível de combinações, com uma só instrução.

Os decimais a colocar no endereço 23693 são determinados através da fórmula seguinte:

```
decimal = (b× 64) + (f× 128) + (p× 8) + i
```

em que:

```
b corresponde a BRIGHT ( 1 ou 0 )
f corresponde a FLASH (1 ou 0)
p corresponde a PAPER (0 a 7)
i corresponde a INK (0 a 7)
```

Segue-se uma tabela dos decimais correspondentes a cada combinação de instruções, sendo:

```
Coluna 1 para PAPER
Coluna 2 para INK
Coluna 3 para BRIGHT
Coluna 4 para FLASH
Coluna 5 para o decimal
```

Exemplifiquemos a utilização:

```
POKE 23693,56 igual a: PAPER 7 : INK 0 : BRIGHT 0 : FLASH 0
POKE 23693,71 igual a: PAPER 0 : INK 7 : BRIGHT 1 : FLASH 0
POKE 23693,135 igual a: PAPER 0 : INK 7 : BRIGHR 0 : FLASH 1
```

A instrução «BORDER» pode, ou não, acompanhar a cor do PAPER. Se quisermos que todo o écran seja afectado pelo valor dos atributos, a instrução deve ser seguida de «CLS»:

```
POKE 23693,71 : BORDER 0 : CLS
```


0-0-0-0: 0	6-5-0-0: 53	5-2-1-0:106	3-7-0-1:159	2-4-1-1:212
0-1-0-0: 1	6-6-0-0: 54	5-3-1-0:107	4-0-0-1:160	2-5-1-1:213
0-2-0-0: 2	6-7-0-0: 55	5-4-1-0:108	4-1-0-1:161	2-6-1-1:214
0-3-0-0: 3	7-0-0-0: 56	5-5-1-0:109	4-2-0-1:162	2-7-1-1:215
0-4-0-0: 4	7-1-0-0: 57	5-6-1-0:110	4-3-0-1:163	3-0-1-1:216
0-5-0-0: 5	7-2-0-0: 58	5-7-1-0:111	4-4-0-1:164	3-1-1-1:217
0-6-0-0: 6	7-3-0-0: 59	6-0-1-0:112	4-5-0-1:165	3-2-1-1:218
0-7-0-0: 7	7-4-0-0: 60	6-1-1-0:113	4-6-0-1:166	3-3-1-1:219
1-0-0-0: 8	7-5-0-0: 61	6-2-1-0:114	4-7-0-1:167	3-4-1-1:220
1-1-0-0: 9	7-6-0-0: 62	6-3-1-0:115	5-0-0-1:168	3-5-1-1:221
1-2-0-0: 10	7-7-0-0: 63	6-4-1-0:116	5-1-0-1:169	3-6-1-1:222
1-3-0-0: 11	0-0-1-0: 64	6-5-1-0:117	5-2-0-1:170	3-7-1-1:223
1-4-0-0: 12	0-1-1-0: 65	6-6-1-0:118	5-3-0-1:171	4-0-1-1:224
1-5-0-0: 13	0-2-1-0: 66	6-7-1-0:119	5-4-0-1:172	4-1-1-1:225
1-6-0-0: 14	0-3-1-0: 67	7-0-1-0:120	5-5-0-1:173	4-2-1-1:226
1-7-0-0: 15	0-4-1-0: 68	7-1-1-0:121	5-6-0-1:174	4-3-1-1:227
2-0-0-0: 16	0-5-1-0: 69	7-2-1-0:122	5-7-0-1:175	4-4-1-1:228
2-1-0-0: 17	0-6-1-0: 70	7-3-1-0:123	6-0-0-1:176	4-5-1-1:229
2-2-0-0: 18	0-7-1-0: 71	7-4-1-0:124	6-1-0-1:177	4-6-1-1:230
2-3-0-0: 19	1-0-1-0: 72	7-5-1-0:125	6-2-0-1:178	4-7-1-1:231
2-4-0-0: 20	1-1-1-0: 73	7-6-1-0:126	6-3-0-1:179	5-0-1-1:232
2-5-0-0: 21	1-2-1-0: 74	7-7-1-0:127	6-4-0-1:180	5-1-1-1:233
2-6-0-0: 22	1-3-1-0: 75	0-0-0-1:128	6-5-0-1:181	5-2-1-1:234
2-7-0-0: 23	1-4-1-0: 76	0-1-0-1:129	6-6-0-1:182	5-3-1-1:235
3-0-0-0: 24	1-5-1-0: 77	0-2-0-1:130	6-7-0-1:183	5-4-1-1:236
3-1-0-0: 25	1-6-1-0: 78	0-3-0-1:131	7-0-0-1:184	5-5-1-1:237
3-2-0-0: 26	1-7-1-0: 79	0-4-0-1:132	7-1-0-1:185	5-6-1-1:238
3-3-0-0: 27	2-0-1-0: 80	0-5-0-1:133	7-2-0-1:186	5-7-1-1:239
3-4-0-0: 28	2-1-1-0: 81	0-6-0-1:134	7-3-0-1:187	6-0-1-1:240
3-5-0-0: 29	2-2-1-0: 82	0-7-0-1:135	7-4-0-1:188	6-1-1-1:241
3-6-0-0: 30	2-3-1-0: 83	1-0-0-1:136	7-5-0-1:189	6-2-1-1:242
3-7-0-0: 31	2-4-1-0: 84	1-1-0-1:137	7-6-0-1:190	6-3-1-1:243
4-0-0-0: 32	2-5-1-0: 85	1-2-0-1:138	7-7-0-1:191	6-4-1-1:244
4-1-0-0: 33	2-6-1-0: 86	1-3-0-1:139	0-0-1-1:192	6-5-1-1:245
4-2-0-0: 34	2-7-1-0: 87	1-4-0-1:140	0-1-1-1:193	6-6-1-1:246
4-3-0-0: 35	3-0-1-0: 88	1-5-0-1:141	0-2-1-1:194	6-7-1-1:247
4-4-0-0: 36	3-1-1-0: 89	1-6-0-1:142	0-3-1-1:195	7-0-1-1:248
4-5-0-0: 37	3-2-1-0: 90	1-7-0-1:143	0-4-1-1:196	7-1-1-1:249
4-6-0-0: 38	3-3-1-0: 91	2-0-0-1:144	0-5-1-1:197	7-2-1-1:250
4-7-0-0: 39	3-4-1-0: 92	2-1-0-1:145	0-6-1-1:198	7-3-1-1:251
5-0-0-0: 40	3-5-1-0: 93	2-2-0-1:146	0-7-1-1:199	7-4-1-1:252
5-1-0-0: 41	3-6-1-0: 94	2-3-0-1:147	1-0-1-1:200	7-5-1-1:253
5-2-0-0: 42	3-7-1-0: 95	2-4-0-1:148	1-1-1-1:201	7-6-1-1:254
5-3-0-0: 43	4-0-1-0: 96	2-5-0-1:149	1-2-1-1:202	7-7-1-1:255
5-4-0-0: 44	4-1-1-0: 97	2-6-0-1:150	1-3-1-1:203	
5-5-0-0: 45	4-2-1-0: 98	2-7-0-1:151	1-4-1-1:204	
5-6-0-0: 46	4-3-1-0: 99	3-0-0-1:152	1-5-1-1:205	
5-7-0-0: 47	4-4-1-0:100	3-1-0-1:153	1-6-1-1:206	
6-0-0-0: 48	4-5-1-0:101	3-2-0-1:154	1-7-1-1:207	
6-1-0-0: 49	4-6-1-0:102	3-3-0-1:155	2-0-1-1:208	
6-2-0-0: 50	4-7-1-0:103	3-4-0-1:156	2-1-1-1:209	
6-3-0-0: 51	5-0-1-0:104	3-5-0-1:157	2-2-1-1:210	
6-4-0-0: 52	5-1-1-0:105	3-6-0-1:158	2-3-1-1:211	

APÊNDICE II

PROGRAMA 1

Converter valores decimais em quantias monetárias, com centavos

```
 10 CLS: INPUT" Quantos valores ? ";n
 20 DIM b(n): FOR f = 1 TO n
 30 INPUT" Valor decimal "; (f);" ";a : LET b(f)=a
 40 PRINT AT f + 1,0;f;TAB 4;a
 50 NEXT f
100 CLS: PRINT" Relacao e soma das quantias em escudos:"
110 LET t = 0: FOR f = 1 TO n
120 LET j = INT b(f): LET j1 = b (f)-j: LET j1 = INT(10 x
ji + 0.5): LET t = t + b(f)
130 LET x$ = STR$ j1 + "0"
140 PRINT: PRINT f; TAB 4;"QUANTIA:"; TAB 18 - LEN STR$
j; j;"$";x$
150 NEXT f
160 LET j = INT t: LET j1 = t-j: LET j1 = INT(10 x j1 + 0
.5)
170 LET x$ = STR$ j1 + "0"
180 PRINT: PRINT" TOTAL: "; TAB 18 - LEN STR$ j;j;"$";x$
```

Arranque com «RUN».

PROGRAMA 2

Converter datas da forma «diamesano» na forma «ano.mes.dia»

```
 10 CLS: INPUT" Data ? "; d
 20 LET d$ = STR$ d
 30 LET d$ = d$ (5 TO) + "." + d$ (3 TO 4) + "." + d$ (TO
2)
 40 PRINT d$
```

Arranque com «RUN».

PROGRAMA 3

Sub-rotina para impressão a meio do écran

```
 10 LET c$ = "Sub-rotina": LET x = 10: GOSUB 5000
 20 STOP
5000 LET a = (32 - LEN c$) /2
5010 PRINT AT x,a; c$
5020 RETURN
```

Arranque com «RUN».

PROGRAMA 4

Sub-rotina para impressão a meio do écran, com esquadria

```
 10 LET c$ = "INTRODUCAO": LET X = 1: GOSUB 5000
 20 STOP
5000 LET l = LEN c$: LET a = INT ((32 - l)/2)
5010 IF l/2 <> INT (l/2) THEN LET a = a - 1
5020 LET c = l x 8
5030 LET b = 8 x (21 - X) - 2
5040 PRINT AT X,a; c$
5050 PLOT (a x 8) - 2,b: DRAW c + 4,0: DRAW 0,11: DRAW -c-
4,0: DRAW 0,-11
5060 RETURN
```

Arranque com «RUN».

PROGRAMA 5

Sub-rotina para impressão «carácter a carácter», com «BEEP»

```
 10 CLS: LET c$ = "Caracter a caracter com BEEP": LET X
 10
 20 GOSUB 5000
 30 STOP
5000 LET a = (32 - LEN c$)/2
5010 FOR f = 1 TO LEN c$
5020 BEEP .3,20
5030 PRINT AT X,f + (a - 1); CHR$ (CODE c$ (f TO))
5040 NEXT f
5050 RETURN
```

Arranque com «RUN».

PROGRAMA 6

Sub-rotina para impressão a partir do fim da cadeia

```
  10 DIM c$ (32)
  20 CLS: LET c$ = "Impressao a partir do fim.": LET X =
0
  30 GOSUB 5000
  40 STOP
5000 FOR f = 32 TO 1 STEP -1
5010 PRINT AT X,0; c$ (f TO)
5020 PAUSE 5: NEXT f
5030 RETURN
```

Arranque com «RUN».