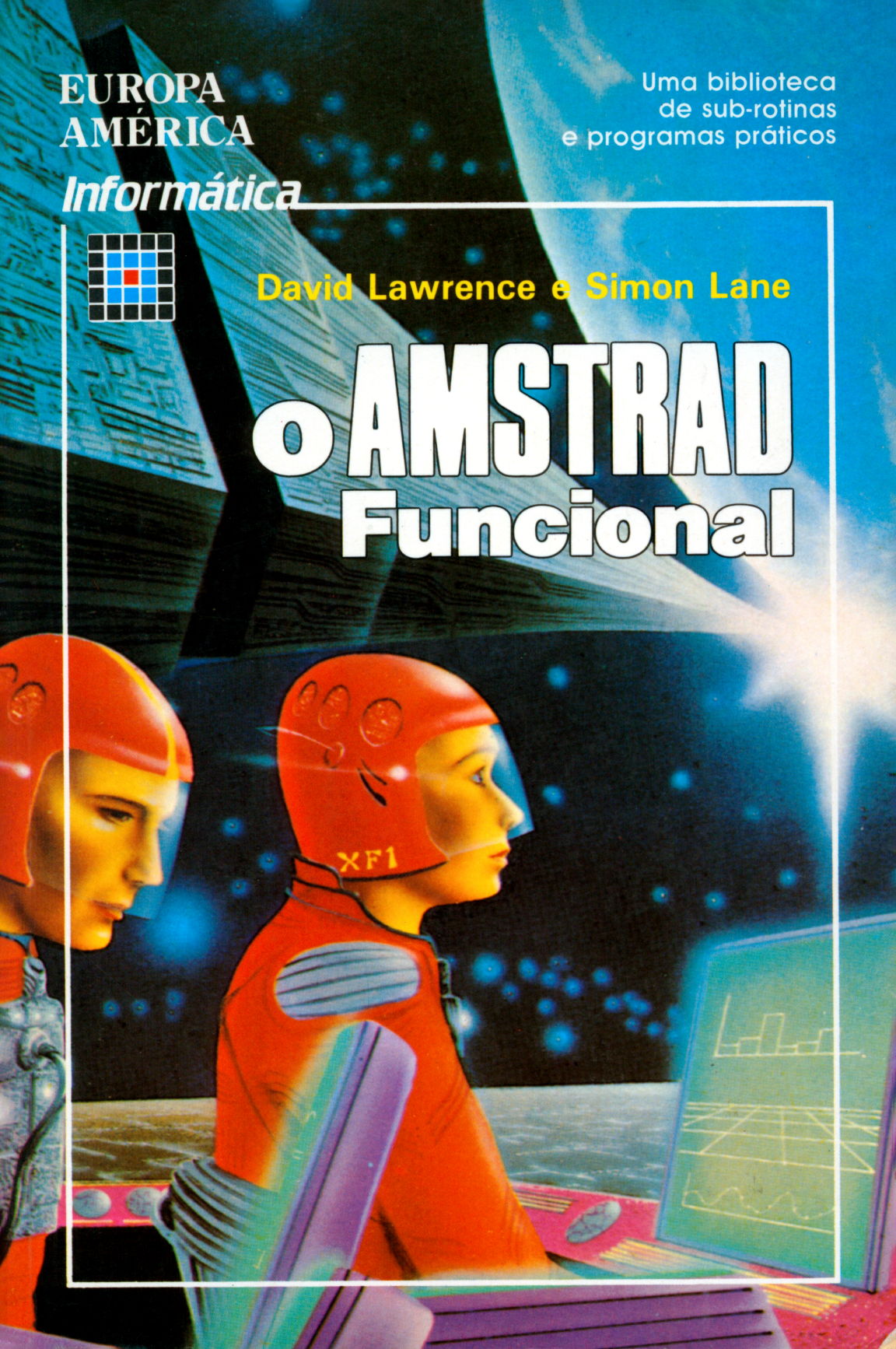
EUROPA AMÉRICA

Informática

Uma biblioteca de sub-rotinas e programas práticos

David Lawrence e Simon Lane

o AMSTRAD Funcional



PARA O LIVREIRO DESTACAR DO LIVRO NO ACTO DA VENDA

AVISO DE LANÇAMENTO

Obra n.º 12500487 1

Título: *O AMSTRAD FUNCIONAL — UMA BIBLIOTEC*
DE SUB-ROTINAS E PROGRAMAS PRÁTICO

Autor: David Lawrence e Simon Lane

Colecção: «Europa-América Informática»

Preço: ..... $ .....

☐ 1.º Fornecimento Factura de fornecimento

☐ Reposição N.º ..... Data .../.../...

Código do cliente

GUARDE ESTE TALÃO

O LIVREIRO DEVE GUARDAR ESTE TALÃO, APÓS A VENDA DO LIVRO, PARA SEU CONTROLO DE VENDAS E REPOSIÇÃO DE NOVO EXEMPLAR.

Este livro só poderá ser devolvido se fornecido em SERVIÇO DE ASSINATURA DE NOVIDADES e de acordo com as cláusulas 12.ª a 17.ª do Contrato de Agência.

© PUBLICAÇÕES EUROPA-AMÉRICA Mod. 12.293 — Impresso por Gráfica Europam, L.da

nas práticos

AVISO DE LANÇAMENTO



O AMSTRAD FUNCIONAL
Uma biblioteca de sub-rotinas e programas práticos

Livros publicados nesta colecção:

1 — *O Sinclair QL Funcional,* David Lawrence
2 — *Como Usar o Atari ST,* Jeremy Vine
3 — *Basic Avançado,* Renato Prista Casquilho
4 — *O Amstrad Funcional,* David Lawrence e Simon Lane

David Lawrence e Simon Lane

o AMSTRAD Funcional

Uma biblioteca
de sub-rotinas
e programas práticos

Informática

PUBLICAÇÕES EUROPA-AMÉRICA

Título original: The Working Amstrad

Tradução de António dos Santos Realinho

Capa: arranjo gráfico de estúdios P. E. A.

© *David Lawrence/Simon Lane*
First published in English 1984 by:
Sunshine Books (an imprint of Scot Books Ltd)
12/13 Little Newport Street
LONDON WC2H 7PP

Direitos reservados por
Publicações Europa-América, Lda.

Nenhuma parte desta publicação pode ser reproduzida ou transmitida por qualquer forma ou por qualquer processo, electrónico, mecânico ou fotográfico, incluindo fotocópia, xerocópia ou gravação, sem autorização prévia e escrita do editor. Exceptua-se naturalmente a transcrição de pequenos textos ou passagens para apresentação ou crítica do livro. Esta excepção não deve de modo nenhum ser interpretada como sendo extensiva à transcrição de textos em recolhas antológicas ou similares donde resulte prejuízo para o interesse pela obra. Os transgressores são passíveis de procedimento judicial

Editor: Francisco Lyon de Castro

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

Edição n.º 125004/4311

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

Depósito Legal n.º 15537/87

Índice

	Pag.
Notas programáticas	9
Introdução	11
1 — *Experiências com o tempo*	13
2 — *Pintura por números*	58
3 — «*Son et lumière*»	84
4 — *Cada vez mais sério*	138
5 — *Questões financeiras*	204

Matérias em pormenor

Capítulo 1 — *Experiências com o tempo*

Anarelógio: apresenta um relógio de mostrador tradicional em alta resolução — Definindo um círculo — Relógio: fornece um modo bem diferente de indicar o tempo — Temporizador: põe à sua disposição dezasseis temporizadores em concorrência, podendo cada um deles fazer soar um alarme e mostrar uma mensagem-memorando — Prova desportiva: transforma o seu *CPC 464* num sofisticado cronómetro.

Capítulo 2 — *Pintura por números*

Gráfico: cria um poderoso e muito eficiente gráfico linear em alta resolução — Gráfico de secções: pega num conjunto de dados limitado e apresenta-o sob a forma de um círculo dividido em segmentos multicolores — Gráfico 3D: permite-lhe criar um espantoso gráfico de barras tridimensional.

Capítulo 3 — *«Son et lumière»*

Caracteres: permite a criação de conjuntos de caracteres ao gosto do utilizador-Memória de caracteres — Desenhador: uma ferramenta para criar desenhos em alta resolução — Música: permite-lhe introduzir melodias bipartidas num formato fácil e reproduzi-las.

Capítulo 4 — *Cada vez mais sério*

Unificheiro: um potente sistema pessoal de ficheiro, capaz de armazenar uma grande variedade de informação, para solicitação instantânea — Pesquisa binária — Nnúmero: cria um dicionário de nomes e números para quase tudo o que o utilizador quiser, permitindo-lhe criar facturas, arquivar avaliações ou mesmo contar as calorias da ementa do dia — Texto: um programa-produto simples, para processamento de texto — MultiQ: um gerador de ensaio de escolha múltipla.

Capítulo 5 — *Questões financeiras*

Banqueiro: mantém um registo ordenado dos pagamentos a débito e a crédito de uma conta bancária — Contabilista: compila um conjunto de contas em formato tradicional.

Notas programáticas

Comandos de cor

Todos os programas incluídos neste livro foram concebidos e ensaiados num *Amstrad* normal, utilizando um monitor de cor. Os leitores que possuírem um monitor monocromático podem ter de considerar, em certas fases, a necessidade de fazer ajustamentos aos comandos de cor.

Armazenamento de dados

O armazenamento de dados é efectuado através do *Datacorder* incorporado. Os leitores que, posteriormente, adquiram um *disk drive* compatível devem encontrar poucas dificuldades em alterar os comandos de abertura e fecho do ficheiro, de modo a adequarem-se à expansão do sistema.

Sinal exponencial

O símbolo ↑ que aparece nos programas é o símbolo da seta para cima, o sinal exponencial.

Introdução

Este livro faz parte de uma das séries de livros para microcomputadores com maior sucesso, de entre todos os que se publicaram até agora. Em 1982, quando foi lançado o primeiro «Working Micro» («Micro Funcional»), poucos editores acreditavam que houvesse vulgares possuidores de micros que desejassem utilizar e compreender as suas máquinas, que quisessem assumir o controlo delas, aprendendo a programar... programando. A maioria dos livros estavam pejados de jogos e outras utilizações triviais, ou consistiam noutro «Guia de Principiantes do...». A perspectiva de um livro que se lançasse no fornecimento de programas sólidos e úteis, dando simultaneamente um conhecimento intrínseco dos métodos empregues em programação a sério, não parecia ser suficiente para entusiasmar o mundo.

Mas eu estava convencido, tal como o estava a Sunshine Books, de que os livros *Micro* «Funcional» eram precisamente aquilo que as pessoas procuravam, para preencher uma enorme lacuna na provisão de livros para o crescente exército dos possuidores de micros. E estávamos certos.

Desde essa altura, os livros «Micro Funcional» têm seguido os microcomputadores para quase todos os países onde são vendidos. Têm sido, ou continuam a ser, traduzidos em catorze línguas.

Aparece agora uma máquina nova e ousada da Amstrad, o *CPC 464,* com uma nova e brilhante versão do BASIC, uma memória mais do que adequada para quase todas as aplicações e uma vasta gama de recursos, que nada têm a ver com o preço notavelmente baixo. O *464,* bem como a ideia subjacente aos livros «Micro Funcional», estão feitos um para a outra — como os programas deste livro, assim o esperamos, irão provar-lhe.

Como utilizar este livro

Pode utilizar o livro de diferentes modos:

1) Como uma colecção de programas úteis, os quais pode adaptar e desenvolver segundo as suas próprias necessidades.
2) Como uma colecção de sub-rotinas, a partir das quais pode construir os seus programas pessoais.
3) Como uma introdução à programação com o BASIC do *CPC 464.*

Seja qual for a forma como decidir utilizá-lo, lembre-se de que ele foi escrito *como um livro,* e não apenas um conjunto desarticulado de programas, sem qualquer

ordem específica. Frequentemente deparamos com leitores em dificuldades por, sem a devida preparação, terem saltado para alguns dos programas mais complexos, os quais se encontram no final do livro. Os primeiros programas do livro, além de terem uma utilidade e interesse intrínsecos, foram também concebidos como uma introdução ao que vem depois. Acompanha estes primeiros programas um grau muito mais alto de explicação, de modo que, ao serem atingidos os programas mais complexos, o leitor tenha um bom domínio de algumas das técnicas em utilização.

CAPÍTULO 1

Experiências com o tempo

É sempre difícil saber por onde recomeçar, num livro destes. Um programa demasiado complexo e os leitores poderão achar-se confusos antes de terem recolhido algumas das indicações simples que tornam os programas progressivamente mais fáceis de compreender, à medida que se avança no livro. Por outro lado, se os primeiros programas forem demasiado triviais, muitos leitores poderão não se dar ao trabalho de descobrir futuras e mais substanciais ofertas.

Posto isto, decidi-me abordar, neste primeiro capítulo, um conjunto de quatro programas que tratam do tempo e da forma como este pode ser manipulado no *CPC 464*.

Os programas deste capítulo são:

ANARELÓGIO: apresenta um relógio de mostrador tradicional em alta resolução.

RELÓGIO: fornece um modo bem diferente de indicar o tempo.

TEMPORIZADOR: põe à sua disposição dezasseis temporizadores em concorrência, podendo cada um deles fazer soar um alarme e mostrar uma mensagem-memorando.

PROVA DESPORTIVA: transforma o seu *CPC 464* num sofisticado (e bastante dispendioso) cronómetro.

PROGRAMA 1.1: ANARELÓGIO

Função do programa

O objectivo deste programa é produzir uma réplica do mostrador de um relógio no écran, justamente com ponteiros que se movem ao ritmo do tempo corrente. Durante a prossecução do programa, aprenderá bastante acerca dos métodos empregues neste livro, pelo que se recomenda a leitura cuidadosa do comentário junto.



Fig. 1.1 — *Impressão de écran do Anarelógio*

Nota. — O efeito ligeiramente ovalado é resultado do processo de impressão do écran — o mostrador do relógio é circular, quando apresentado no écran.

As ideias apresentadas no decurso do programa incluem:

1) Salvaguarda de programas durante o desenvolvimento.
2) Inicialização do programa.
3) Módulos de controlo.
4) Tempos com o comando EVERY.
5) Programação modular.
6) O módulo de controlo.
7) A matemática simples de um círculo.

Módulo 1.1.1: Salvaguarda do programa

Estas três linhas podem parecer um local trivial para começar, mas aqueles que trabalharam com os livros «Funcional», para máquinas anteriores a esta, devem saber que este pequeno módulo pode evitar muitas dores de cabeça durante o desenvolvimento de programas.

A maioria das pessoas aprende apenas por experiência amarga que os programas *devem* ser salvaguardados regularmente, ao longo do seu desenvolvimento. Ce-

do ou tarde, muitos de nós passam por um momento em que horas de trabalho são deitadas à rua devido a um momentâneo corte de energia, a um fusível queimado, a uma pancada no micro ou numa ficha. Para os utilizadores experimentados, apenas se terão perdido uns quinze minutos de trabalho, já que nunca permitirão a existência de mais de quinze minutos de programa sem a respectiva salvaguarda.

O objectivo das três linhas deste módulo é encorajar o utilizador a fazer cópias regulares do programa em que estiver a trabalhar, bastando para isso teclar GOTO 2. Outro pequeno pormenor que, futuramente, poupará tempo, é que um módulo como este, junto ao início de todos os seus programas, fornece ao programa uma linha inicial padrão, ou seja, «1». É frequentemente desejável começar um programa com GOTO, se tiverem sido estabelecidas variáveis que deseje limpar com RUN — com este módulo no lugar, não terá de determinar o número da primeira linha, sabendo que pode sempre começar com GOTO 1.

O módulo precedeu todos os programas do livro, à medida que iam sendo desenvolvidos, mas, visto que apenas o nome do programa muda, ele não será incluído no resto dos programas listados neste livro.

Módulo 1.1.1: linhas 1-3

```
1 GOTO 3
2 SAVE «anarelógio»:STOP
3 REM
```

Ensaio

Assegure-se de que tem uma cassete no gravador. Faça

GOTO 2[ENTER]

Siga as instruções do sistema para pôr o gravador em marcha. Após alguns momentos, deverá ver a mensagem BREAK IN 2. Pode agora apagar as três linhas em memória e voltar a carregar o programa a partir da cassete, rebobinando e carregando-o na memória.

NEW[ENTER] (apaga o programa corrente)
LOAD «ANACLOCK»[ENTER]

Após ter terminado o processo de carregamento, liste o programa e o módulo deverá ter sido reinstalado.

Módulo 1.1.2: Inicialização do programa

Qualquer programa digno desse nome utiliza variáveis e constantes, quer dizer, etiquetas cujos valores podem ser alterados no decurso do programa ou, pelo menos, de programa para programa. A vantagem das variáveis consiste, simplesmente, no

facto de permitirem escrever linhas programáticas que se apliquem a mais de uma situação. Por exemplo, PRINT 2*A poderia ser utilizado sem se atender ao valor de A. Muito poucas variáveis *devem obrigatoriamente* ver os seus valores declarados ao ser passado (RUN) o programa pela primeira vez e muitas pessoas deixam, frequentemente, a definição dos valores para o meio do programa, quando uma variável é absolutamente vital. Trata-se de uma política errada porque, à medida que o programa é desenvolvido, torna-se cada vez mais difícil descobrir qual o valor das variáveis importantes, ao iniciar o programa. É boa prática, regra geral, declarar logo no início do programa o valor das principais variáveis, sendo este processo conhecido por «inicialização».

Assim sendo, uma excepção sensata, quando a memória é limitada, é omitir variáveis cujos valores *não interessam* quando o programa se inicia. Excepto os valores que representam as cores a utilizar, todas as variáveis importantes usadas neste programa são derivadas daquilo que o utilizador introduz, quando se lhe pede que especifique o tempo. Tais variáveis, perfeitamente irrelevantes quando o programa se inicia, são frequentemente omitidas no módulo de inicialização.

No caso deste programa particular, o objectivo do módulo é estabelecer os parâmetros para a eventual visualização dos gráficos e colocar a máquina em modo DEG (de *DEGree,* ou seja, grau), o qual é conveniente para cálculos que envolvam tempo.

Módulo 1.1.2: linhas 2000-2110

```
2000 REM*********************************
2010 REM Initialise
2020 REM*********************************
2030 MODE 1
2040 INK 0,1
2050 INK 1,24
2060 INK 2,3
2070 INK 3,6
2080 BORDER 1
2090 ORIGIN 200,200
2100 DEG
2110 RETURN
```

Ensaio

Pode fazer-se um ensaio rudimentar do módulo inserindo:

```
GOTO 2000
```

Se o módulo foi correctamente introduzido, o écran limpará e será apresentada a mensagem «Unexpected RETURN in 2110». Só será possível uma verificação completa do funcionamento do módulo após terem sido introduzidos módulos subsequentes.

Módulo 1.1.3: Marcação do tempo

Antes de nos lançarmos na criação de um relógio, devemos ter os meios para marcar o tempo. O objectivo deste módulo é permitir ao utilizador inserir o tempo corrente, em horas e minutos, armazenando-o na memória do *464* sob a forma das duas variáveis, HOUR e MINUTE (hora e minuto).

Módulo 1.1.3: linhas 3000-3080

```
3000 REM*********************************
3010 REM Set Time
3020 REM*********************************
3030 PRINT "Clock Setting:":PRINT
3040 INPUT "Hour (1-12):";hour
3050 IF hour<1 OR hour>12 THEN PRINT "**
* Out of range: please try again ***":GO
TO 3040
3060 INPUT "Minute (0-59):";minute
3070 IF minute<0 OR minute>59 THEN PRINT
 "*** Out of range: please try again ***
":GOTO 3060
3080 RETURN
```

Ensaio

Introduza

GOTO 3000[ENTER]

e ser-lhe-á solicitado que forneça o tempo em horas e minutos. Se introduzir um valor não válido para qualquer das duas variáveis, resultará daí uma mensagem de erro, ao passo que um tempo razoável deverá ser aceite. Depois de isto ter acontecido, o programa parará com a mensagem de erro «Unexpected RETURN». Nada de mal se passa — quando o módulo final for introduzido, este módulo tornar-se-á uma sub-rotina, chamada por um GOSUB.

Pode executar uma outra verificação, se assim o desejar, escrevendo:

PRINT hour, minute[ENTER]

o que deverá resultar na visualização em écran dos valores de hora e minuto por si introduzidos.

Módulo 1.1.4: Montagem do mostrador

Após ter inserido o tempo, chegamos à concepção do mostrador do relógio, no qual módulos posteriores irão pôr as mãos.

Módulo 1.1.4: linhas 4000-4090

```
4000 REM*******************************
4010 REM Clock Face
4020 REM*******************************
4030 CLS
4040 FOR a=0 TO 359 STEP 6
4050 MOVE 200*SIN(a),200*COS(a)
4060 r=195:IF a MOD 30=0 THEN r=185
4070 DRAW r*SIN(a),r*COS(a),1
4080 NEXT a
4090 RETURN
```

Comentário

Definição de um círculo

O método empregue neste módulo baseia-se no facto de qualquer ponto da circunferência de um círculo poder ser determinado, caso sejam conhecidos os seguintes elementos de informação:

a) O raio do círculo (RADIUS).
b) O ângulo que tem de ser percorrido no sentido dos ponteiros do relógio, a partir da posição das três horas, para se chegar ao ponto especificado (ANGLE).
c) As coordenadas do centro do círculo (CENTRE X e CENTRE Y).

Dados estes três elementos, a posição será expressa por duas fórmulas:

X coordinate = RADIUS*COSINE(ANGLE) + CENTRE X
Y coordinate = RADIUS*SINE(ANGLE) + CENTRE Y[1]

O espaço não nos permite analisar aqui por que razão isto deve ser assim, mas qualquer bom livro de trigonometria poderá aprofundar esta lógica. No nosso caso particular, as fórmulas são ainda mais simples, já que uma das coisas que fizemos durante o módulo de inicialização foi deslocar a origem (ORIGIN) dos gráficos do écran para 200,200. Isto significa que qualquer referência à posição 0,0 do écran aparecerá, medida a partir do canto inferior esquerdo do écran, numa posição em *pi-*

[1] Coordenada X = RAIO*COSENO(ÂNGULO) + CENTRO X; Coordenada Y = RAIO* SENO(ÂNGULO) + CENTRO Y. *(N. do T.)*

xel de 200 para cima e 200 para a direita. A vantagem disto consiste em que, tendo movido a origem dos gráficos para o local onde se deseja colocar o centro do mostrador do relógio, pode descurar-se qualquer referência às coordenadas X e Y do centro do mostrador — ambas são zero. As fórmulas para a posição de qualquer ponto específico da circunferência do círculo são agora:

X coordinate = RADIUS*COSINE(ANGLE)
Y coordinate = RADIUS*SINE(ANGLE)

que é exactamente o que verá em utilização no módulo.

Linhas 4040-4080: este ciclo leva-nos através dos 360 graus de um círculo, em saltos de seis graus. Visto que há 60 minutos numa hora e 360/60 = 6, não ficará surpreendido por saber que o ciclo tem algo a ver com as marcações dos minutos no mostrador, as quais serão desenhadas por linhas subsequentes.

Linha 4050: o cursor dos gráficos é deslocado (MOVE) para um ponto na circunferência do círculo, utilizando as fórmulas descritas acima. A primeira posição, quando a variável A do ciclo está em zero, ficará no topo do círculo. Posições subsequentes aparecerão seguindo o círculo no sentido dos ponteiros do relógio, em incrementos de seis graus.

Linhas 4060-4070: tendo estabelecido uma posição ao longo do exterior do círculo, estas linhas desenham uma linha interior, em direcção ao centro. O que as linhas fazem, de facto, é desenhar uma linha da circunferência de um círculo para a circunferência de um outro mais pequeno, no interior daquele. O círculo exterior tem um raio de 200 *pixels,* enquanto a dimensão do círculo interior dependerá de o ângulo apontado pela variável A do ciclo apontar ou não para cinco minutos completos (30 graus). Se o ângulo não apontar para cinco minutos completos, então o círculo interior ficará apenas cinco *pixels* para o interior do círculo original — a marca desenhada terá cinco *pixels* de comprimento.

A utilização da função MOD assegura que o programa saiba quando é atingido um ângulo de 30 graus. 5 MOD 2, por exemplo, dá o resultado 1, ou o que fica (o resto) da divisão de 5 por 2. A MOD 30, tal como no programa, dá o resto da divisão da variável A do ciclo por 30 — quando este é zero, então o ângulo é exactamente divisível por 30. Sempre que isto acontecer, o círculo interior ficará mais pequeno ou, dito de outra forma, a marca desenhada será mais comprida, salientando assim as marcações de cinco minutos.

Ensaio

Introduza

GOTO 4000 [ENTER]

e deverá ver o écran ser limpo, aparecendo o mostrador de um relógio. O programa terminará então, com uma mensagem de erro «Unexpected RETURN».

Módulo 1.1.5: Ajuste dos minutos e horas

Neste módulo passamos ao verdadeiro trabalho do programa, que é o de calcular o tempo corrente e, nessa base, as coordenadas para os ponteiros do relógio. Este módulo não poderá ser utilizado convenientemente até que seja introduzido o último dos módulos, já que só então ele será chamado nos intervalos próprios, ou seja, uma vez em cada minuto.

Módulo 1.1.5: linhas 5000-5130

```
5000 REM*********************************
5010 REM Adjust Time
5020 REM*********************************
5030 c=0
5040 angle=minute*6:size=180:GOSUB 6000
5050 angle=hour*30+minute/2:size=120:GOS
UB 6000
5060 minute=minute+1
5070 IF minute=60 THEN minute=0:hour=hou
r+1
5080 IF hour=13 THEN hour=1
5090 c=2
5100 angle=minute*6:size=180:GOSUB 6000
5110 c=3
5120 angle=hour*30+minute/2:size=120:GOS
UB 6000
5130 RETURN
```

Comentário

O módulo encontra-se dividido, na verdade, em três secções distintas. A tarefa da primeira secção é chamar um módulo subsequente, para que ele apague os ponteiros na posição em que existem. A segunda secção adiciona 1 a MINUTE e faz qualquer alteração consequente a HOUR. A terceira secção pega nos novos valores de hora/minuto e chama um módulo subsequente para redesenhar os ponteiros.

Linhas 5030-5050: a variável C será utilizada por um módulo subsequente para estabelecer a cor da tinta (INK). Em primeiro lugar é colocada em 0. Se fizer uma retrospectiva do módulo de inicialização verificará que a cor 0, a cor de fundo, foi definida para azul. A linha 5040 estabelece os parâmetros para o ponteiro dos minutos, designadamente o ângulo em que deve ser desenhado e a dimensão. Uma vez que o valor do minuto não tenha sido alterado, o desenho do ponteiro na cor de fundo apagará, de facto, o ponteiro existente. A linha 5050 faz o mesmo trabalho relativamente ao ponteiro das horas.

Linhas 5060-5080: os valores de minuto e hora são actualizados, pela adição de 1 aos minutos, fazendo-se depois quaisquer ajustamentos para assegurar que os valores resultantes sejam razoáveis.

Linhas 5090-5120: os parâmetros dos ponteiros de hora e minuto são estabelecidos de novo, utilizando os valores actualizados, antes da chamada do módulo seguinte para os retirar. O número de cor C é posto em 2, para o ponteiro dos minutos, e em 3, para o ponteiro das horas. Se voltar a observar o módulo de inicialização, verá que isto corresponde, respectivamente, a vermelho e vermelho brilhante.

Ensaio

Escreva:

6000 RETURN[ENTER]

Esta linha é temporária e tem a seu cargo assegurar que o módulo acabado de introduzir solicita um que ainda não exista.

Agora, escreva:

HOUR = 0[ENTER]
MINUTE = 59[ENTER]
GOTO 5000[ENTER]

e deverá verificar, quase instantaneamente, que a execução pára, com uma mensagem de erro «Unexpected RETURN». Escreva agora:

Deverá ver que o que foi impresso é:

0 1

reflectindo o facto de os seus 59 minutos originais terem sido aumentados para 60, sendo o valor da hora acrescido em concordância com esse facto.

Módulo 1.1.6: Desenho dos ponteiros

Após termos calculado todos os valores necessários para chegarmos à posição dos ponteiros, podemos passar agora a desenhar os ponteiros do relógio.

Módulo 1.1.6: linhas 6000-6110

```
6000 REM********************************
6010 REM Draw Hand
6020 REM********************************
6030 size2=size*2/3
6040 PLOT 0,0,c
6050 DRAW size2*SIN(angle-8),size2*COS(a
ngle-8)
```

```
6060 DRAW size*SIN(angle),size*COS(angle
)
6070 DRAW size2*SIN(angle+8),size2*COS(a
ngle+8)
6080 DRAW 0,0
6090 MOVE size2*SIN(angle),size2*COS(ang
le)
6100 GOSUB 7000
6110 RETURN
```

Comentário

Linha 6030: os ponteiros a desenhar terão a forma de um losango muito alongado. SIZE 2 representa a distância ao centro a partir da qual o ponteiro começará a afunilar em direcção à ponta.

Linha 6040: o cursor de gráficos é deslocado até ao centro do mostrador e a cor de desenho regulada para C, permitindo assim ao módulo anterior ditar a cor ou o que for desenhado.

Linhas 6050-6080: estas linhas desenham quatro traços no écran, os quais definem o ponteiro. A primeira linha tem o comprimento SIZE 2 e é desenhada a um ângulo de oito graus em sentido inverso ao dos ponteiros do relógio até ao ângulo correcto para a hora e minuto. A linha seguinte é desenhada a partir do fim da primeira linha, até uma posição que se encontra no ângulo e *pixels* SIZE correctos a partir do centro. A terceira linha é desenhada até uma posição distante do centro SIZE 2 *pixels* e oito graus no sentido dos ponteiros do relógio até ao ângulo correcto. Finalmente, a forma é completada desenhando uma linha de retorno ao centro.

Linhas 6090-6100: estas duas linhas são necessárias para os objectivos do próximo módulo, que são colorir as formas vazias criadas por este módulo. A linha operativa é 6090, a qual desloca o cursor de gráficos para um ponto claramente dentro da forma do ponteiro desenhado.

Ensaio

Em primeiro lugar, tem de ser adicionada uma linha temporária para se atingir o módulo seguinte:

```
7000 RETURN
```

Agora, escreva:

CLEAR:CLS:DEG:GOTO 5000[ENTER]

e deverá ver o écran limpar, bem como a impressão dos dois ponteiros do relógio, sensivelmente às doze e um minuto. Os ponteiros estarão apenas em esboço, já que o processo de preenchimento com cor é efectuado pelo próximo módulo.

Módulo 1.1.7: Preenchimento de uma forma com cor

Uma das poucas deficiências do *464,* se comparado com outros computadores pessoais, é a falta de um comando para preenchimento de uma forma esboçada com cor. O módulo aqui fornecido é bastante mais lento do que um comando incorporado e tem algumas limitações, ainda que execute o seu trabalho, pelo que vale a pena tê-lo.

A rotina, tal como vem listada neste módulo, preencherá qualquer polígono cujos ângulos sejam côncavos, quando vistos do exterior, ou seja, cujos ângulos apontem para fora e não para dentro. Preencherá *algumas* formas cujos ângulos apontem para dentro, mas não todas. Se realmente pretende utilizar a rotina em formas tão irregulares, então terá de preenchê-las por secções. Note, especialmente, que se a rotina for utilizada numa forma que não esteja fechada, provavelmente trancará enquanto procura pelo écran os limites da figura.

Módulo 1.1.7: linhas 7000-7300

```
7000 REM*********************************
7010 REM Fill
7020 REM*********************************
7030 IF TESTR(0,0)=c THEN RETURN
7040 s=2
7050 MOVE 2*INT(XPOS/2),2*INT(YPOS/2)
7060 :
7070 :
7080 REM search up/right ***************
7090 IF TESTR(0,s)<>c THEN GOTO 7090
7100 IF TESTR(s,-s)<>c THEN GOTO 7090
7110 :
7120 :
7130 REM search up/left ****************
7140 MOVER -s,0
7150 IF TESTR(0,s)<>c THEN GOTO 7090
7160 IF TESTR(-s,-s)<>c THEN GOTO 7150
7170 :
7180 :
7190 REM ink in lines across ***********
7200 x1=XPOS
7210 MOVER s,0
7220 PLOTR 0,0,c
7230 IF TESTR(s,0)<>c THEN GOTO 7220
7240 x2=XPOS-s
7250 MOVE x1,YPOS-s
7260 IF TESTR(s,0)<>c THEN GOTO 7290
7270 IF XPOS=x2 THEN RETURN
```

```
7280 GOTO 7260
7290 IF TESTR(-s,0)=c THEN GOTO 7200
7300 GOTO 7290
```

Comentário

Linha 7030: a rotina, tal como acontece com a maioria das outras rotinas de «pintura» ou «preenchimento», não funcionará se a cor do *pixel* onde for instruída para começar já for a cor que irá utilizar no preenchimento.

Linha 7040: a variável S regista o mínimo avanço horizontal ocorrido no decurso do programa. O seu valor pode ser reinicializado se estiverem a ser utilizados outros gráficos.

Linha 7050: em modo gráfico 1, que temos estado a utilizar para este programa, os *pixels* são agrupados por pares. Esta linha assegura que eles sejam agora fixos em números pares, independentemente do que possa ter acontecido às coordenadas de desenho.

Linhas 7080-7100: começa agora a busca de uma parte da margem da figura. Estas linhas podem parecer estranhas, de início, já que parece não se registar movimento, a não ser uma linha que tranca num ciclo infinito. De facto, tanto TEST como TESTR(elativo) deslocam o cursor de gráficos para o local especificado. Assim, a primeira linha faz a busca para cima, em linha recta, até ao primeiro *pixel* da cor correcta. Uma vez encontrada parte da margem, a linha seguinte retrocede um passo para baixo e para a direita, para verificar se esse ponto está vazio. Se *estiver* vazio, então a primeira linha é chamada de novo, para ver se é possível avançar mais para cima.

Linhas 7130-7160: após terem procurado para cima e para a direita em toda a extensão, estas linhas começam a procurar para a esquerda, em busca de um caminho que as conduza mais acima. No final destas linhas e do grupo anterior, o programa terá encontrado o topo da figura, ou então terá chegado a um beco sem saída, já que a figura não é regular.

Linhas 7200-7230: depois de ter encontrado aquele que será o ponto mais alto, o processo de «preenchimento» pode começar em pleno. Os comandos MOVER(elativo) e PLOTR(elativo) são utilizados para desenhar uma linha de *pixels,* da esquerda para a direita, até ser atingida outra parte da margem.

Linhas 7240-7250: após ter terminado uma linha horizontal, a variável X2 é igualada em valor à posição preenchida mais à direita e a posição do cursor é deslocada de volta ao princípio da linha, embora um *pixel* abaixo.

Linhas 7260-7280: esta verificação é efectuada quando o cursor é deslocado novamente para a esquerda. Se o *pixel* atingido já estiver colorido na cor do preenchimento, o programa faz um varrimento da esquerda para a direita, procurando o primeiro *pixel* em branco. Se a busca continuar para além do final da linha acima, a qual acabou de ser preenchida, então foi atingido o extremo inferior da figura.

Linhas 7290-7300: se, ao mover o cursor para baixo e para a esquerda para começar um nova linha, se atingir um *pixel* vazio, o programa procura para a esquerda até chegar à margem da figura, antes de começar a preenchê-la.

Ensaio

O ensaio mais simples para este módulo é seguir o procedimento de ensaio do módulo anterior. A única diferença no resultado deverá ser que, agora, ambos os ponteiros se encontram preenchidos com a cor correcta.

Módulo 1.1.8: Pondo tudo a funcionar

Neste ponto, deve estar a perguntar por que razão o programa está escrito como está e se todas as funções que descrevemos não poderiam ter sido aglomeradas e postas a funcionar com a ajuda de alguns GOTO. Infelizmente, é assim que muitos dos programas publicados se encontram construídos.

Neste livro, verificará que todos os programas estão construídos a partir de módulos claramente identificáveis. A razão disto reside no facto de os programas escritos em módulos poderem ser lidos mais facilmente, mais facilmente depurados, de poderem ser alterados pela substituição de módulos que trabalhem mais eficientemente se se aprenderem novos métodos, e de poderem ser acrescentados com a introdução de mais módulos. Muito há a aprender com estes programas, mas a lição mais valiosa de todas será, provavelmente, para as suas futuras programações, a técnica da programação modular.

O corrente módulo é a chave para essa técnica, pois que, quando todos os módulos funcionais tiverem sido introduzidos e ensaiados, precisaremos de mais um para controlar o fluxo do programa. De certa forma, o módulo que você está prestes a introduzir *é* o programa — tudo o mais é uma extensão dele.

Módulo 1.1.8: linhas 1000-1100

```
1000 REM********************************
1010 REM Control
1020 REM********************************
1030 GOSUB 2000
1040 GOSUB 3000
1050 GOSUB 4000
1060 GOSUB 5000
1070 EVERY 3000 GOSUB 5000
1080 IF INKEY$<>" " THEN 1080
1090 CLS
1100 END
```

Comentário

Linha 1070: é a chave do programa. Esta é a linha única que fornece ao programa a noção de tempo, utilizando o poderoso comando EVERY do *464*. A função deste comando é permitir ao utilizador especificar um período de tempo e, sempre que esse período de tempo se tenha esgotado, interromper qualquer tarefa em decurso, para realizar uma outra coisa. Terminada a interrupção, a tarefa original, seja ela qual for, é recomeçada até à interrupção seguinte. O período após o qual ocorrerá a interrupção é cronometrado em quinquagésimos de segundo, pelo que esta linha diz ao *464* para chamar o módulo na linha 5000 uma vez por minuto, aumentando assim o valor do minuto e redesenhando os ponteiros.

Linhas 1080-1100: durante o resto do tempo, o programa trancará no ciclo infinito representado pela linha 1080, aguardando que o utilizador prima a barra de espaços. Logo que tal aconteça, o écran limpa e termina o programa.

Ensaio

O programa deverá agora funcionar em pleno. Execute-o, introduza o tempo e verá o mostrador do relógio visualizado, com os ponteiros na posição correcta. Pode efectuar uma verificação mais rápida do funcionamento da visualização apagando dois zeros no valor 3000, na linha 1070. Isto fará que os ponteiros sejam constantemente desenhados e redesenhados.

PROGRAMA 1.2: RELÓGIO

Função do programa

Uma das coisas mais agradáveis acerca dos computadores com visualizações gráficas tão boas como as do *CPC 464* é que lhes permitem recrear-se na visualização de coisas, de forma nova e imaginativa. Após termos introduzido um relógio bastante normal, este próximo programa dá-lhe uma perspectiva bastante diferente do tempo. Em «Relógio», horas e minutos são representados por duas linhas que deslizam da esquerda para a direita e de cima para baixo, dividindo o écran em quatro rectângulos de diferentes cores. Muito do material de «Relógio» é semelhante ao de «Anarelógio», pelo que as explicações podem ser abreviadas.

Novos tópicos abordados no programa:

1) Janelas.
2) Formatação de números.
3) Corte de cadeias.

Módulo 1.2.1: Inicialização

Trata-se de um simples módulo de inicialização para estabelecer as cores necessárias à visualização. O objectivo da janela estruturada na linha 2100 será explicado na secção «Comentário» de um próximo módulo.

Módulo 1.2.1: linhas 2000-2120

```
2000 REM********************************
2010 REM Initialise
2020 REM********************************
2030 MODE 1
2040 INK 0,1
2050 INK 1,24
2060 INK 2,9
2070 INK 3,6
2080 BORDER 1
2090 PAPER 0:PEN 1
2100 WINDOW #1,39,39,9,18
2110 PAPER #1,0:PEN #1,1
2120 RETURN
```

Ensaio

Tal como no primeiro programa, o único ensaio que pode fazer-se nesta altura é escrever:

GOTO 2000[ENTER]

o que deverá resultar no fim do programa, com uma mensagem de erro «Unexpected RETURN». Qualquer outro erro indicará que foi cometida uma falta de introdução do módulo.

Módulo 1.2.2: Introdução do tempo

O mesmo módulo que para o «Anarelógio».

Módulo 1.2.2: linhas 3000-3090

```
3000 REM********************************
3010 REM Set Time
3020 REM********************************
3030 PRINT "Clock Setting:":PRINT
3040 INPUT "Hour (1-12):";hour
```

```
3050 IF hour<1 OR hour>12 THEN PRINT "**
* Out of range: please try again ***":GO
TO 3040
3060 INPUT "Minute (0-59):";minute
3070 IF minute<0 OR minute>59 THEN PRINT
 "*** Out of range: please try again ***
":GOTO 3060
3080 second=0
3090 RETURN
```

Módulo 1.2.3: Organização da margem do écran

Este módulo imprime uma grelha para a hora e minuto ao longo da faixa lateral esquerda e superior do écran.

Módulo 1.2.3: linhas 4000-4120

```
4000 REM*********************************
4010 REM Set Up Display
4020 REM*********************************
4030 CLS:PRINT"   ";
4040 FOR i=5 TO 55 STEP 10
4050 PAPER 0:PEN 1:PRINT USING"###";i;
4060 PAPER 1:PEN 0:PRINT USING"###";i+5;
4070 NEXT i
4080 PAPER 0:PEN 1
4090 FOR i=1 TO 12
4100 LOCATE 1,i*2+1:PRINT USING"##";i
4110 NEXT i
4120 RETURN
```

Comentário

Linhas 4040-4070: o ciclo imprime os valores de minuto espaçados ao longo da parte superior do écran, alternando entre inscrição normal e invertida (cores do papel e caneta trocadas). PRINT USING é empregue para assegurar que cada marca de cinco minutos seja impressa utilizando três caracteres — visto que todos os números têm menos de três caracteres de extensão, PRINT USING acrescentá-los-á com um ou dois espaços antes.

Linhas 4080-4110: a variável de ciclo I é utilizada conjuntamente com o comando LOCATE, para imprimir as horas ao longo da margem esquerda do écran.

Ensaio

Para que se efectue um eficiente ensaio do módulo, precisará de introduzir uma linha temporária:

4115 GOTO 4115

O objectivo desta é evitar que o écran faça o desenrolamento *(scrolling)* para cima, imprimindo simultaneamente a mensagem de erro «Unexpetecd RETURN». Introduza agora:

GOTO 4000[ENTER]

e deverá ver a grelha impressa em volta dos limites do écran. Prima ESCAPE duas vezes quando estiver satisfeito... e não se esqueça de remover a linha temporária.

Módulo 1.2.4: Criação de uma cadeia de tempo

Uma das funções do programa será fazer a visualização do tempo, de modo integralmente digital e também usando o método mais singular descrito anteriormente. A cadeia baseia-se em variáveis que serão criadas pelo próximo módulo, embora este deva ser inserido em primeiro lugar para possibilitar a execução correcta do seguinte.

Módulo 1.2.4: linhas 7000-7050:

```
7000 REM*********************************
7010 REM Create Time String
7020 REM*********************************
7030 tim$=STR$(1000000+hour*10000+minute
*100+second)
7040 tim$=MID$(tim$,3,2)+"  "+MID$(tim$,
5,2)+"  "+MID$(tim$,7,2)
7050 RETURN
```

Comentário

Linhas 7030-7040: um expediente simples para combinar séries de números num só número é multiplicar cada um por decrescentes potências de 10, adicionando-os depois com uma maior potência de 10. Se, por exemplo, HOUR for igual a 1, MINUTE = 36 e SECOND = 2, o resultado desta linha seria 1013602. Porquê? Bem,

vejamos o valor da hora — o número 1000000 que foi adicionado colocou um 0 em frente do simples «1» correspondente à hora, tal como aconteceu com o «2» dos segundos.

Utilizamos agora aquilo que é conhecido por «divisão de cadeia», cortando três secções de dois dígitos do número, confiantes em que foi adicionado um 0 precedente, caso algum dos valores originais fosse um único dígito. As funções de cadeia na segunda linha extraem os três algarismos especificando a posição de partida no número criado e o número de caracteres a serem extraídos. Assim, por exemplo, **MID$(A$,3,2)** representa a parte de **A$** que se inicia no carácter 3 e possui dois caracteres de extensão.

Se observar os comandos da divisão de cadeia, na linha 7040, poderá ficar confuso com o facto de o valor da hora ser retirado do carácter 3 para a frente, quando seguramente se inicia o dígito 2 do número artificial criado na linha 7030. A resposta a esta aparente contradição reside no facto de que, para extrair o valor sob a forma de cadeia, transformámos primeiro o número numa cadeia que utiliza a função **STR$**. Ao fazermos isto, o número fica exactamente com a mesma aparência, mas pode agora ser tratado como uma cadeia — a única diferença é que, se for um número positivo, é acrescentado um espaço para compensar a falta de um sinal « + ». O primeiro carácter utilizável de um número que foi transformado em cadeia aplicando **STR$** é o carácter número 2.

O resultado desta obra de manipulação consiste no facto de acabarmos com uma curta cadeia, sob a forma de:

TWO DIGIT HOUR/SPACE/TWO DIGIT MINUTE/SPACE/
TWO DIGIT SECOND

Ensaio

Escreva:

HOUR = 1[ENTER]
MINUTE = 1[ENTER]
SECOND = 1[ENTER]
GOTO 7000[ENTER]

Quando o programa lhe responder com um erro «Unexpetecd RETURN», escreva:

PRINT tim$[ENTER]

e deverá ver:

010101

Módulo 1.2.5: Ajuste do tempo

Este módulo é paralelo ao módulo de ajuste do tempo, no programa anterior.

Módulo 1.2.5: linhas 6000-6160

```
6000 REM*********************************
6010 REM Adjust Time
6020 REM*********************************
6030 second=second+1
6040 WHILE second>59
6050 second=0:minute=minute+1
6060 WHILE minute>59
6070 minute=0:hour=hour+1
6080 WHILE hour>12
6090 hour=1
6100 WEND
6110 WEND
6120 WEND
6130 LOCATE #1,1,1
6140 GOSUB 7000
6150 PRINT#1,tim$;
6160 RETURN
```

Comentário

Linhas 6030-6120: estes três ciclos aninhados atravessam claramente a sequência de passos que podem ser necessários quando o segundo valor for aumentado em 1. Cada ciclo sucessivo apenas será activado se, seguindo a alteração dos segundos, o valor do minuto ou da hora tiver de ser ajustado... isto é, se o limite final do minuto ou da hora tiver sido ultrapassado.

Linhas 6130-6160: após a criação de uma cadeia que representa o tempo, no módulo anterior, estas linhas imprimem-se verticalmente. A utilização da janela definida no módulo de inicialização torna esta questão mais simples. Se se recordar o módulo de inicialização, verificar-se-á que a janela definida fica no lado direito do écran e que *apenas tem um carácter de amplitude.* Resulta daqui que tudo o que for impresso nessa janela aparecerá com cada carácter sucessivo por baixo do anterior.

Ensaio

Escreva:

```
CLEAR
SECOND = 59
MINUTE = 59
HOUR = 12
GOTO 6000[ENTER]
```

e deverá ser confrontado com uma mensagem «Unexpetecd RETURN», aparecendo «01 00 00» impresso verticalmente, na direcção da margem direita do écran, demonstrando que os ciclos na primeira metade do programa são capazes de actualizar todos os três valores correctamente.

Módulo 1.2.6: Visualização do tempo

Neste módulo, atacamos a tarefa de visualizar no écran os rectângulos que representarão o tempo. Aquilo que o módulo pretende alcançar é o efeito de uma linha varrendo o écran, para representar os minutos, e uma outra descendente, que registe as horas. Isto é conseguido dividindo o écran em quatro secções rectangulares, sendo duas vermelhas e duas verdes, e representando os limites destes rectângulos as linhas para as horas e minutos, tal como na figura 1.2.

Sem dúvida que seria possível colocar cada um destes rectângulos directamente no écran, no lugar correcto, mas isso é desnecessariamente difícil. Tudo o que é realmente preciso fazer é desenhar na metade superior do écran, em sentido horizontal, linhas de espaços coloridos que mudem de vermelho para verde no ponto correcto, e, na metade inferior do écran, linhas que mudem de verde para vermelho. Quanto às próprias linhas, serão criadas por um simples ciclo e pelo comando LOCATE.

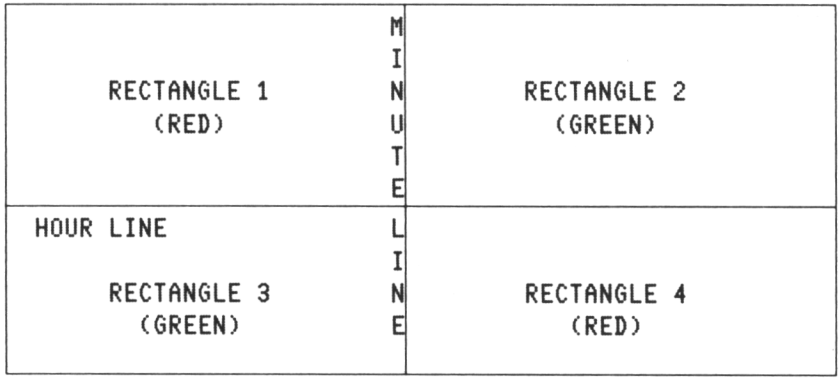


Fig. 1.2 — *Divisão do écran em quatro secções rectangulares*

Módulo 1.2.6: linhas 5000-5100

```
5000 REM*********************************
5010 REM Display Time
5020 REM*********************************
5030 PAPER 2:PEN 3
5040 FOR i=2 TO 25
5050 IF i=INT(hour*2+minute/30+2) THEN P
APER 3:PEN 2
5060 LOCATE 3,i
5070 PRINT STRING$(minute*0.6,CHR$(143))
;TAB (39)
5080 NEXT i
5090 RETURN
5100 RETURN
```

Comentário

Linha 5030: ao papel é atribuída a cor verde e à impressão a vermelha.

Linhas 5040-5080: serão impressas vinte e quatro linhas de visualização, representando cada linha meia hora e começando a visualização na linha 2 do écran (numerado a partir de 0).

Linha 5050: se o valor do ciclo tiver atingido a posição em que a linha da hora deve estar, as cores do papel e da impressão serão invertidas.

Linhas 5060-5070: estas duas linhas localizam e imprimem uma das linhas longitudinais à visualização. A função STRING$ é usada para criar uma linha de espaços inversos (CHR$(143)) até à linha divisória dos minutos. A palavra TAB no fim da linha desloca a posição de impressão para o final da linha, atribuindo a cor de papel corrente a quaisquer espaços não utilizados. A extensão da cadeia criada basear-se-á no valor da variável MINUTE, embora sendo ajustada (multiplicada por 0,6), para assegurar que a visualização terá 36 caracteres de amplitude, e não 60.

Ensaio

Introduza, em modo directo:

```
HOUR = 6[ENTER]
MINUTE = 30[ENTER]
GOTO 5000
```

Deverá ver o écran dividido em quatro rectângulos sensivelmente iguais, ao longo das linhas da ilustração supra. O programa parará então, com uma mensagem de erro «Unexpetecd RETURN».

Módulo 1.2.7: Juntando tudo

Após ter introduzido todos os elementos funcionais do programa, podemos agora construir um módulo de controlo para as executar pela ordem correcta.

Módulo 1.2.7: linhas 1000-1090

```
1000 REM********************************
1010 REM Control
1020 REM********************************
1030 ON BREAK GOSUB 1090
1040 GOSUB 2000
1050 GOSUB 3000
1060 EVERY 50 GOSUB 6000
1070 GOSUB 4000
1080 GOSUB 5000:GOTO 1080
1090 PEN 1 : PAPER 0 : CLS : CLEAR : END
```

Comentário

Linha 1030: o comando ON BREAK GOSUB permite-nos definir o que o programa fará se a tecla ESCAPE for premida duas vezes. Neste caso, queremos que ele salte até ao fim do programa, onde fará o *reset* das cores, limpará a memória e parará.

Linha 1060: uma vez mais, o comando EVERY fornece a chave da temporização do programa. Esta chamada do módulo de ajuste do tempo é feita de segundo a segundo.

Ensaio

Está agora em posição de executar todo o programa, introduzir o tempo e vê-lo apresentado no écran.

PROGRAMA 1.3: TEMPORIZADOR

Função do programa

Este programa fornece-lhe dezasseis flexíveis temporizadores de contagem decrescente, cada um dos quais pode ser programado separadamente para fazer soar um alarme, após um período de tempo especificado, *e* visualizar uma curta mensagem indicativa da finalidade da ocorrência do alarme.

Quando observar o programa, notará que a sua apresentação é diferente da dos programas anteriores. Visto que se trata do programa mais longo e mais complexo até agora encontrado neste livro, foram inseridas linhas programáticas em branco

para o ajudar, neste caso único, a identificar as unidades funcionais que constituem o programa. As linhas em branco não são necessárias ao funcionamento do programa e podem ser omitidas, se assim o desejar. O programa é um dos poucos a ser apresentado desta forma, pela simples razão de que o espaço extra requerido para destacar cada programa acarretaria que se tivesse de retirar material ao livro.

```
Time now: 06:30:10

          TIMERS
          ======

1) 07:00:00 Wake up
2) 07:15:00 I said wake up
3) 07:30:00 This is your last chance
4) 08:00:00 You missed the train
5) 17:45:00 Tom and Jerry on TV
6) 20:30:00 Phone Grandma
```

Fig. 1.3 — *Visualização típica, retirada do «Temporizador»*

Novas técnicas abordadas neste programa:

1) O módulo do *menu*.
2) A utilização simples do SOUND (som).
3) Inserção e apagamento em *arrays* (quadros).
4) Criação de um estado de espera.

Módulo 1.3.1: Inicialização

É um módulo mais complexo do que aqueles que foram empregues pelos dois programas precedentes, reflectindo a maior complexidade do programa no seu todo. A maior parte do módulo relaciona-se com a tarefa de montar o écran, incluindo duas janelas, para utilização do restante programa.

Módulo 1.3.1: linhas 2000-2150

```
2000 REM********************************
2010 REM Initialise
2020 REM********************************
2030 MODE 1
2040 GOSUB 12000
2050 WINDOW 1,40,25,25
2060 PAPER 0:PEN 1:CLS
2070 WINDOW #1,12,29,1,1
2080 PAPER #1,0:PEN #1,1:CLS #1
2090 PRINT #1,"Time now:"
```

```
2100 WINDOW #2,1,40,3,23
2110 PAPER #2,3:PEN #2,2:CLS #2:PRINT #2
2120 SPEED INK 50,25
2130 DIM thour(15),tminute(15),ttime(15)
,message$(15)
2140 ttime(0)=-1
2150 RETURN
```

Comentário

Linha 2040: em contraste com programas anteriores, este módulo não conterá quaisquer instruções INK. Isto prende-se com o facto de as cores INK (tinta), em certas ocasiões, virem a mudar, no decurso do programa. Este GOSUB chama a pequena rotina que determina as cores iniciais: esta sub-rotina será introduzida imediatamente a seguir à rotina corrente.

Linha 2050: o comando WINDOW (janela) pode ser usado de duas maneiras, quer para reservar uma área de écran que possa ser chamada utilizando um «número de fluxo» — um método que foi utilizado no programa «Relógio» — quer simplesmente especificando a dimensão da janela. Isto define a janela ausente, ou aquela em que é feita normalmente a impressão. A janela ausente é aqui definida como uma única linha, ao fundo do écran.

Linhas 2070-2120: estas linhas definem mais duas janelas, uma delas (# 1) sendo uma única linha no topo do écran e a outra (# 2) o corpo principal do écran. O comando SPEED INK é utilizado quando um número INK possui duas cores atribuídas e especifica a velocidade *(speed)* a que as letras pintadas nessa tinta cintilam *(flash)* entre as duas cores.

Linha 2130: são os quadros em que o programa irá trabalhar. A sua utilidade será descrita no decurso do comentário sobre a parte principal do programa.

Módulo 1.3.2: Cores do écran para o início

Tal como foi explicado no comentário ao último módulo, as cores do écran podem modificar-se durante o programa. Este pequeno módulo estabelece as cores iniciais.

Módulo 1.3.2: linhas 12000-12080

```
12000 REM*******************************
12010 REM Turn Screen On
12020 REM*******************************
12030 INK 0,1
```

```
12040 INK 1,24
12050 INK 2,26
12060 INK 3,2
12070 BORDER 1
12080 RETURN
```

Módulo 1.3.3: Marcação do tempo

É o módulo padrão encontrado no «Anarelógio», com a excepção de que o tempo é introduzido de acordo com o relógio de 24 horas.

Módulo 1.3.3: linhas 3000-3100

```
3000 REM*********************************
3010 REM Set Time
3020 REM*********************************
3030 CLS:PRINT"Clock Setting:":PRINT
3040 INPUT "Hour (0-23):";hour
3050 IF hour<0 OR hour>23 THEN PRINT "**
* Out of range: please try again ***":GO
TO 3040
3060 INPUT "Minute (0-59):";minute
3070 IF minute<0 OR minute>59 THEN PRINT
 "*** Out of range: please try again ***
":GOTO 3060
3080 second=0
3090 tim=hour*60+minute
3100 RETURN
```

Módulo 1.3.4: Ajuste do tempo

É um módulo semelhante aos anteriores módulos de ajuste do tempo.

Módulo 1.3.4: linhas 14000-14180

```
14000 REM*********************************
14010 REM Adjust Time
14020 REM*********************************
14030 second=second+5
14040 WHILE second>59
14050 second=0:minute=minute+1
14060 WHILE minute>59
14070 minute=0:hour=hour+1
14080 WHILE hour>23
14090 hour=0
14100 WEND
```

```
14110 WEND
14120 WEND
14130 LOCATE#1,11,1
14140 h=hour:m=minute:s=second:GOSUB 150
00
14150 PRINT#1,tim$;
14160 tim=hour*60+minute
14170 GOSUB 9000
14180 RETURN
```

Módulo 1.3.5: Formatação do tempo

Tal como no programa anterior, o módulo de ajuste do tempo funciona em conjugação com este, que transforma o tempo corrente numa forma compreensível pelo utilizador.

Módulo 1.3.5: linhas 15000-15050

```
15000 REM*******************************
15010 REM Create Time String
15020 REM*******************************
15030 tim$=STR$(1000000+h*10000+m*100+s)
15040 tim$=MID$(tim$,3,2)+":"+MID$(tim$,
5,2)+":"+MID$(tim$,7,2)
15050 RETURN
```

Módulo 1.3.6: Ensaio dos temporizadores e activação do alarme

Este módulo é parte integrante do programa, de modo que permite que outras partes do programa introduzam um estado de espera, durante o qual o utilizador pode fazer uma introdução de dados. No entanto, e simultaneamente, o programa está constantemente a efectuar o trabalho de selecção dos temporizadores, para verificar se há situação para um alarme. Se for necessário, fá-lo-á soar. Vamos agora introduzir o módulo, ainda que não seja integralmente utilizado até que vários módulos posteriores tenham sido introduzidos, porque constitui uma sub-rotina essencial ao módulo de *menu* que se segue.

Módulo 1.3.6: linhas 9000-9300

```
9000 REM*******************************
9010 REM Check Timer
9020 REM*******************************
9030 IF ttime(0)<>tim THEN RETURN
9040 CLS #2:PRINT #2
9050 GOSUB 12000:INK 2,26,2
9060 PRINT #2,TAB(19);"ALARM";TAB(19);"=
```

```
====":LOCATE #2,1,11
9070 mlen=LEN(message$(0))
9080 PRINT #2,TAB(19-mlen/2);STRING$(mle
n+4,"*")
9090 PRINT #2,TAB(19-mlen/2);"*";TAB(22+
mlen/2);"*"
9100 PRINT #2,TAB(19-mlen/2);"* ";messag
e$(0);" *"
9110 PRINT #2,TAB(19-mlen/2);"*";TAB(22+
mlen/2);"*"
9120 PRINT #2,TAB(19-mlen/2);STRING$(mle
n+4,"*")
9130 note=250
9140 WHILE INKEY$="" AND ttime(0)=tim
9150 WHILE (SQ(1) AND 7)>0
9160 SOUND 1,note,15,15
9170 note=750-note
9180 WEND
9190 WEND
9200 sounded=1
9210 FOR i=1 TO timers-1
9220 thour(i-1)=thour(i)
9230 tminute(i-1)=tminute(i)
9240 ttime(i-1)=ttime(i)
9250 message$(i-1)=message$(i)
9260 NEXT i
9270 timers=timers-1
9280 IF timers=0 THEN ttime(0)=-1
9290 INK 2,26
9300 RETURN
```

Comentário

Linha 9030: é a nossa primeira referência funcional ao quadro TTIME, o qual é utilizado para armazenar os tempos a que os temporizadores devem expirar. Módulos subsequentes, que permitirão ao utilizador regular os temporizadores, assegurarão que o temporizador inicial seja sempre colocado na primeira linha do quadro, no elemento 0. Na grande maioria dos casos em que esta sub-rotina é chamada, a execução terminará nesta linha. Só se o tempo corrente, TIM (que foi calculado pelo módulo de ajuste do tempo), for o mesmo que o tempo de alarme armazenado na primeira linha de TTIME é que o alarme soará.

Linhas 9040-9120: estas linhas parecem bastante mais complexas do que são, de facto. O seu objectivo é imprimir a palavra «ALARM» na janela # 2 (a parte principal do écran), juntamente com qualquer mensagem associada ao temporizador específico. A mensagem será rodeada por uma margem de asteriscos e será centrada no écran pela impressão do primeiro carácter na posição determinada pela fórmula:

MIDDLE POSITION ON SCREEN – LENGTH OF STRING TO PRINT / 2

Pode ver esta fórmula empregue nas linhas 9090-9120, onde assume a forma «90-mlen/2».

Linhas 9130-9200: estas linhas fazem soar (SOUND) um alarme de dois tons pelo período de um minuto — enquanto o tempo-limite do temporizador for o mesmo do tempo corrente em minutos. O alarme cessa se for premida uma tecla. O mais interior dos dois ciclos toca a nota efectiva, assegurando a linha 9150 que cada nota seja retida até que a função SQ indique que a nota anterior parou de soar. À variável SOUNDED é atribuído o valor 1, para indicar ao módulo do *menu* que foi despoletado um alarme. Veja o comentário sobre o *menu* para saber da razão subjacente a isto.

Linhas 9210-9280: após ter soado um alarme, estas linhas retiram-no dos quadros que armazenam os tempos e mensagens para os temporizadores. Isto é efectuado deslocando todos os dados do segundo temporizador em diante um espaço para baixo, obliterando assim o primeiro item, o qual acabou de soar. A variável TIMERS regista a quantidade de temporizadores correntemente regulados e, desta forma, deve ser reduzida em 1. Finalmente, se já não restarem temporizadores, a linha 9280 assegura a colocação de um tempo impossível na posição do primeiro temporizador, para que ele não desperte a uma hora indesejada.

Módulo 1.3.7: O «menu» do programa

Chegamos agora a uma técnica que desempenhará um vasto papel nos programas incluídos neste livro — o «*menu*». Nos programas que conduziram a este, o controlo do programa foi deixado para ele próprio. Uma vez executado (RUN), um módulo de controlo assumiu o comando e ditou o fluxo de execução do programa, até à indicação para terminar por parte do utilizador.

Este programa (e muitos dos que se seguem) é diferente, já que não há uma única directiva quanto ao fluxo do programa. O programa apresenta uma variedade de possibilidades ao utilizador, e deverá ser este que, em grande parte, dita o que acontece. Isto é concretizado através de um módulo conhecido como «*menu* do programa», o qual apresenta ao utilizador uma lista das opções que o programa fornece e permite ao utilizador especificar qual delas deverá ser activada. Numa parte mais avançada do livro, programas mais complexos fazem uso de vários *menus,* reflectindo cada um deles a variedade de escolhas sob um cabeçalho principal. De momento, no entanto, ficaremos com o único *menu* necessário a este programa.

Módulo 1.3.7: linhas 4000-4190

```
4000 REM********************************
4010 REM Menu
4020 REM********************************
4030 WHILE x$<>"5"
4040 CLS #2:PRINT #2
4050 PRINT #2,TAB(19);"MENU";TAB(19);"==
==":PRINT #2
4060 PRINT #2," 1) Set timer"
4070 PRINT #2," 2) Delete timer"
4080 PRINT #2," 3) Display timers and wa
it"
4090 PRINT #2," 4) Blank screen and wait
"
4100 PRINT #2," 5) Stop"
4110 sounded=0:x$=""
4120 WHILE x$="" AND sounded=0
4130 x$=INKEY$
4140 IF sounded THEN x$=""
4150 WEND
4160 ON VAL(x$) GOSUB 5000,6000,7000,800
0
4170 PRINT
4180 WEND
4190 RETURN
```

Comentário

Linhas 4030-4170: este ciclo continuará a visualizar o *menu* de cada vez que a execução regressar a este módulo, até que o utilizador introduza o número 5. Logo que isso aconteça, a execução do programa regressará ao módulo de controlo, o qual ainda não introduzimos.

Linhas 4040-4100: estas linhas imprimem a lista de opções programáticas na janela #2, a janela principal que ocupa a maior parte do écran.

Linhas 4110-4150: em condições normais, neste ponto do módulo do *menu,* deveria haver uma vulgar instrução INPUT. A razão da inclusão deste conjunto mais complexo de linhas para chamar o módulo anterior deve-se ao nosso desejo de proceder a uma contínua amostragem do estado dos temporizadores, mesmo com o *menu* no écran. Isto não pode fazer-se se for utilizada a instrução INPUT, já que uma INPUT tranca o programa até ser premido ENTER — nem mesmo EVERY pode interromper uma INPUT.

O método empregue pelas linhas é determinar a variável a utilizar para armazenar a entrada de dados do utilizador (X$) como uma cadeia vazia, continuando a executar o ciclo enquanto aquela permanecer vazia. Se o utilizador premir uma tecla, isso será detectado por INKEY$, na linha 4130, e o carácter associado a essa tecla será armazenada em X$, terminando assim o ciclo.

No entanto, enquanto decorre esta amostragem do teclado, o módulo anterior é constantemente chamado para verificar se é preciso despoletar um alarme. A propósito, a variável SOUNDED tem uma utilidade bastante clara. Normalmente, quando o módulo anterior é chamado, nada acontece, pelo que o *menu* pode continuar sem alterações. Se for despoletado um alarme, no entanto, o *menu* será apagado do écran para ser visualizado o alarme e uma mensagem — neste caso, a SOUNDED será atribuído o valor 1, para indicar ao módulo do *menu* que este tem de ser reimpresso no écran.

Linha 4140: os dados introduzidos pelo utilizador são, se possível, traduzidos para um número, utilizando a função VAL. O comando ON... GOSUB escolhe da lista de destinos aquele que corresponde ao número introduzido pelo utilizador. Se for introduzido um número que esteja fora do âmbito da lista de destinos GOSUB, isto é, se o número for 0 ou maior do que a extensão da lista, o comando ON...GOSUB não é activado e o ciclo visualiza o *menu* novamente.

Ensaio

Antes de ensaiar o programa até este ponto, é mais razoável introduzir o curto módulo de controlo, que vem a seguir.

Módulo 1.3.8: O módulo de controlo

Este é muito mais simples do que em muitos programas, porque a maior parte do trabalho é suportado pelo *menu*. No entanto, e uma vez mais, a temporização do programa é feita por um comando EVERY intrínseco a este módulo. Neste caso, a temporização é feita de cinco em cinco segundos, tal como fizemos notar no comentário ao módulo de ajuste do tempo.

Módulo 1.3.8: linhas 1000-1080

```
1000 REM*********************************
1010 REM Control
1020 REM*********************************
1030 GOSUB 3000
1040 GOSUB 14000
1050 GOSUB 2000
1060 GOSUB 4000
1070 MODE 1
1080 END
```

Ensaio

A execução do programa deverá agora resultar na solicitação de introdução do tempo, seguida do *menu* do programa. Introduza alguns números sem validade para confirmar que não são aceites. A especificação das opções 1 a 4 do programa resultará na paragem deste, com um erro «*Line does not exist*» (linha não existente), enquanto a opção 5 limpará o écran e terminará o programa.

Módulo 1.3.9: Visualização das marcações correntes do temporizador

O objectivo do módulo é imprimir, de forma ordenada, os tempos para que qualquer um dos dezasseis temporizadores estão correntemente regulados. De momento, visto que não introduzimos o módulo que nos permite marcar os valores do temporizador, a visualização será nula, embora a inserção presente do módulo possibilite o ensaio do módulo que regula os temporizadores mal ele seja introduzido.

Módulo 1.3.9: linhas 7000-7170

```
7000 REM*********************************
7010 REM Display Timers and Wait
7020 REM*********************************
7030 k$=""
7040 WHILE k$=""
7050 CLS #2:PRINT #2
7060 PRINT #2,TAB(18);"TIMERS";TAB(18);"
======":PRINT #2
7070 FOR i=0 TO timers-1
7080 h=thour(i):m=tminute(i):s=0:GOSUB 1
5000
7090 PRINT #2,USING " ##) &";i+1;tim$+"
"+message$(i)
7100 NEXT i
7110 sounded=0
7120 WHILE k$="" AND sounded=0
7130 k$=INKEY$
7140 IF sounded=1 THEN k$=""
7150 WEND
7160 WEND
7165 GOSUB 60000
7170 RETURN
```

Comentário

Linhas 7030-7040 e 7160: a visualização continuará até ser premida uma tecla.

Linhas 7070-7100: como salientámos anteriormente, o número de temporizadores correntemente regulados é registado pela variável TIMERS. Este ciclo imprime as marcações do número de temporizadores indicado por TIMERS, utilizando o módulo 1.3.5 para criar uma apresentação em cadeia do tempo registado nos quadros THOUR e TMINUTE, juntamente com PRINT USING, para assegurar um quadro ordenado. O «&» na instrução PRINT USING pode confundir algumas pessoas — o seu objectivo é permitir-nos mencionar um número e depois uma cadeia, na mesma instrução. Se não fosse o «&», seria gerado um erro «*Type mismatch*» (escrita incompatível) quando o programa tentasse interpretar TIM$ como um número da linha 7090.

Linhas 7110-7150: são um estado de espera, tal como no *menu,* durante o qual um comando EVERY fará a amostragem dos temporizadores. Note-se que, se ocorrer um alarme durante a operação deste ciclo e o utilizador premir uma tecla para parar o alarme, a visualização dos temporizadores não será terminada. A linha 7140 assegura que, se a variável SOUNDED indicar que um alarme foi despoletado, uma tecla tenha de ser premida de novo, antes que o programa regresse ao *menu.*

Ensaio

Escreva:

CLEAR:TIMERS = 5:GOTO 7000[ENTER]

Deverá obter a visualização dos tempos para os cinco temporizadores, embora o tempo para cada um seja «00-00-00» e não haja mensagens junto a eles, já que nenhuma foi introduzida. Se premir qualquer tecla, isso resultará numa mensagem de erro «*Unexpected RETURN*».

Módulo 1.3.10: Introdução do tempo para um temporizador absoluto

Dentro de momentos, introduziremos o módulo que permite a marcação dos temporizadores. Mas, antes de o fazermos, são necessários mais dois módulos, para permitir ao utilizador introduzir o tempo-limite para um temporizador de uma ou duas formas diferentes. Ao marcar um temporizador, o utilizador é solicitado a especificar se o tempo a ser introduzido é um tempo absoluto (isto é, se é o minuto e hora a que o alarme deve ser despoletado), ou se é um tempo de contagem decrescente (isto é, se o alarme deve ser despoletado *após* o número especificado de horas e minutos). O módulo corrente trata da introdução de tempos absolutos e é semelhante ao módulo utilizado para introduzir o tempo principal.

Módulo 1.3.10: linhas 11000-11070

```
11000 REM*******************************
11010 REM Input Absolute Timer
11020 REM*******************************
11030 INPUT "Hour (0 to 23):",thour
11040 IF thour<0 OR thour>23 THEN PRINT
```

```
"**** OUT OF RANGE: PLEASE TRY AGAIN ***
*";:FOR i=1 TO 2000:NEXT i:GOTO 11030
11050 INPUT "Minute (0 to 59):",tminute
11060 IF tminute<0 OR tminute>59 THEN PR
INT "**** OUT OF RANGE: PLEASE TRY AGAIN
 ****";:FOR i=1 TO 2000:NEXT i:GOTO 1105
0
11070 RETURN
```

Módulo 1.3.11: Introdução do tempo para um temporizador em contagem decrescente

Esta segunda forma de temporizador é inevitavelmente mais complicada, visto que, em vez de se lhe dizer apenas o tempo em que o alarme deve ser despoletado, o programa tem de calcular esse tempo adicionando qualquer período que o utilizador especifique para o tempo corrente.

Módulo 1.3.11: linhas 10000-10090

```
10000 REM*******************************
10010 REM Input Countdown Timer
10020 REM*******************************
10030 INPUT "Hours to go (0 to 23):",tho
ur
10040 IF thour<0 OR thour>23 THEN PRINT
"**** OUT OF RANGE: PLEASE TRY AGAIN ***
*";:FOR i=1 TO 2000:NEXT i:GOTO 10030
10050 INPUT "Minutes to go (0 to 59):",t
minute
10060 IF tminute<0 OR tminute>59 THEN PR
INT "**** OUT OF RANGE: PLEASE TRY AGAIN
 ****";:FOR i=1 TO 2000:NEXT i:GOTO 1005
0
10070 tminute=tminute+minute:IF tminute>
59 THEN tminute=tminute-60:thour=thour+1
10080 thour=thour+hour:IF thour>23 THEN
thour=thour-24
10090 RETURN
```

Comentário

Linhas 10070-10080: o valor da hora e minuto correntes é calculado pelo módulo de marcação de tempo, o qual será chamado regularmente por uma instrução

EVERY, quando todo o programa tiver sido introduzido. As horas e minutos introduzidos pelo utilizador (THOUR e TMINUTE) podem, portanto, ser acrescentados às variáveis HOUR e MINUTE.

Módulo 1.3.12: Marcação dos temporizadores

Após termo-nos dotado da capacidade de introduzir o tempo para um temporizador, podemos passar ao módulo principal, que permite ao utilizador especificar as marcações para os dezasseis temporizadores. Note-se que este módulo utiliza INPUT para obter os três itens de dados que necessita, pelo que, enquanto o módulo estiver a operar, os temporizadores não serão alvo de amostragem e qualquer temporizador a que caiba dar um alarme não soará, até que este módulo tenha sido completado.

Note também que não pode deixar a máquina à espera indefinidamente, aguardando resposta a uma instrução INPUT. Enquanto isto se passa, o sistema regista o número de vezes que deveria ter efectuado o comando EVERY, o qual será usado, eventualmente, para marcar a temporização do programa. É possível que venha a utilizar todo o espaço reservado para este propósito e, por cada EVERY evitado, será esquecida uma chamada ao módulo de temporização, pelo que o tempo registado pelo programa será incorrecto.

Módulo 1.3.12: linhas 5000-5300

```
5000 REM*********************************
5010 REM Set Timer
5020 REM*********************************
5030 IF timers=16 THEN PRINT "     ******
** NO TIMERS FREE ********":RETURN
5040 INPUT "Countdown (y/n)";down$
5050 IF LOWER$(down$)="y" THEN GOSUB 100
00 ELSE GOSUB 11000
5060 ttime=thour*60+tminute
5070 rtim1=ttime-tim:IF rtim1<=0 THEN rt
im1=rtim1+1440
5080 FOR i=0 TO timers-1
5090 rtim2=ttime(i)-tim:IF rtim2<0 THEN
rtim2=rtim2+1440
5100 IF rtim1>rtim2 THEN NEXT i
5110 IF i<timers AND ttime=ttime(i) THEN
 PRINT "     ****** TIMER ALREADY SET ***
***":FOR j=1 TO 2000:NEXT:RETURN
5120 tnum=i
5130 message$=""
5140 WHILE message$=""
5150 INPUT "Message";message$
5160 IF LEN(message$)>25 THEN PRINT "***
```

```
 MESSAGE TOO LONG (25 CHARS MAX.) ***";:
FOR i=1 TO 2000:NEXT i:message$=""
5170 WEND
5180 PRINT
5190 FOR i=timers-1 TO tnum STEP -1
5200 thour(i+1)=thour(i)
5210 tminute(i+1)=tminute(i)
5220 ttime(i+1)=ttime(i)
5230 message$(i+1)=message$(i)
5240 NEXT i
5250 timers=timers+1
5260 thour(tnum)=thour
5270 tminute(tnum)=tminute
5280 ttime(tnum)=ttime
5290 message$(tnum)=message$
5300 RETURN
```

Comentário

Linha 5030: se o número de temporizadores já marcados for igual a 16, o máximo disponível, então é mostrada uma mensagem de erro e o programa regressa ao *menu*.

Linhas 5040-5050: o utilizador é solicitado a especificar se é necessário um temporizador de contagem decrescente. A linha 5050 traduz a resposta do utilizador para minúscula, se necessário, para que «Y» ou «y» possam ser tratados, chamando depois um dos dois módulos anteriores.

Linhas 5060-5070: o tempo de retorno do módulo de entrada do tempo relevante é convertido num número representativo da quantidade de minutos decorridos desde o princípio do dia, sendo depois subtraído o tempo corrente em minutos. Se o resultado for menor que zero, então o tempo especificado pelo utilizador destina-se ao dia seguinte, sendo adicionados 1440 minutos (um dia, expresso em minutos). Por exemplo, se o tempo corrente for 1830 (18.30h.) e o temporizador tiver de ser marcado para 1800 (18.00h.), então o programa parte do princípio de que o alarme deve soar aos 1800 no dia seguinte.

Linhas 5080-5120: a razão da presença deste ciclo está em que o programa armazena todas as marcações dos temporizadores pela ordem correcta, ou seja, o primeiro temporizador a ser marcado será também o primeiro da linha. Para conseguir isto, o novo item tem de ser comparado, em primeiro lugar, com os tempos de temporizadores existentes, para ver qual o lugar onde deve ser colocado na lista.

Se o(s) primeiro(s) temporizador(es) da lista tiver(em) um tempo menor do que

o tempo corrente (isto é, se estiver marcado para despoletar no dia seguinte), então também lhe serão adicionados 1440 minutos, visando a comparação. Pode encontrar-se, eventualmente, um tempo que seja posterior à nova marcação do temporizador e a instrução IF, na linha 5100, termina o ciclo FOR. Se não houver qualquer marcação presente que seja posterior à nova marcação, então o ciclo prossegue até que I seja igual a TIMERS, pois que, quando um ciclo termina de forma natural, a sua variável é acrescida uma última vez, dado que a execução abandona o ciclo.

O valor de I apontará agora, portanto, *quer* para uma marcação presente do temporizador, posterior à nova marcação, *quer* para o primeiro espaço após o final da lista corrente de temporizadores. É então efectuado mais um ensaio, para verificar se a posição indicada já possui um temporizador que esteja marcado para o tempo acabado de introduzir pelo utilizador. Se assim for, é gerada uma mensagem de erro e o programa regressa ao *menu*. Desde que este erro não seja encontrado, TNUM, o local em que o novo temporizador deve ser inserido, é igualado a I.

Linhas 5130-5170: é solicitada a mensagem que vai ficar associada ao temporizador.

Linhas 5190-5240: visto que sabemos onde o novo temporizador deve ser colocado no âmbito da lista, podemos agora deslocar todos os temporizadores existentes posteriormente a essa posição uma posição para cima, criando assim um intervalo. Cada temporizador necessita de quatro itens de informação para ser registado, e estes são guardados nos quadros THOUR, TMINUTE, TTIME e MESSAGE$.

Linhas 5250-5290: a variável TIMERS é agora aumentada, para indicar que foi acrescentado outro temporizador, e os quatro itens necessários são inseridos nos quatro quadros, na posição indicada por TNUM.

Ensaio

Nesta altura, deve poder ensaiar o módulo executando todo o programa. Quando dispuser do *menu,* especifique a opção 1 e introduza:

Y, 0, 1, e TEST

em resposta às quatro solicitações, após o que o programa deverá regressar ao *menu*. Especifique agora a opção 2 para visualizar os temporizadores, e deverá verificar que ao temporizador 0 foi atribuído um valor algo menos de um minuto adiantado ao tempo corrente (este adiantamento depende da sua rapidez), mais a etiqueta «TEST». Não se incomode a esperar pelo som do alarme, pois ainda não introduzimos o módulo que realiza essa operação.

Módulo 1.3.13: Limpeza do écran

Todo o objectivo deste programa aponta para a sua execução nos bastidores, para que possa actuar como um sistema de temporização. Não há qualquer proble-

ma em deixar ligado o *CPC 464* para realizar aquele propósito, mas deixar ligado o monitor e visualizar o mesmo écran durante muito tempo seguido pode levar a que algumas das partes mais brilhantes do que estiver a ser visualizado se tornem ligeiramente cauterizadas no écran, de modo que apareçam como fantasmas permanentes sobre o que o écran apresentar. (Não fique em pânico por causa disto; não é algo que aconteça apenas porque se esqueceu do *464* e do monitor ligados durante algumas horas — estamos a falar de utilização contínua do mesmo desenho de écran durante um longo período.)

Para ultrapassar este possível problema, o programa tem incluído um módulo de protecção do écran que limpa este até que seja premida uma tecla. Enquanto o écran estiver em branco, os temporizadores serão constantemente amostrados e quaisquer alarmes que despoletem soarão normalmente.

O módulo corrente é utilizado para limpar o écran colocando todas as tintas na cor de fundo. O próximo módulo chama este e cria um estado de espera até que o écran deva ser reactivado.

Módulo 1.3.13: linhas 13000-13080

```
13000 REM*******************************
13010 REM Turn Screen Off
13020 REM*******************************
13030 INK 0,0
13040 INK 1,0
13050 INK 2,0
13060 INK 3,0
13070 BORDER 0
13080 RETURN
```

Módulo 1.3.14: Um estado de espera com um écran em branco

Este módulo não é mais do que um ciclo de espera, do género do que já vimos no *menu* e nos módulos de visualização dos temporizadores, acrescido do facto de que, quando o módulo é chamado, limpa o écran, e quando é premida uma tecla restabelece as cores iniciais.

Módulo 1.3.14: linhas 8000-8130

```
8000 REM********************************
8010 REM Blank Screen and Wait
8020 REM********************************
8030 k$=""
8040 WHILE k$=""
8050 GOSUB 13000
8060 sounded=0
8070 WHILE k$="" AND sounded=0
```

```
8080 k$=INKEY$
8090 IF sounded=1 THEN k$=""
8100 WEND
8110 WEND
8120 GOSUB 12000
8130 RETURN
```

Ensaio

Regule um ou dois temporizadores para períodos curtos e, depois, chame o estado de limpeza do écran, a partir do *menu*. Ainda que não possa ver o estado dos temporizadores, deverá verificar que os alarmes soam de forma normal.

Módulo 1.3.15: Eliminação de um temporizador

Em certos casos, o utilizador pode perfeitamente achar que um temporizador já não é necessário ou está incorrecto. Este módulo dá oportunidade de remover tal temporizador. A técnica de base é importante em programas de tratamento de dados, embora se trate de uma técnica simples, que consiste em provocar o colapso do ficheiro no temporizador a eliminar.

Módulo 1.3.15: linhas 6000-6140

```
6000 REM*********************************
6010 REM Delete Timer
6020 REM*********************************
6030 IF timers=0 THEN PRINT "     *******
* NO TIMERS SET ********":FOR i=1 TO 200
0:NEXT:RETURN
6040 INPUT "Which timer";tnum
6050 IF tnum>timers THEN PRINT "     ****
**** NO SUCH TIMER ********":FOR i=1 TO
2000:NEXT:RETURN
6060 FOR i=tnum TO timers-1
6070 thour(i-1)=thour(i)
6080 tminute(i-1)=tminute(i)
6090 ttime(i-1)=ttime(i)
6100 message$(i-1)=message$(i)
6110 NEXT i
6120 timers=timers-1
6130 IF timers=0 THEN ttime(0)=0
6140 RETURN
```

Comentário

Linhas 6030-6050: são as mensagens de erro, caso o utilizador introduza um número de temporizador que não exista.

Linhas 6060-6110: o processo de colapso, que consiste em deslocar para baixo um espaço tudo o que estiver acima do item a ser removido.

Ensaio

Deverá agora poder marcar um temporizador e chamar depois a opção 2 do *menu* para o apagar, antes de ele se ouvir. Se este ensaio for satisfatório, então o programa estará pronto a ser utilizado.

PROGRAMA 1.4: PROVA DESPORTIVA

Função do programa

O último programa deste capítulo de experiências com o tempo é bastante invulgar, já que pretende transformar o *464* num cronómetro. É claro que a precisão de um microcomputador não deve ser comparada com a dos relógios especializados, mas aquele tem algumas vantagens, tais como poder manter um registo dos tempos produzidos, efectuar cálculos sobre eles e assim por diante. O corrente programa foi concebido para cronometrar uma ou várias provas, visualizar os diversos tempos, com ou sem mensagem adjunta, e, para cada prova, fazer o cálculo do período decorrido desde a prova anterior. Pode também, se lhe for pedido, imprimir a lista de tempos, para poder conservar-se uma *hard copy* (cópia permanente).

```
11:15:04 (00:00:04)
11:15:05 (00:00:01)
11:15:08 (00:00:03)
11:15:15 (00:00:07) BUS
11:15:24 (00:00:09)
11:15:25 (00:00:01)
11:15:27 (00:00:02)
11:15:27 (00:00:00)
11:15:35 (00:00:08) LORRY
11:15:37 (00:00:02)
11:15:38 (00:00:01)
11:15:40 (00:00:02)
11:15:48 (00:00:08) VAN
11:15:50 (00:00:02)
11:15:51 (00:00:01)
11:15:54 (00:00:03)
11:15:57 (00:00:03)
11:16:01 (00:00:05) TANK
```

Fig. 1.4 — *Impressão da saída do programa «Prova Desportiva»*

Novos tópicos apresentados no decurso deste capítulo:

1) Cálculo de períodos de tempo.
2) Saída de informações para uma impressora.

Módulo 1.4.1: Inicialização

É um módulo de inicialização padrão, que define uma pequena janela ausente ao fundo do écran, uma janela de uma linha (#1) no topo do écran e uma segunda (#2) a cobrir a área principal do écran.

Módulo 1.4.1: linhas 2000-2160

```
2000 REM*********************************
2010 REM Initialise
2020 REM*********************************
2030 MODE 1
2040 INK 0,1
2050 INK 1,24
2060 INK 2,26
2070 INK 3,2
2080 BORDER 1
2090 WINDOW 1,40,25,25
2100 PAPER 0:PEN 1:CLS
2110 WINDOW #1,6,35,1,1
2120 PAPER #1,0:PEN #1,2:CLS #1
2130 PRINT #1,"Time now:"
2140 WINDOW #2,1,40,3,23
2150 PAPER #2,3:PEN #2,2:CLS #2:PRINT #2
2160 RETURN
```

Módulo 1.4.2: Marcação do tempo

É o módulo padrão de marcação de tempo encontrado nos dois últimos programas.

Módulo 1.4.2: linhas 3000-3090

```
3000 REM*********************************
3010 REM Set Time
3020 REM*********************************
3030 CLS:PRINT"Clock Setting:":PRINT
3040 INPUT "Hour (1-12):";hour
3050 IF hour<1 OR hour>12 THEN PRINT "**
* Out of range: please try again ***":GO
TO 3040
```

```
3060 INPUT "Minute (0-59):";minute
3070 IF minute<0 OR minute>59 THEN PRINT
 "*** Out of range: please try again ***
":GOTO 3060
3080 second=0
3090 RETURN
```

Módulo 1.4.3: Espera por entrada de dados

INKEY$ é utilizado, uma vez mais, para evitar os problemas relacionados com EVERY, sempre que INPUT é utilizado. Para além de criar o estado de espera, o módulo também é capaz de aceitar uma instrução para enviar futuros dados para uma impressora.

Módulo 1.4.3: linhas 4000-4180

```
4000 REM*********************************
4010 REM Wait
4020 REM*********************************
4030 t$="":message$=""
4040 WHILE t$=""
4050 WHILE t$=""
4060 t$=INKEY$
4070 WEND
4080 WHILE LOWER$(t$)="p"
4090 printer=1-printer
4100 t$=""
4110 WEND
4120 WHILE t$=CHR$(13)
4130 INPUT "Message (<=18 chars):",messa
ge$
4140 IF LEN(message$)<=18 THEN t$="*"
4150 WEND
4160 CLS
4170 WEND
4180 RETURN
```

Comentário

Linha 4030: esta cadeia será usada para armazenar qualquer tecla premida pelo utilizador.

Linhas 4040-4170: são o ciclo principal, o qual continuará até que seja atribuído um valor a T$, quer pelo programa, quer pela pressão de uma tecla.

Linhas 4050-4070: são o verdadeiro estado de espera, que continuará até que seja premida qualquer tecla.

Linhas 4080-4110: se a tecla premida for a da letra P, então o valor da variável PRINTER será encravado entre 0 e 1. Vale a pena prestar atenção a esta simples linha, pois que ela representa a forma clássica de movimentar uma variável entre dois valores. Se possui uma variável X e pretende movimentá-la para trás e para a frente, entre os valores V1 e V2 (sendo V1 o mais baixo), o formato será:

$X = V1 + V2 - X$

mas, como o nosso valor mais baixo é zero, é simplificado para:

$X = V2 - X$

Linhas 4120-4150: se a tecla premida for ENTER, então o utilizador será solicitado, na linha de comando ao fundo do écran, para introduzir uma pequena mensagem a acompanhar o tempo da prova, voltando depois a linha de comando a ficar em branco.

Ensaio

Assegure-se de que o programa foi inicializado, escrevendo depois:

GOTO 4000[ENTER]

e o *464* deverá passar ao estado de espera. Se premir a tecla P, nada de visível deverá acontecer. Se premir [ENTER], dever-lhe-á ser solicitada uma mensagem, terminando a espera. Qualquer outra tecla apenas porá fim ao módulo.

Módulo 1.4.4: Ajuste do tempo

Este módulo é semelhante aos dos dois programas anteriores, com a excepção de que, para além de actualizar constantemente o tempo principal, actualiza também uma segunda forma do tempo que representa o período desde a última vez que uma tecla foi premida. Esta segunda forma do tempo é armazenada sob a aparência de variáveis começadas pela letra P.

Módulo 1.4.4: linhas 6000-6280

```
6000 REM*******************************
6010 REM Adjust Time
6020 REM*******************************
6030 second=second+1
```

```
6040 WHILE second>59
6050 second=0:minute=minute+1
6060 WHILE minute>59
6070 minute=0:hour=hour+1
6080 WHILE hour>12
6090 hour=1
6100 WEND
6110 WEND
6120 WEND
6130 psecond=psecond+1
6140 WHILE psecond>59
6150 psecond=0:pminute=pminute+1
6160 WHILE pminute>59
6170 pminute=0:phour=phour+1
6180 WHILE phour>12
6190 phour=1
6200 WEND
6210 WEND
6220 WEND
6230 LOCATE#1,11,1
6240 h=hour:m=minute:s=second:GOSUB 7000
6250 PRINT#1,tim$;
6260 h=phour:m=pminute:s=psecond:GOSUB 7
000
6270 PRINT#1," (";tim$;")"
6280 RETURN
```

Ensaio

O módulo não pode ser efectivamente ensaiado até que o próximo seja introduzido, o que permitirá a impressão do tempo.

Módulo 1.4.5: Visualização dos resultados

Este módulo, tal como nos programas anteriores, constrói uma cadeia a partir do tempo corrente.

Módulo 1.4.5: linhas 7000-7050

```
7000 REM*********************************
7010 REM Create Time String
7020 REM*********************************
7030 tim$=STR$(1000000+h*10000+m*100+s)
```

```
7040 tim$=MID$(tim$,3,2)+":"+MID$(tim$,5
,2)+":"+MID$(tim$,7,2)
7050 RETURN
```

Ensaio

Escreva:

CLEAR:CLS:GOTO 6000 [ENTER]

e deverá ver:

00:00:01 (00:00:01)

impresso no topo do écran. Se introduzir GOTO 6000 repetidamente, verificará que o tempo aumenta um segundo de cada vez.

Módulo 1.4.6: Impressão dos resultados

Após ter introduzido os resultados que colocam o programa em espera e lhe permitem actualizar o tempo, precisamos agora conseguir imprimir o tempo calculado quando uma tecla for premida.

Módulo 1.4.6: linhas 5000-5100

```
5000 REM*********************************
5010 REM Print Values
5020 REM*********************************
5030 h=hour:m=minute:s=second:GOSUB 7000
5040 p$=" "+tim$
5050 h=phour:m=pminute:s=psecond:GOSUB 7
000
5060 p$=p$+" ("+tim$+") "+message$
5070 PRINT#2,p$
5080 IF printer=1 THEN PRINT#8,p$
5090 phour=0:pminute=0:psecond=0
5100 RETURN
```

Comentário

Linhas 5030-5070: é construída uma cadeia consistindo no tempo corrente (HOUR,MINUTE,SECOND), seguida do tempo-período (PHOUR, PMINUTE, PSECOND) e qualquer mensagem que o utilizador tenha introduzido. Esta é impressa na janela principal do écran (#).

Linha 5080: o *464,* contrariamente a muitos outros micros, mantém aberto um canal permanente de comunicação com qualquer impressora que lhe esteja ligada.

Na maioria dos outros micros seria necessário «abrir um ficheiro para a impressora», requerendo várias linhas em BASIC. No *464,* tudo o que é preciso é enviar o que quer que queiramos ver impresso para o número de «cadeia» *(string)* 8, que está sempre ligado directamente à saída da impressora.

Linha 5090: os valores do tempo-período passam a 0, visto que apenas se pretende que eles representem o tempo desde a última tecla premida.

Módulo 1.4.7: O módulo de controlo

O toque final é o módulo de controlo, que liga o programa como um todo funcional.

Módulo 1.4.7: linhas 1000-1110

```
1000 REM*********************************
1010 REM Control
1020 REM*********************************
1030 ON BREAK GOSUB 1110
1040 GOSUB 3000
1050 EVERY 50 GOSUB 6000
1060 GOSUB 2000
1070 WHILE 1
1080 GOSUB 4000
1090 GOSUB 5000
1100 WEND
1110 CLEAR:MODE 1:END
```

Comentário

Linha 1070: um pequeno retoque em que pode encontrar utilidade — o «1» depois de WHILE permite que este ciclo continue indefinidamente. Qualquer valor positivo realizaria a mesma função.

Conclusão

Há várias lições a retirar dos programas que compõem este capítulo, a menor das quais não será que as utilizações do seu *464* apenas estão limitadas pela imaginação do utilizador. Mas, talvez a lição mais importante a retirar, fazendo uma retrospectiva dos programas, seja a forma como se torna fácil a concepção de programas, quando tudo está contido em módulos claros que podem ser transferidos de um programa para o seguinte. Com o seu comando MERGE de fácil utilização, o *464* implora que o usem desta maneira. Uma vez construída uma biblioteca suficiente de rotinas úteis, descobrirá que grande parte da sua programação se identifica com a junção das peças de um quebra-cabeças, excepto que, pelo menos em certos casos, o quebra-cabeças completo terá mais utilidade.

CAPÍTULO 2

Pintura por números

Neste capítulo, passamos dos jogos com o tempo para algo que o *464* faz particularmente bem: a visualização de informação sob formas que são mais imediatamente compreensíveis do que listas de factos e números. O capítulo é formado por três programas, que criarão gráficos de um ou outro tipo, tanto em baixa como em alta resolução.

À medida que se avança neste capítulo, e nos que se seguem, verifica-se que as explicações se tornam mais breves, excepto nas situações em que se apresentam novas ideias ou módulos complexos. Isto é deliberado e é uma tentativa de estabelecer um equilíbrio entre a informação necessária para compreender os programas que se estiver a introduzir e os próprios programas, que são a razão que o levou a comprar o livro. Se *realmente* ficar encalhado em qualquer sítio, descobrirá, em geral, que pode voltar a um programa anterior onde uma técnica semelhante tenha sido utilizada, relendo o comentário a ela referente antes de continuar. É por esta razão que lhe foi recomendado, na «Introdução», que avançasse ao longo do livro, em vez de saltar até alguns dos programas mais ambiciosos em partes posteriores do livro.

Os programas incluídos neste capítulo são:

GRÁFICO: cria um poderoso e muito eficiente gráfico linear em alta resolução.

GRÁFICO DE SECÇÕES: pega num conjunto de dados limitado e apresenta-o sob a forma de um círculo dividido em segmentos multicolores.

GRÁFICO 3D: permite-lhe criar um espantoso gráfico de barras tridimensional.

PROGRAMA 2.1: GRÁFICO

Função do programa

Um dos problemas programáticos de tratamento de dados que mais tempo e memória consome é a forma como esses dados devem ser introduzidos. Isso, frequentemente, requer secções programáticas bastante longas e complexas, devotadas à introdução, alteração e remoção de dados.

Nos dois primeiros programas deste capítulo, examinamos uma forma de con-

tornar aquela limitação, através da criação de um programa fácil de utilizar, mas que não precisa de dedicar memória à colocação de solicitações no écran, ao armazenamento de dados em quadros ou à colocação de dados para armazenamento em fita. Os programas interactivos, ou seja, programas que pedem informações ao utilizador durante a execução, são certamente agradáveis de utilizar, mas podem ser um luxo sempre que a memória é apertada. Neste programa, que desenha um gráfico em alta resolução, fazemos uso de instruções DATA para criar um programa que é a própria simplicidade na utilização e também muito mais simples de escrever do que um programa interactivo, o qual alcançaria o mesmo resultado.

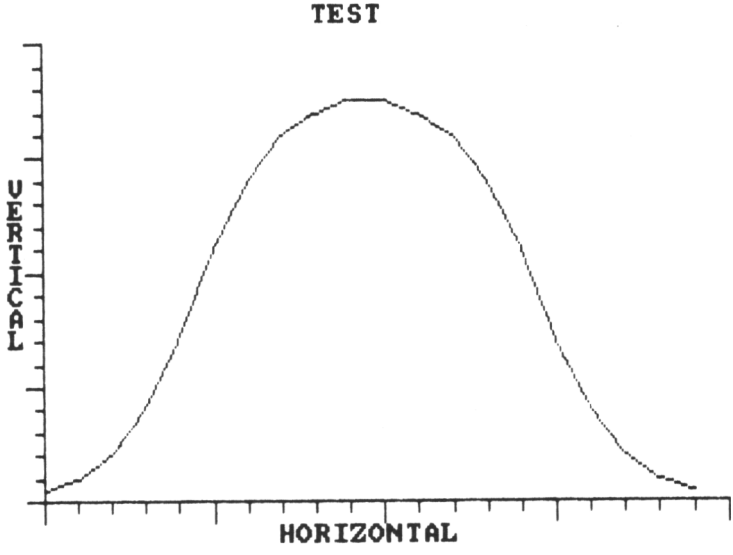


Fig. 2.1 — *Visualização do écran no «Gráfico»*

Técnicas apresentadas no decurso do programa incluem:

1) Uso flexível de instruções DATA.

Módulos 2.1.1 e 2.1.2: Os dados para o gráfico

Estes dois módulos são a chave da simplicidade do programa. Nestas poucas linhas está contida toda a informação que será necessária para desenhar um gráfico satisfatório, todo ele claramente etiquetado para que o utilizador não possa ter qualquer espécie de dificuldade em organizar a visualização. Não fique preocupado se, nesta altura, encontrar dificuldades em descobrir a relevância de todos os números — a sua utilização tornar-se-á clara logo que tenha introduzido e executado o programa uma ou duas vezes e experimentado fazer algumas alterações.

Módulos 2.1.1 e 2.1.2: linhas 3000-4050

```
3000 REM ********************************
3010 REM Data 1
3020 REM ********************************
3030 DATA GRAPH TITLE,TEST
3040 DATA H/AXIS NAME,HORIZONTAL
3050 DATA V/AXIS NAME,VERTICAL
3060 DATA UNITS ON H/AXIS,20
3070 DATA UNITS ON V/AXIS,20
3080 DATA AMOUNT PER V/UNIT,10
4000 REM ********************************
4010 REM Data 2
4020 REM ********************************
4030 DATA 5,10,20,40,70,110,140,160,170,
175
4040 DATA 175,170,160,140,110,70,40,20,1
0,5
4050 DATA END
```

Comentário

Linhas 3030-3050: nome que o utilizador deseja dar ao gráfico, no seu conjunto, e as etiquetas dos eixos vertical e horizontal. Note que, em cada caso, a frase antes da vírgula na instrução de dados está lá apenas para conveniência do utilizador e será ignorada pelo programa — a vírgula é essencial para separar a primeira frase da informação importante que se lhe segue.

Linhas 3060-3070: os dois eixos do gráfico serão divididos em unidades para facilidade de leitura. Os eixos terão sempre o mesmo comprimento, mas o utilizador pode especificar em quantas unidades eles irão dividir-se.

Linha 3080: se o gráfico se destinasse, por exemplo, a registar o número de toneladas de trigo produzidas por um país ao longo de vários anos, o utilizador poderia desejar que cada unidade do eixo vertical registasse uma unidade 1000 toneladas. Em vez de obrigar o utilizador a dividir o número real em unidades de 1000, antes de o introduzir, o número desta linha permite que as unidades sejam especificadas, de modo que cada número possa ser introduzido por inteiro.

Linhas 4030-4040: constituem os dados em que o gráfico irá basear-se. No caso da presente organização do programa, estes números produzirão uma curva suave, em forma de sino. Note que apenas há contemplação para um número por cada unidade do eixo horizontal, começando na posição um, embora nem todo o eixo horizontal tenha de ser usado. Podem ser incluídos em cada linha separada tantos itens de dados quantos os desejados.

Linha 4050: pode utilizar tantas instruções de dados quantas a memória permitir, mas a informação terá de terminar com uma instrução de dados que contenha a palavra END (fim). É este o sinal que indica ao programa que atingiu o final dos dados a serem utilizados para o gráfico — ainda que se sigam mais instruções de dados.

Módulo 2.1.3: Desenho da grelha para o gráfico

Este módulo desenha a armação sobre a qual irá assentar o eventual gráfico, juntamente com as unidades dos eixos e as várias etiquetas especificadas.

Módulo 2.1.3: linhas 1000-1190

```
1000 REM ******************************
1010 REM Draw Grid
1020 REM ******************************
1030 MODE 1
1040 MOVE 32,368
1050 DRAW 32,32,2
1060 DRAW 639,32
1070 RESTORE 3000
1080 READ t$,t$:LOCATE 20-LEN(t$)/2,1:PR
INT t$
1090 READ t$,t$:LOCATE 21-LEN(t$)/2,25:P
RINT t$
1100 READ t$,t$:FOR i=1 TO LEN(t$):LOCAT
E 1,12-LEN(t$)/2+i:PRINT MID$(t$,i,1):NE
XT i
1110 READ t$,nv:lv=338/nv
1120 FOR i=0 TO nv
1130 MOVE 24+8*(i/5=INT(i/5)),32+i*lv:DR
AW 32,32+i*lv
1140 NEXT i
1150 READ t$,nh:lh=606/nh
1160 FOR i=0 TO nh
1170 MOVE 32+i*lh,24+8*(i/5=INT(i/5)):DR
AW 32+i*lh,32
1180 NEXT i
1190 READ t$,unit
```

Comentário

Linhas 1040-1060: são desenhados os dois eixos, partindo uma linha das proximidades do canto superior esquerdo do écran até perto do canto inferior esquerdo, de onde continua em ângulo recto ao longo do fundo do écran até perto do canto inferior direito.

Linha 1070: o «apontador de dados» do *464* é regulado para apontar para o primeiro item de dados a seguir ao início do módulo, em 3000. Este RESTORE evitará que seja gerado um erro *«DATA exhausted»* (dados esgotados), se o programa for iniciado com GOTO. A utilização de RUN, seja como for, regula o apontador para o primeiro item de dados.

Linhas 1080-1100: as etiquetas para o gráfico, no seu todo, o eixo horizontal e o eixo vertical são lidos a partir das instruções de dados e impressos no écran. No caso da etiqueta para o eixo vertical, é utilizado um ciclo para imprimir a etiqueta, carácter por carácter, ao longo da margem esquerda do écran, no sentido descendente. Note que, em cada caso, existem duas instruções READ. A primeira recolhe a frase *antes* da vírgula, na instrução de dados. Aquela é, então, imediatamente descartada pela leitura (READ) de outra cadeia na mesma variável, T$.

Linha 1110: o número de divisões a colocar no eixo vertical (NV) é lido a partir da instrução de dados seguinte. A extensão do eixo vertical (338 *pixels*) é, então, dividida por este número, para se chegar ao comprimento de cada divisão em *pixels* (LV).

Linhas 1120-1140: este ciclo desenha pequenas marcas no eixo vertical, para registar as divisões especificadas pelo utilizador. Uma coisa a ter aqui em especial atenção é a expressão 8*(I/5 = INT(I/5)), a qual tem como efeito tornar a marca para cada quinta unidade maior do que as outras. Para se compreender a expressão, é preciso saber alguma coisa acerca da forma como as «condições lógicas» são tratadas pelo *464*.

Quando o intérprete de BASIC do *464, o* maciço programa em código-máquina que executa o BASIC para si, encontra uma condição a seguir a um IF, algo como «A > B», «A = B» ou «A < = B», precisa de determinar se essa condição é verdadeira ou falsa, antes de decidir se deve continuar com a acção especificada pela instrução IF. Assim:

```
IF A> B THEN GOSUB 1000
```

será activado se «A > B» for verdadeiro (exemplo: A = 10 e B = 9) e ignorado se for falso (exemplo: A = 9 e B = 10). Para que seja tomada uma decisão, a condição é avaliada de acordo com os valores correntes de A e B (ou se tiverem sido especificadas variáveis) e é dado à condição um valor que é − 1, se a condição for verdadeira, e 0, se a condição for falsa. É o *valor* que interessa, e não tanto a condição, facto que você pode demonstrar a si mesmo realizando o curto ensaio que se segue. Introduza, em modo directo:

```
A = 5[ENTER]
IF A THEN PRINT "TRUE"[ENTER]
```

O resultado será que «TRUE» é impresso no écran, visto que o valor a seguir à instrução IF não é 0 (qualquer valor que não seja zero, positivo ou negativo, produzirá o mesmo efeito). Experimente agora:

A = 0[ENTER]
IF A THEN PRINT "TRUE"[ENTER]

Desta vez, nada será impresso no écran. De momento, no entanto, não estamos tanto interessados no modo como IF funciona, mas na forma como as condições são avaliadas. Por isso, experimente o seguinte:

A = 1[ENTER]
B = 1[ENTER]
PRINT A = B[ENTER]

O que deverá ver é « – 1», o valor de uma condição verdadeira. Experimente agora:

A = 1[ENTER]
B = 2[ENTER]
PRINT A = B[ENTER]

O resultado será agora «0», já que a condição não é verdadeira. A princípio, isto pode parecer interessante, mas moderadamente irrelevante. De facto, esta capacidade de extrair um valor de uma condição lógica é bastante valiosa em programação, tal como a linha 1130 ilustra.

O que a linha 1130 faz é desenhar uma série de linhas em ângulo recto com o eixo vertical, para marcar as divisões especificadas pelo utilizador. O comprimento destas linhas é, normalmente, de oito *pixels*. No entanto, sempre que o valor da variável I do ciclo for exactamente divisível por cinco (isto é, em cada quinta marca), a condição (I/5 = INT(I/5)) será verdadeira e assumirá o valor – 1, em vez de 0. Por outras palavras, a inclusão de « – 8*(I/5 = INT(I/5))» na linha que especifica o comprimento da marca a ser desenhada (DRAW) permite a duplicação do comprimento de cada quinta marca, sem a utilização de uma complexa instrução IF... THEN... ELSE.

Note que, para adicionar oito ao comprimento da marca, é preciso *retirar* oito vezes o valor da condição, posto que o seu valor, quando verdadeiro, é um *negativo* — a retirada de um número negativo é equivalente à adição de um número positivo.

Linhas 1150-1180: é executado exactamente o mesmo processo para o eixo horizontal.

Linha 1190: o número de unidades representadas por cada divisão, no eixo vertical, é lido a partir da instrução de dados.

Ensaio

Tudo o que é preciso para ensaiar esta parte do programa é executá-lo até ao ponto a que chegámos. Deverá ver a grelha do gráfico desenhada no écran, com «TEST» em cima, «VERTICAL» ao longo da margem esquerda e «HORIZONTAL» ao longo da margem inferior. Ambos os eixos devem apresentar-se claramente divididos em vinte unidades.

Módulo 2.1.4: Desenho do gráfico

Dispondo já da estrutura, tudo o que resta fazer é desenhar o próprio gráfico, utilizando a informação especificada no módulo DATA2.

Módulo 2.1.4: linhas 2000-2160

```
2000 REM ******************************
2010 REM Draw Graph
2020 REM ******************************
2030 ON ERROR GOTO 2140
2040 RESTORE 4000
2050 READ t
2060 PLOT 32,32+t/unit*lv,1
2070 column=1
2080 READ t$
2090 IF t$="END" THEN GOTO 2130
2100 DRAW 32+lh*column,32+VAL(t$)/unit*l
v
2110 column=column+1
2120 GOTO 2080
2130 IF INKEY$="" THEN GOTO 2130
2140 CLS
2150 LIST 3000-
2160 END
```

Comentário

Linhas 2030 e 2130-2160: se for gerado um erro durante a sequência de desenho principal, ou uma tecla premida uma vez desenhado o gráfico, o écran limpará e listará os dados em que se baseia o gráfico. Isto torna extremamente fácil o exame do gráfico e a alteração de pormenores a ele referentes.

Linha 2050: é recolhido o primeiro item de dados.

Linha 2060: o gráfico é iniciado no eixo vertical, num ponto representando a dimensão do primeiro item de dados, tal como se encontra expresso nas unidades especificadas pelo utilizador.

Linha 2070: COLUMN (coluna) será utilizado para registar a quantidade de itens de dados lidos e, portanto, até onde o gráfico se deslocou ao longo do eixo horizontal.

Linhas 2080-2090: itens de dados subsequentes são lidos como cadeias. Isto permite-nos verificar se o item lido é realmente a palavra «END» e terminar o programa, se assim for.

Linhas 2100-2120: como o cursor dos gráficos já se encontra no final da parte do gráfico já desenhada, tudo o que é preciso fazer é desenhar (DRAW) até ao ponto apropriado seguinte, aumentando COLUMN por cada item de dados lido. As coordenadas «X», ou horizontais, são calculadas pela multiplicação de COLUMN (ou o número de unidades que o gráfico avançou ao longo do eixo horizontal) pelo comprimento em *pixels* das unidades horizontais (LH) — a constante 32 é a distância para o interior a que o início do eixo horizontal está do limite esquerdo do écran. As coordenadas «Y», ou verticais, são divididas primeiro por UNIT.

Assim, se o item de dados fosse 1,000,000 e o utilizador tivesse especificado que o eixo vertical devia ser dividido em unidades de 100,000 (UNIT = 100,000), então o resultado seria 1,000,000/100,000 ou 10 unidades. Tendo chegado ao número de unidades, este é então multiplicado pelo comprimento das unidades verticais (LV) em *pixels*.

Ensaio

Execute o programa terminado e deverá ver desenhar-se uma curva suave, em forma de sino. Quando o desenho terminar, prima uma tecla qualquer e deverá ver os módulos DATA listados no écran, para que você possa alterá-los à vontade. Se este ensaio for satisfatório, o programa deverá estar pronto a usar.

PROGRAMA 2.2: GRÁFICO DE SECÇÕES

Função do programa

Uma forma muito útil de apresentar pequenas quantidades de dados é a técnica do «gráfico de secções», em que um círculo representando um valor total é dividido em segmentos que representam as diferentes partes que compõem o todo. No programa que se segue, iremos desenhar com base no que já aprendemos acerca da matemática dos círculos e do uso flexível das instruções de dados, no último programa. Se não estiver esclarecido em qualquer dos aspectos, por favor volte atrás e reveja esses assuntos, pois não iremos demorar-nos a explicar novamente os trâmites das técnicas.

Não é necessário apagar as linhas introduzidas durante os procedimentos de ensaio, já que essas linhas terão de ser introduzidas como parte do último módulo, o módulo de controlo.

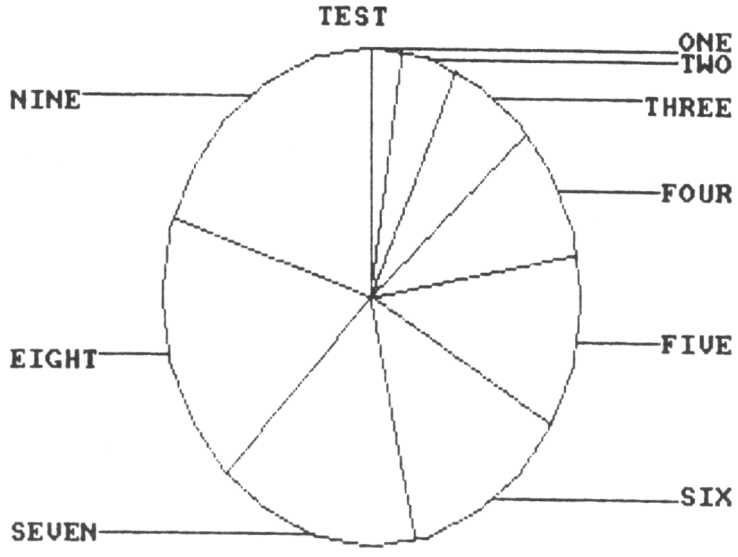


Fig. 2.2 — *Visualização do «Gráfico de Secções»*

Módulo 2.2.1: Os dados para o gráfico

Tal como no gráfico de alta resolução anterior, os valores em que se baseará o presente gráfico estão contidos em instruções de dados que se auto-explicam. No entanto, note que, no programa tal como está listado, os dois quadros que serão utilizados para conter o nome de cada item e o seu valor não são dimensionados, pelo que você está limitado a dez itens. Honestamente, um gráfico de secções com mais de dez itens tem pouco valor, porque se torna demasiado cheio para aceitar realmente a informação. Ainda assim, pode incluir, se desejar, uma instrução dimensionadora no início do programa, para aumentar o número de itens a tratar.

Módulo 2.2.1: linhas 4000-4080

```
4000 REM********************************
4010 REM data for chart
4020 REM********************************
4030 DATA TITLE,TEST
4040 DATA NUMBER OF ITEMS,9
4050 DATA NAMES,ONE,TWO,THREE,FOUR,FIVE,
SIX,SEVEN,EIGHT,NINE,TEN
4060 DATA
```

```
4070 DATA QUANTITIES,1,2,3,4,5,6,7,8,9,1
0
4080 DATA
```

Módulo 2.2.2: Processamento dos dados para o gráfico

A informação contida no módulo DATA é lida nas variáveis NAME$ e ITEMS, e nos quadros NAME$ e A.

Módulo 2.2.2: linhas 5000-5110

```
5000 REM*********************************
5010 REM Process Data
5020 REM*********************************
5030 RESTORE 4000
5040 READ t$,name$
5050 READ t$,items
5060 READ t$:FOR i=0 TO items-1:READ nam
e$(i):NEXT i
5070 RESTORE 4070
5080 sum=0:READ t$:FOR i=0 TO items-1:RE
AD t:sum=sum+t:NEXT i
5090 RESTORE 4070
5100 READ t$:FOR i=0 TO items-2:READ t:a
(i+1)=(t/sum)*360+a(i):NEXT i
5110 RETURN
```

Comentário

Linhas 5080-5100: o valor dos itens a serem incluídos no gráfico são primeiro adicionados entre si, para descobrir o total que o círculo representará. O apontador DATA será então restaurado (RESTORE) no início dos valores de quantidade e cada quantidade será traduzida num segundo valor, que, quando dividido, deverá dar o mesmo resultado que a quantidade original dividida pelo total. Por exemplo, se o total fosse 100 e a quantidade para um item 25, isto seria traduzido em 90, ou 25% de 360. Estes novos valores serão utilizados mais tarde, para determinar qual a fatia do gráfico a atribuir a cada item.

Ensaio

Introduza as linhas seguintes a partir do que será, eventualmente, o módulo de controlo, e depois execute o programa:

```
1040 GOSUB 5000
1100 END
```

Se tudo estiver bem, então nada de visível deve acontecer — só se houver um erro de qualquer espécie é que verá alguma coisa. Se desejar, no entanto, pode imprimir o conteúdo das variáveis e quadros designados no módulo, para poder ficar tranquilo.

Módulo 2.2.3: Organização do écran

Este módulo elabora o modo gráfico e cores associadas, para além de desenhar um círculo de 80*80 no centro do écran, juntamente com o nome do gráfico.

Módulo 2.2.3: linhas 2000-2090

```
2000 REM*********************************
2010 REM draw framework
2020 REM*********************************
2030 MODE 1:ORIGIN 320,184:BORDER 0
2040 INK 0,0:INK 1,2:INK 2,6:INK 3,18
2050 LOCATE 20-LEN(name$)/2,1:PEN 2:PRIN
T name$
2060 DEG
2070 PLOT 0,184,3
2080 FOR a=0 TO 360 STEP 15:DRAW 184*SIN
(a),184*COS(a):NEXT a
2090 RETURN
```

Comentário

Linha 2080: se ler a explicação sobre a matemática do desenho de um círculo, no comentário do primeiro capítulo do livro (cap. 1), perceberá que esta linha apenas desenha um círculo. Em vez de traçar cada ponto da circunferência individualmente, o que acontece com esta linha é que são calculados vários pontos afastados entre si 15 graus, sendo depois desenhadas linhas entre eles.

Ensaio

Acrescente as linhas seguintes e depois execute o programa:

```
1060 GOSUB 2000
1080 IF INKEY$ = "" THEN GOTO 1080
```

O resultado deverá ser nada mais espectacular do que o título do gráfico e um círculo amarelo. Prima qualquer tecla, excepto ESC, para voltar ao écran normal.

Módulo 2.2.4: Introdução dos pormenores

Este módulo desenha os segmentos em que o gráfico irá dividir-se, pinta-os e junta-lhes as etiquetas especificadas no módulo DATA. Para perceber o que se passa, precisará de lembrar-se da matemática simples de um círculo, tal como foi explicado no programa «Anarelógio». Se se esqueceu, deve voltar a esse programa e ler de novo o comentário.

Módulo 2.2.4: linhas 3000-3170

```
3000 REM*********************************
3010 REM Insert Segments
3020 REM*********************************
3030 FOR i=0 TO items-1
3040 MOVE 0,0:DRAW 184*SIN(a(i)),184*COS
(a(i))
3050 NEXT i
3055 PEN 1
3060 FOR i=0 TO items-1
3070 ta=((a(i)+a(i+1+items*(i+1=items)))
)/2
3080 IF a(i+1+items*(i+1=items))<a(i) TH
EN ta=ta+180
3090 MOVE 180*SIN(ta),180*COS(ta):c=i MO
D 3+1:IF i=items-1 AND c=1 THEN c=2
3100 GOSUB 6000
3110 tx=184*SIN(ta)
3120 ty=184*COS(ta)
3130 dx=320*SGN(tx)
3140 MOVE tx,ty:DRAW dx,ty,3
3150 LOCATE 1+(LEN(name$(i))-40)*(dx=320
),14-INT((ty+9)/16):PRINT name$(i);
3160 NEXT i
3170 RETURN
```

Comentário

Linhas 3030-3050: é desenhada uma série de linhas do centro do círculo para a circunferência, dividindo o círculo nos segmentos que constituem o gráfico. Os valores utilizados são os calculados no módulo 2.2.2.

Linhas 3070-3100: são calculados mais ângulos, mas, desta vez, a sua posição é intermédia dentro de cada um dos segmentos acabados de desenhar. A expressão lógica nas linhas 3070 e 3080 (ITEMS* (I + 1 = ITEMS) assegura que, quando o último

segmento tiver sido alcançado, o ponto de partida do primeiro segmento seja utilizado como o segundo ângulo para o cálculo da posição intermédia. Ao efectuar-se este último cálculo, a resposta aparecerá errada: em vez de ser adicionada um ângulo a um ângulo maior e dividido por 2 para encontrar o ponto intermédio, o ângulo do último segmento será adicionado a 0 (o início do primeiro segmento), produzindo assim um ângulo intermédio do início do primeiro e último segmentos. Isto é rectificado adicionando 180 graus à última resposta.

A linha 3090 calcula, então, uma posição baseada neste ângulo, que se situa junto ao interior da circunferência do círculo. É então feita uma chamada ao módulo seguinte, que é bastante semelhante ao módulo de «preenchimento» utilizado no «Anarelógio», para colorir o segmento em forma de cunha sobre o qual recai o ponto corrente. A linha 3090 produz também um ciclo regular das três cores de primeiro plano especificadas no módulo de inicialização, com excepção do último segmento, que não pode ter a mesma cor do primeiro segmento. A razão disto está em que se assegura que o segmento final não virá a ter a mesma cor do primeiro — uma vez que se encontram juntos, isto tornaria difícil a leitura do gráfico.

Linhas 3110-3120: estas duas linhas calculam um ponto na circunferência do círculo, num ângulo intermédio dos pontos inicial e final do corrente segmento.

Linha 3130: na montagem da estrutura do gráfico, já deslocámos a origem dos comandos dos gráficos para o centro do écran. Esta linha calcula uma posição na horizontal distando 320 *pixels* do centro. A variável TX contém já a coordenada X de um ponto da circunferência que tanto pode ser negativo (para a esquerda do centro) ou positivo (para a direira do centro). Multiplicando 320 por SGN(TX) assegura-se que, se o centro do segmento ficar para a esquerda do centro do écran, o mesmo acontecerá com DX e vice-versa.

Linhas 3140-3150: é desenhada uma linha a partir da circunferência do círculo até ao limite do écran, quer para a direita, quer para a esquerda, tal como definido em DX. No final da linha, ou antes, por cima dela, é impressa a etiqueta do segmento para onde aponta a linha. A posição de impressão das etiquetas da margem direita é deslocada para a esquerda, para que não sejam despejadas para fora do écran, utilizando uma condição lógica para determinar a coordenada X. A linha parece mais complexa do que é, na verdade, devido à necessidade de traduzir a posição previamente calculada em *pixels* numa posição para caracteres.

Ensaio

Acrescente as linhas seguintes e execute o programa:

```
1070 GOSUB 3000
6000 RETURN
```

Deverá obter uma visualização como a do início da secção deste programa, embora os segmentos individuais não estejam coloridos.

Módulo 2.2.5: Preenchimento dos segmentos

Este módulo é quase idêntico ao que está contido no programa «Anarelógio». A única diferença é que o processo de preenchimento decorre até encontrar qualquer cor que não seja a cor de fundo.

Módulo 2.2.5: linhas 6000-6300

```
6000 REM*********************************
6010 REM Fill
6020 REM*********************************
6030 IF TESTR(0,0)<>0 THEN RETURN
6040 s=2
6050 MOVE s*INT(XPOS/s),2*INT(YPOS/2)
6060 :
6070 :
6080 REM search up/right **************
6090 IF TESTR(0,s)=0 THEN GOTO 6090
6100 IF TESTR(s,-s)=0 THEN GOTO 6090
6110 :
6120 :
6130 REM search up/left ***************
6140 MOVER -s,0
6150 IF TESTR(0,s)=0 THEN GOTO 6090
6160 IF TESTR(-s,-s)=0 THEN GOTO 6150
6170 :
6180 :
6190 REM ink in lines across **********
6200 x1=XPOS
6210 MOVER s,0
6220 PLOTR 0,0,c
6230 IF TESTR(s,0)=0 THEN GOTO 6220
6240 x2=XPOS-s
6250 MOVE x1,YPOS-s
6260 IF TESTR(s,0)=0 THEN GOTO 6290
6270 IF XPOS=x2 THEN RETURN
6280 GOTO 6260
6290 IF TESTR(-s,0)<>0 THEN GOTO 6200
6300 GOTO 6290
```

Ensaio

O mesmo que para o módulo anterior, excepto que os segmentos devem agora ficar coloridos.

Módulo 2.2.6: O módulo de controlo

Muitas das linhas a ele destinadas foram já introduzidas, mas certifique-se de que possui todas as linhas listadas em baixo, sem o que lhe faltarão um ou dois refinamentos.

Módulo 2.2.6: linhas 1000-1120

```
1000 REM*********************************
1010 REM Control
1020 REM*********************************
1030 ON ERROR GOTO 1110
1040 GOSUB 5000
1050 ON BREAK GOSUB 1090
1060 GOSUB 2000
1070 GOSUB 3000
1080 IF INKEY$="" THEN GOTO 1080
1090 CLEAR:CLS:LIST 4000-4999
1100 END
1110 PRINT"  *** PROBABLE INVALID DATA F
ORMAT ***"
1120 FOR i=1 TO 3000:NEXT i:GOTO 1090
```

Comentário

Linhas 1030 e 1110-1120: uma breve mensagem de erro para indicar que existe, provavelmente, um erro no plano do módulo DATA — este não é infalível como um ensaio, já que certos enganos gerarão erros detectáveis por ON ERROR.

Ensaio

Altere alguns dos valores sob o cabeçalho «QUANTITIES» (quantidades) para uma letra, executando depois o programa. Deverá ver a mensagem de erro do programa listada e também o módulo DATA. Corrija o engano deliberado e volte a executar o programa. Desta feita, ao premir duas vezes ESC (ou uma vez qualquer tecla, logo que o gráfico esteja acabado), deverá aparecer a listagem do módulo DATA.

PROGRAMA 2.3: GRÁFICO 3D

Função do programa

Depois de termos visto duas formas diferentes de apresentação de dados em modo de alta resolução, é conveniente lembrarmo-nos da imensa flexibilidade que o conjunto excelente de gráficos de baixa resolução do *464* proporciona. Utilizando os caracteres gráficos de baixa resolução, que não são acessíveis a partir do teclado, mas que pode ver no «Apêndice 3» do manual, o utilizador pode aproveitar os efeitos já prontos, que seria muito difícil conseguir em alta resolução.

No programa seguinte, criaremos um gráfico de barras tridimensional, cuja visualização é, penso eu, uma das melhores demonstrações de quão impressionante pode ser a baixa resolução no *464* — de facto, trata-se do tipo de visualização que o fará chamar toda a família para lhes mostrar como você é inteligente.

Novos conceitos apresentados no decurso do programa são:

1) Utilização do *Datacorder* (gravador de dados) para armazenar dados para o programa.
2) Utilização dos caracteres gráficos de baixa resolução.
3) Entrada de dados interactiva.

Módulo 2.3.1: Inicialização

Trata-se de um módulo claro, que declara um pequeno quadro e pergunta se devem ser introduzidos dados a partir da fita — adiante se falará mais sobre isto.

Módulo 2.3.1: linhas 2000-2080

```
2000 REM*********************************
2010 REM Initialise
2020 REM*********************************
2030 CLS
2040 PRINT TAB(17);"3D GRAPH";TAB(17);"=
=======":PRINT
2050 DIM hh(2,6)
2060 INPUT "Load data from tape (y/n)";q
$
2070 r$=CHR$(13)
2080 RETURN
```

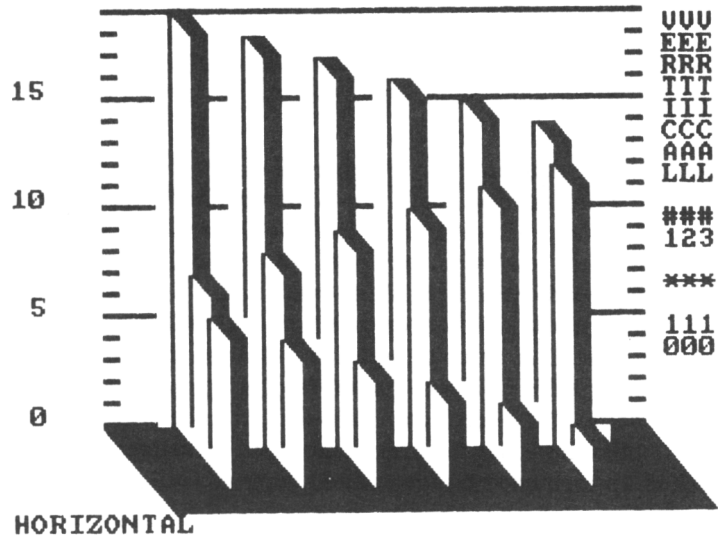


Fig. 2.3 — *Visualização do «Gráfico 3D»*

Comentário

Linha 2050: O quadro HH será utilizado para armazenar os dados para o gráfico. Uma vez que os valores na instituição DIM têm de ser contados a partir de zero, o que aqui se proporciona é espaço para três conjuntos de sete itens de dados.

Linha 2060: esta entrada de dados (INPUT) está concebida para permitir que um módulo posterior chame um conjunto de dados para um gráfico, a partir da fita.

Linha 2070: a cadeia R$ será utilizada no módulo do ficheiro de dados e as explicações podem esperar, até que se chegue a esse módulo.

Módulo 2.3.2: Aceitação dos dados

No caso deste último programa do capítulo, não iremos adoptar o estratagema de utilizar instruções de dados para armazenar informação alterada. A maioria dos programas que trabalham em informação útil terão em conta a introdução de tal informação pelo utilizador, durante a execução do programa — tais programas são designados «interactivos». No caso do programa presente, toda a informação pode ser reunida de uma só vez. Assim, o que vemos é um módulo que pede informação, que a utiliza para pedir mais informações e executa verificações de erros na entrada.

Módulo 2.3.2: linhas 3000-3250

```
3000 REM*********************************
3010 REM Accept Data
3020 REM*********************************
3030 CLS
3040 PRINT TAB(17);"3D GRAPH";TAB(17);"=
=======":PRINT
3050 PRINT"There are 19 units vertically
.":PRINT
3060 INPUT"Number represented by each un
it:",uv:PRINT
3070 INPUT"Columns (1-6):",nd:PRINT
3080 INPUT"Banks (1-3):",nb:PRINT
3090 PRINT"*****************************
***********":PRINT
3100 INPUT "Name for horizontal axis:",n
h$:PRINT
3110 FOR i=0 TO nb-1
3120 PRINT"Name for vertical axis ";i+1;
:INPUT nv$(i):PRINT
3130 NEXT i
3140 CLS
3150 FOR i=0 TO nb-1
3160 FOR j=1 TO nd
3170 t=20*uv
3180 WHILE t/uv>19
3190 PRINT:PRINT"Input Bank";i+1;"value"
;j;":";
3200 INPUT t
3210 IF INT(t/uv)>19 THEN PRINT:PRINT"**
**** Value too high ******"
3220 WEND
3230 hh(i,j)=t
3240 NEXT j,i
3250 RETURN
```

Comentário

Linhas 3050-3060: tal como no anterior programa do gráfico linear, cada unidade do eixo vertical pode representar qualquer valor especificado pelo utilizador. Note que, por estarmos a trabalhar em baixa resolução, não temos a mesma flexibilidade do programa anterior, no que respeita à dimensão das unidades verticais. A única dimensão prática para cada unidade é a altura de um carácter quadrado e o formato do gráfico permitirá dezanove unidades no eixo vertical.

Linhas 3070-3080: tal como foi mencionado anteriormente, o gráfico permitirá a apresentação de três conjuntos de seis itens, que serão visualizados sob a forma de três linhas, até um máximo de seis barras sólidas. A cada linha de barras chamar-se-á um «banco» e às barras individuais «colunas».

Linhas 3110-3130: visto que pode haver até três bancos, será preciso que haja até três nomes para o eixo horizontal, como por exemplo, preço de venda, custo, lucro. Note-se que os nomes devem ser colocados no quadro NV$ (Nome da Vertical), a partir do elemento 0 — este quadro não está dimensionado na rotina de inicialização, porque em caso algum terá mais ou menos de três elementos. A simples menção do nome do quadro no programa estrutura o quadro, automaticamente, com dez elementos (0-9).

Linhas 3150-3240: é solicitada ao utilizador a introdução, por cada banco, dos dados para o número de colunas especificado. O número introduzido é verificado, para assegurar que não irá deslocar o gráfico acima da marca das dezanove unidades. O objectivo do ciclo nas linhas 3180-3220 é assegurar que nenhum valor será introduzido, que requeira um gráfico maior do que dezanove quadrados de carácter no écran. Se um tal valor for introduzido, o utilizador será avisado do facto e solicitado a repetir a introdução.

Ensaio

Introduza as linhas que se seguem (algumas tornar-se-ão, eventualmente, parte do módulo de controlo do programa):

```
1030 GOSUB 2000
1040 GOSUB 3000
1110 CLS : END
```

Execute agora o programa introduzido até este momento e responda às suas perguntas do modo seguinte:

> *Number represented by each unit* (número representado por cada unidade): 1
> *Columns* (colunas): 1
> *Banks* (bancos): 1
> *Name for vertical axis 1* (nome do eixo horizontal): HORIZONTAL
> *Name for vertical axis 1* (nome do eixo vertical): VERTICAL 1
> *Bank 1, value 1* (banco 1, valor 1): 10

Após a introdução do último valor, o écran limpa e o programa pára, com a mensagem «READY» (pronto). Introduza agora:

```
? UV,ND,NB,NH$,NV$(0),HH(0,1)
```

e o resultado deverá ser:

```
1               1               1
HORIZONTAL  VERTICAL 1      10
```

Se quiser, pode executar o programa novamente e tentar introduzir valores falsos. Não deverá poder introduzir qualquer valor de coluna que seja maior do que dezanove vezes o valor de uma só unidade do eixo vertical.

Módulo 2.3.3: Desenho da estrutura do gráfico

Tal como o gráfico linear, também este precisa do desenho prévio de uma estrutura, antes de a informação que apresenta se tornar compreensível. A tarefa é efectuada pelo presente módulo, o qual utiliza ciclos, variáveis e o comando LOCATE para colocar caracteres de baixa resolução nas posições correctas no écran.

Módulo 2.3.3: linhas 4000-4260

```
4000 REM*********************************
4010 REM Draw Framework
4020 REM*********************************
4030 CLS
4040 FOR i=0 TO 3
4050 LOCATE 6+i,21+i
4060 PEN 2:PRINT CHR$(213);STRING$(29,CH
R$(143));CHR$(215)
4070 NEXT i:PEN 1
4080 LOCATE 6,1:PRINT STRING$(30,CHR$(21
0))
4090 FOR i=2 TO 20
4100 LOCATE 6,i:PRINT CHR$(210)
4110 LOCATE 35,i:PRINT CHR$(210)
4120 NEXT i
4130 FOR i=5 TO 20 STEP 5
4140 LOCATE 6,i:PRINT STRING$(30,CHR$(21
0))
4150 NEXT i
4160 LOCATE 1,25:PRINT nh$
4170 FOR h=0 TO nb-1
4180 PEN h+1
4190 tt$=nv$(h)+" *"+STR$(uv)
4200 FOR i=1 TO LEN (tt$)
4210 LOCATE 37+h,i+1:PRINT MID$(tt$,i,1)
4220 NEXT i,h
```

```
4230 FOR i=0 TO 15 STEP 5
4240 LOCATE 1,20-i:PRINT USING "##";i
4250 NEXT i
4260 RETURN
```

Comentário

Linhas 4040-4070: este ciclo imprime uma base, sobre a qual irá erguer-se o gráfico. Os caracteres gráficos estão listados no «Apêndice 3» do seu manual.

Linha 4080: uma linha ao longo do topo do écran, consistindo no número de carácter 210 (uma barra horixontal), trinta vezes.

Linhas 4090-4120: são as fileiras verticais de marcas que são colocadas em cada terminal do gráfico, para representar as dezanove unidades possíveis.

Linhas 4130-4150: quatro linhas que atravessam transversalmente o gráfico, representando unidades de cinco no eixo vertical.

Linha 4160: é a etiqueta do eixo horizontal.

Linhas 4170-4220: estes ciclos imprimem o(s) nome(s) do eixo vertical ao longo da margem direita do écran, juntamente com o valor representado por cada unidade. O valor de ciclo I é utilizado, tanto para movimentar nomes através do eixo, como para modificar a cor de cada nome. A cor de cada título corresponderá à cor de um dos bancos.

Linhas 4230-4250: este ciclo coloca números junto aos marcadores de cinco unidades, na esquerda do gráfico.

Ensaio

Introduza mais uma linha:

```
1050 GOSUB 4000
```

e execute agora o programa. Especifique o valor unitário, uma coluna e três bancos. Forneça os três nomes para o eixo vertical. Quando solicitado a especificar o valor das colunas, apenas precisará de premir RETURN. Deverá então obter a visualização da estrutura do gráfico, com o nome do eixo horizontal ao fundo e os nomes do eixo vertical em cima, à direita.

Módulo 2.3.4: Desenho do gráfico

Este módulo é, na verdade, mais fácil de compreender após ter sido introduzido e observado o seu efeito. O módulo não se baseia tanto em princípios gerais como em resultados da experimentação, em que se procura ver quais as combinações de caracteres, e as posições em que estão impressos, que proporcionaram o efeito desejado.

Módulo 2.3.4: linhas 5000-5220

```
5000 REM*********************************
5010 REM Draw Blocks
5020 REM*********************************
5030 FOR h=0 TO nb-1
5040 PEN h+1
5050 FOR i=nd TO 1 STEP -1
5060 condition=1:WHILE INT(hh(h,i)/uv)<>
0 AND condition
5070 FOR j=1 TO INT(hh(h,i)/uv)+1
5080 LOCATE 9+4*(i-1)+h,22+h-j
5090 IF j=1 THEN PEN 2:PAPER 0:PRINT CHR
$(143);CHR$(215);:PEN h+1:PRINT CHR$(143
)
5100 IF j>1 AND j<INT(hh(h,i)/uv)+1 THEN
 PRINT CHR$(209);" ";CHR$(143)
5110 IF j=INT(hh(h,i)/uv)+1 THEN PRINT C
HR$(209);CHR$(213);CHR$(215)
5120 NEXT j
5130 condition=0:WEND
5140 NEXT i
5150 NEXT h
5160 FOR i=0 TO nd-1
5170 FOR j=1 TO nb
5180 LOCATE 9+4*i+j,20+j
5190 IF j>1 OR hh(2,i)=0 OR (j=1 AND nd=
0) THEN PEN 2:PRINT CHR$(215)
5200 NEXT j
5210 NEXT i
5220 RETURN
```

Comentário

Linhas 5030-5150: este ciclo cria o número de bancos especificado.

Linhas 5050-5140: o segundo ciclo criará o número de colunas especificado, funcionando da direita para a esquerda.

Linhas 5060-5130: estas linhas criam um ciclo fictício WHILE para operar em torno da secção que desenha uma coluna, se o valor dessa coluna for 0 — o ciclo é fictício porque nunca é executado mais de uma vez, em quaisquer circunstâncias. Isto é assegurado tornando o valor da variável CONDITION uma das condições de repetição do ciclo. A redução de CONDITION a 0 antes de WEND ser alcançado, na linha 5130, assegura que o ciclo seja sempre executado uma única vez. Torna-se útil lembrar esta técnica, já que ao *464* falta um comando EXIT, o qual permitiria que o ciclo terminasse automaticamente.

Linhas 5070-5120: este ciclo constrói uma coluna. A linha 5080 fornece a posição de desenho de cada carácter da coluna. A posição começará no extremo direito do número de colunas (representado por I), deslocar-se-á para cima, pela coluna individual (ditada por J), e quatro espaços para a esquerda, por cada nova coluna. Para além disto, quando um banco é terminado, o banco de colunas seguinte (representado por H) será impresso um espaço abaixo e para a direita do anterior.

A variável de ciclo J é, tal como foi dito, regulada para se deslocar de 1 até à altura da coluna. Na primeira passagem pelo ciclo, é criada a base da coluna (linha 5090). Em passagens subsequentes, são usados diferentes caracteres para representar a parte lateral e frontal da coluna, à medida que vai sendo construída, utilizando LOCATE para posicionar a impressão. Finalmente, é acrescentado o topo da coluna (linha 5110).

Linhas 5160-5210: depois de terminadas as colunas, restam algumas arestas por limar, ao fundo, e estas linhas encarregam-se dessa tarefa.

Ensaio

Introduza uma nova linha:

```
1060 GOSUB 5000
```

e execute o programa. Especifique um valor de unidade 1, três colunas e três bancos. Os nomes dos eixos não são relevantes, por isso, escolha à vontade. Quando lhe forem solicitados os valores das colunas, introduza o seguinte:

```
6,12,18,6,12,18,6,12,18
```

Deverá agora obter a visualização clara dos três bancos e das três colunas, parecendo que os topos de cada uma das três colunas formam uma superfície plana, desde o banco de trás até ao da frente. Note que, ao ler os valores dos três bancos, deve partir do princípio de que o topo da barra mais adiantada continua para trás, até à posição mais atrasada, sendo o valor da coluna lido a partir do limite traseiro da coluna, tal como aparecerá no banco mais atrasado. No exemplo do écran, o que se vê são três colunas, representando as três barras de cada coluna o mesmo valor, embora, fisicamente, a barra frontal apareça mais baixa no écran. Isto é necessário para preservar a ilusão tridimensional.

Faça experiências com o programa, para ver como ele trata diferentes valores de

dados. Verificará que apenas funciona com dados em que um banco nunca seja mais alto do que o que se encontra atrás. Existe muito material adequado a este tipo de padrão, tal como o dos dados de preço/custo/lucro mencionados anteriormente.

Módulos 2.3.5. e 2.3.6: Armazenamento de dados em fita

Passamos agora a uma área que muitas pessoas negligenciam nos programas que fazem, à sua própria custa — o armazenamento de dados em fita. Se você cometeu um ou dois erros na entrada do programa, sem dúvida descobriu que a reintrodução dos dados pode tornar-se um pouco mais do que irritante, ao fim de algum tempo. Além disto, em muitos programas há pouca vantagem em colocar os itens de dados no computador se, no fim e apesar de tudo, você tiver de se lembrar deles de cada vez que desliga o *464* — talvez que, com o programa corrente, não seja completamente impraticável introduzir os dados de novo. Mas que se passa com programas mais complexos, em que podem existir, literalmente, centenas de itens de informação?

Todos estes problemas podem ser ultrapassados com a utilização permanente do gravador de cassetes para registar os dados introduzidos e, após tê-lo feito, voltar a passá-los para o *464* sempre que o utilizador o deseje. Tal tarefa é concretizada por estes dois módulos.

Módulos 2.3.5. e 2.3.6: linhas 6000-7110

```
6000 REM*********************************
6010 REM Save Data to Tape
6020 REM*********************************
6030 OPENOUT "graphdata"
6040 PRINT #9,nb;r$;nd;r$;nh$;r$;uv
6050 FOR i=0 TO nb-1
6060 PRINT #9,nv$(i)
6070 FOR j=0 TO nd
6080 PRINT #9,hh(i,j)
6090 NEXT j,i
6100 PRINT #9:CLOSEOUT
6110 RETURN
7000 REM*********************************
7010 REM Load Data from Tape
7020 REM*********************************
7030 OPENIN "graphdata"
7040 INPUT #9,nb,nd,nh$,uv
7050 FOR i=0 TO nb-1
7060 INPUT #9,nv$(i)
7070 FOR j=0 TO nd
7080 INPUT #9,hh(i,j)
7090 NEXT j,i
```

```
7100 CLOSEIN
7110 RETURN
```

Comentário

Linha 6030: antes de os dados poderem ser armazenados em fita, deve ser preparado lugar para eles, processo conhecido por «abertura de um ficheiro». O formato correspondente é:

OPENOUT "FILENAME"

Linhas 6040-6090: depois de termos aberto o ficheiro, tudo o que é preciso fazer é imprimir-lhe informação. Isto é conseguido através de um comando especial PRINT #. Os itens impressos no ficheiro são formados pelo conteúdo do quadro principal (HH$) e pelas variáveis principais.

Linha 6040: uma coisa a salientar quanto à instrução PRINT # é a presença de vários R$ na linha. Deve lembrar-se que, no primeiro módulo do programa, R$ foi igualado a CHR$(13), que é o carácter RETURN e significa o sinal de um item a ser impresso. Quando se imprimem diversos itens num ficheiro, a partir de uma só instrução PRINT #, todos os itens são executados em conjunto, a menos que seja incluído R$ entre os itens a serem impressos.

Linha 6060: no comentário sobre a última linha, afirmou-se que R$ (ou qualquer outra variável igual a CHR$(13)) devia ser incluído para separar os itens — então, por que razão isso não é feito no caso destes dois ciclos, os quais imprimem em fita o conteúdo de dois quadros do ficheiro? A resposta é que, sempre que uma instrução PRINT, ou PRINT #, termina sem pontuação, o *464* segue automaticamente o último item impresso com um carácter RETURN — é por isto que os itens são impressos em linhas separadas do écran, caso o item precedente não tenha uma vírgula ou ponto e vírgula no final.

Linha 6100: quando se imprime num ficheiro, é aconselhável terminar com um único e vazio PRINT # <FILE NUMBER>, o que assegura que quaisquer itens na memória do computador ainda à espera de serem impressos serão apagados. Finalmente, o ficheiro deve ser fechado (CLOSE), utilizando CLOSEOUT num ficheiro em que tenham sido salvaguardados dados. Se isto não for efectuado, significa que o número de ficheiro não estará disponível para futura utilização e que mesmo os dados em fita poderão perder-se, quando o programa tentar voltar a lê-los.

Linhas 7000-7110: estas linhas executam a função oposta às de cima, ou seja, voltam a chamar dados previamente armazenados a partir da fita.

Linha 7030: uma vez mais, deve ser aberto (OPEN) um ficheiro. A única diferença entre esta instrução OPEN e a anterior é que se utiliza a forma OPENIN de OPEN, dizendo ao sistema que pretendemos apanhar dados *da* fita.

Linhas 7040-7090: o oposto de PRINT # é INPUT #. Note-se que não precisa-

mos de utilizar o separador R$ ao introduzir (INPUT) dados. É da natureza de INPUT e INPUT # não reconhecerem que receberam um item de dados até que ENTER seja premido ou um carácter RETURN lido a partir da fita.

Linha 7100: tal como acontece na linha 6100, o ficheiro deve ser fechado (CLOSE) quando se tiver terminado.

Ensaio

Será melhor deixar o ensaio deste módulo até terem sido introduzidas as poucas linhas necessárias para completar o módulo de controlo do programa.

Módulo 2.3.7: O módulo de controlo

Foram já introduzidas muitas das linhas deste módulo ao construírem-se os procedimentos de ensaio ao programa. Tudo o que resta fazer é assegurar que o módulo seja completado, voltando a verificar a listagem seguinte.

Módulo 2.3.7: linhas 1000-1110

```
1000 REM*********************************
1010 REM Control
1020 REM*********************************
1030 GOSUB 2000
1040 IF LOWER$(q$)="y" THEN GOSUB 7000 E
LSE GOSUB 3000
1050 GOSUB 4000
1060 GOSUB 5000
1070 WHILE INKEY$=""
1080 WEND
1090 LOCATE 1,25:INPUT "Save data to tap
e (y/n)";q$
1100 IF LOWER$(q$)="y" THEN GOSUB 6000
1110 CLS:END
```

Ensaio

Execute apenas o programa. Deve agora ser capaz de introduzir os dados para um gráfico. Quando o gráfico tiver sido visualizado, a pressão de qualquer tecla resultará na solicitação da salvaguarda dos dados. Antes de responder «Y», assegure-se de que se encontra no gravador de dados uma fita apropriada (com certeza não pretende escrever um programa por cima do outro). Quando o programa tiver terminado a salvaguarda dos dados, volte a executá-lo e, desta vez, responda «Y» se ele lhe perguntar se pretende carregar a partir da fita. Deverá agora obter a visualização do mesmo gráfico. Se este ensaio for satisfatório, o programa estará pronto para utilização.

CAPÍTULO 3

«Son et lumière»

Vamos observar mais de perto, neste capítulo, as possibilidades do *464* no que respeita ao som e gráficos. Na maioria dos casos, os programas que você escrever terão algumas pequenas rotinas incluídas, para desempenharem tarefas gráficas ou de som, ainda que seja apenas algo tão trivial como o fácil módulo de título ilustrado no último capítulo ou o alarme de dois tons. Noutros casos, porém, há necessidade de algo mais ambicioso, um desenho complexo ou uma peça musical para alegrar um programa. Nestes casos, em vez de se escrever uma rotina separada para criar o desenho ou melodia, de cada vez que ela é necessária, torna-se mais prático ter à mão algumas ferramentas que lhe permitam criar desenhos mais facilmente, chamando-as depois a partir da fita, para utilização em programas subsequentes.

Neste capítulo encontrará três dessas ferramentas, que lhe permitirão criar desenhos em alta resolução, desenhar os seus próprios conjuntos de caracteres e escrever e editar música no seu *464*. Com as vantagens proporcionadas pelos programas deste capítulo, o único limite à utilização de som e gráficos nos seus próprios programas será a imaginação que conseguir arranjar.

Os programas incluídos neste capítulo são:

CARACTERES: permite a criação de conjuntos de caracteres ao gosto de cada um.

DESENHADOR: uma ferramenta para criar desenhos em alta resolução.

MÚSICA: permite-lhe introduzir melodias bipartidas num formato fácil e reproduzi-las.

PROGRAMA 3.1: CARACTERES

Função do programa

Depois de termos visto, no capítulo anterior, algo do que pode obter-se com gráficos de alta resolução, passamos ao outro lado do problema, os gráficos de baixa resolução, com um programa que lhe permite alterar a forma dos caracteres que o *464* imprime no écran. Mas, antes de passarmos ao próprio programa, é necessária

uma palavra de explicação sobre a forma como são criados e impressos os caracteres em modo de baixa resolução.

O écran de baixa resolução do *464* tem espaço para vinte e cinco linhas, cada uma com quarenta caracteres, num total de mil espaços de caracteres. Por outras palavras, você poderia imprimir no écran mil itens separados, ainda que alguns deles se repetissem, já que o *464* não pode gerar mil caracteres *diferentes* ao mesmo tempo. Este milhar, no entanto, não põe fim à história. Se se observar de perto qualquer carácter do écran, verificar-se-á que não são formados por linhas compactas, tal como as palavras com que se escrevem estas linhas, mas antes por pontos. De facto, cada um dos mil espaços de caracteres do écran é constituído por sessenta e quatro posições de pontos e são as combinações destes sessenta e quatro pontos que formam cada carácter que o *464* consegue visualizar. A letra «A», por exemplo, tem a constituição que se apresenta na figura 3.1.

```
                ==================
ROW 1 -->         0 0
    2 -->       0 0 0 0
    3 -->     0 0     0 0
    4 -->     0 0 0 0 0 0
    5 -->     0 0     0 0
    6 -->     0 0     0 0
    7 -->     0 0     0 0
    8 -->
                ==================
```

Fig. 3.1 — *Ampliação da letra «A» tal como aparece no écran*

Os pontos que constituem os caracteres são conhecidos por *pixels,* que é a abreviatura de *«picture elements»,* e representam o item mais pequeno que o *464* pode tratar no écran, quer em modo de alta quer de baixa resolução.

Formas tão complexas não aparecem por acaso. Parece claro que, algures na memória do *464,* deve estar estabelecido que, quando você prime a tecla marcada com «A», o padrão de pontos mostrado na ilustração apareça no écran. De facto, todos os caracteres que o *464* pode imprimir estão armazenados sob a forma de números, num bloco de memória chamado «memória de caracteres» ou «carácter ROM». A cada carácter são atribuídos oito *bytes* de memória, e cada um destes *bytes* determina o local onde os *pixels* irão aparecer, numa das fiadas do carácter. Isto é conseguido transformando o valor de cada *byte* numa imagem da fiada do sistema binário de numeração utilizado pelo *464,* em que os números são expressos como potências do algarismo 2, em vez do número 10, tal como acontece com o nosso sistema normal (decimal) de contagem. Assim:

2013006

significa, no nosso modo de contagem usual:

(2*10^6) + (0*10^5) + (1*10^4) + (3*10^3) + (0*10^2) + (0*10^1) + (6*10^0)

No sistema binário, no entanto, os únicos dígitos permitidos são o «1» e o «0», pelo que um número como:

11001010

significa:

(2 ^7) + (2 ^6 + (2 ^3) + (2 ^1) — ignorando os zeros.

Não é necessário um conhecimento completo do sistema binário, desde que você se lembre de que um único *byte* de memória no *464* é capaz de conter um número binário de oito dígitos e que todos aqueles 1 e 0 são uma forma perfeita de registar quais os *pixels* de uma fiada de um carácter que são «ligados». A letra «A», por exemplo, transforma-se nos oito números binários que se seguem:

0 0 0 1 1 0 0 0
0 0 1 1 1 1 0 0
0 1 1 0 0 1 1 0
0 1 1 0 0 1 1 0
0 1 1 1 1 1 1 0
0 1 1 0 0 1 1 0
0 1 1 0 0 1 1 0
0 0 0 0 0 0 0 0

Se olhar com atenção, ainda pode distinguir o «A» com bastante clareza, desta vez impresso em «1», em vez de *pixels,* embora você tivesse algumas dificuldades em o reconhecer como:

24,60,102,102,126,102,102,0

que é a forma correspondente aos números binários, quando traduzidos para o nosso sistema mais confortável de numeração decimal.

O objectivo de tudo isto é conduzir ao facto de que, quando solicitado a imprimir um carácter no écran, o *464* procura-o na «memória de caracteres» e utiliza o que aí encontra para desenhar o carácter no écran.

Decorre de tudo isto que, se fosse possível *alterar* o conteúdo da memória de caracteres, a forma dos caracteres impressos no écran alterar-se-ia também. Poderíamos determinar os tipos de letra, novos caracteres gráficos ou qualquer outra coisa que coubesse no básico quadrado gráfico de 8*8.

O problema, no entanto, é que a memória de caracteres *não pode* ser alterada. Faz parte da ROM, ou «*read only memory*» (memória morta)[1], permanentemente incluída no *464*. O que *podemos* fazer é copiá-la para algures na RAM, ou tipo de memória que pode ser alterado, utilizando depois os dados alterados em conjugação

[1] Memória que apenas permite leitura e cujo programa foi preestabelecido em circuitos, díodos ou transístores. *(N. do T.)*

com o poderoso comando SYMBOL, para modificar os próprios caracteres. Desta forma, é possível manipular o conjunto de caracteres como se queira — e é este o objectivo do programa seguinte.

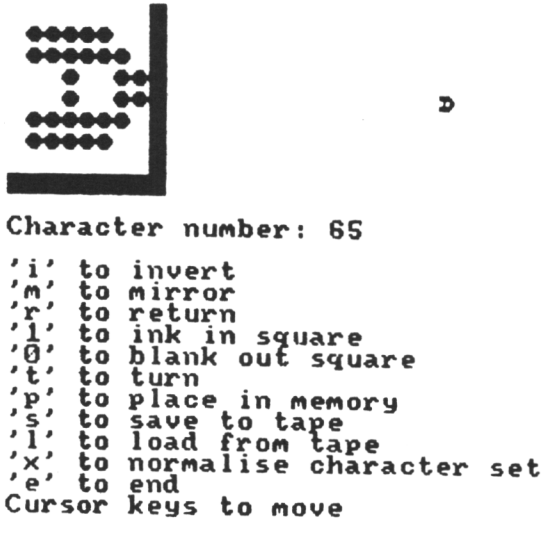


Fig. 3.2 — *Visualização do «Gráfico 3D»*

A visualização mostra o écran durante o modo de edição de caracteres, com uma letra «A» que foi rodada 90 graus.

Módulo 3.1.1: O módulo de controlo

Em condições normais, não apresentaríamos um módulo de controlo no início do processo de introdução de um programa, mas este é curto e, sem ele, o programa necessitará de um GOSUB para ser introduzido, antes de começar em condições convenientes.

Módulo 3.1.1: linhas 10000-10050

```
10000 REM*******************************
10010 REM Control
10020 REM*******************************
10030 GOSUB 11000
10040 GOSUB 12000
10050 CLS:END
```

Módulo 3.1.2: Comutação para o novo conjunto de caracteres

O objectivo deste módulo é levar a bom termo a tarefa delineada na introdução do programa, ou seja, a de transferir os dados dos caracteres para alguma parte de memória onde possam ser manipulados. De facto, o local onde eles ficarão armazenados é um quadro denominado C%, o que nos permite todo o género de flexibilidade no armazenamento e alteração dos dados.

Módulo 3.1.2: linhas 11000-11190

```
11000 REM*******************************
11010 REM Initialise
11020 REM*******************************
11030 CLS:PEN 1:PRINT "First user define
d character":INPUT "(32-255)";fc
11040 IF fc<32 OR fc>255 THEN PRINT "
**** OUT OF RANGE: TRY AGAIN ****":GOTO
11030
11050 SYMBOL AFTER fc
11060 IF in=0 THEN ch=fc:DIM a%(7),t%(7)
,c%(255,7):in=1
11070 menu$="imr10tpslxe"+CHR$(240)+CHR$
(241)+CHR$(242)+CHR$(243)
11080 MODE 2
11082 LOCATE 36,14:PRINT "PLEASE WAIT"
11084 LOCATE 1,1
11090 FOR i=fc TO 255
11100 PRINT CHR$(i);
11110 NEXT i
11120 FOR i=0 TO 255-fc
11130 FOR j=0 TO 7
11140 c%(fc+i,j)=PEEK(48*1024+j*2048+i)
11150 NEXT j
11160 NEXT i
11170 MODE 1
11180 a$="r"
11190 RETURN
```

Comentário

Linhas 11030-11050: o *464* proporciona-lhe a oportunidade de definir qualquer coisa a partir de 16 até 255 dos seus 255 caracteres — mas você apenas pode definir mais de dezasseis se lhe transmitir que o deseja fazer, utilizando o comando SYMBOL AFTER. SYMBOL AFTER, seguido de um número, significa apenas que,

a partir desse momento, você deseja poder alterar qualquer carácter a partir do representado pelo número a seguir a SYMBOL AFTER até ao carácter número 255, utilizando o comando SYMBOL, que iremos abordar daqui a pouco. Estas linhas permitem-lhe capacitar o primeiro carácter para ser redefinido em qualquer ponto, desde o carácter número 32 até ao 255. Em utilização normal, o *464* permite-lhe redefinir dezasseis caracteres, do número 240 em diante.

Linha 11060: excepto num caso específico durante a utilização do programa, esta linha será usada para estabelecer o quadro principal, C%, que conterá os dados dos caracteres (256 linhas com oito números em cada linha) e dois quadros que são usados para a manipulação temporária de um carácter individual.

Linha 11070: trata-se de uma linha importante, cujo significado será explicado quando chegarmos ao módulo que trata do *menu* do programa.

Linhas 11080-11170: um pedaço trabalhoso, este. Em vez de retirar os dados da memória, é mais simples fazê-lo a partir de um local óbvio, o próprio écran. A vantagem disto é que, no futuro, se você desejar recolher dados para caracteres seleccionados, tudo o que é preciso é imprimir esses caracteres no écran. Para isto, precisamos primeiro mudar para o modo 2 (MODE 2) do écran, o modo de alta resolução. A razão disto é que, em modo 2, a disposição dos *pixels* no écran corresponde exactamente ao padrão descrito na introdução do programa. Em modo 1 (MODE 1), os dígitos binários individuais não são copiados exactamente pelos *pixels* no écran, visto que são impressos *dois pixels,* horizontalmente, por cada dígito binário.

Após a mudança de modo, a tarefa seguinte é apenas imprimir todos os caracteres de FC (*First Character,* ou seja, o primeiro carácter especificado pelo utilizador) até 255, da primeira posição no écran para a frente. O segundo par de ciclos (linhas 11120-11160) lê agora os *bytes* de memória que constituem a visualização do écran. Este processo não é tão directo quanto podia ser, devido à forma como a memória de écran está concebida. O bloco de memória que regista o conteúdo do écran está assim concebido (não entraremos em razões) para que, se se pretender encontrar os oito *bytes* que formam o carácter no canto superior esquerdo (em MODE 2), seja preciso procurar no *byte* de memória 49152 o *byte* superior do carácter, depois no *byte* 51200 (49152 + 2048) o seguinte abaixo do anterior, e assim por diante, em passos de 2048 *bytes* por cada um dos oito *bytes.* O carácter seguinte para a direita começa no *byte* 49153 e continua para baixo, em passos de 2048 *bytes. Quando é atingido o fim da primeira linha de caracteres, a deslocação de um byte* para a frente conduz-nos até ao primeiro *byte* do primeiro carácter da segunda linha. Se ainda não ficou, de modo algum, esclarecido quanto a tudo isto, experimente introduzir o pequeno programa de ensaio que se segue:

```
10 MODE 2
20 FOR I = 49152 TO 49152 + 1024
30 POKE I,255
40 NEXT I
```

Verificará que, embora esteja a percorrer (POKE) a memória de écran em passos de um *byte,* a linha criada com a activação dos *pixels* se encontra ao longo do topo das linhas de caracteres.

Já de posse destes elementos, você deve poder ver aquilo que a linha 11140 está a fazer, enquanto recolhe um *byte* de cada vez, a partir dos dados de caracteres da memória de écran.

Linhas 11180: esta linha estabelece uma variável utilizada para armazenar respostas de *menu,* de modo que o programa não termine acidentalmente aquando da primeira execução.

Módulo 3.1.3: Visualização de um carácter ampliado

A essência deste programa é tornar mais fácil a edição de caracteres. Uma forma de obter isto é imprimindo uma versão ampliada de um carácter especificado, para que possa deslocar-se um cursor em torno dele. Este módulo permite a especificação do carácter e imprime-o.

Módulo 3.1.3: linhas 12000-12220

```
12000 REM*******************************
12010 REM Print Grid
12020 REM*******************************
12030 WHILE a$<>"e"
12040 CLS:PEN 1
12050 FOR i=0 TO 7
12060 IF done=0 THEN a%(i)=c%(ch,i)
12070 b$=RIGHT$(BIN$(256+a%(i)),8)
12080 FOR j=1 TO 8
12090 IF MID$(b$,j,1)="1" THEN PRINT CHR
$(231); ELSE PRINT " ";
12100 NEXT j
12110 PEN 3:PRINT CHR$(143):PEN 1
12120 NEXT i:done=1
12130 PEN 3:PRINT STRING$(9,CHR$(143)):P
EN 1
12140 LOCATE 25,5:PRINT CHR$(ch)
12150 PEN 3:LOCATE 1,11
12160 PRINT "Character number:";ch:PRINT
12170 IF a$="r" THEN INPUT "Number to mo
ve pointer (0 to redefine)  ";mm:ch=ch+m
m
12180 IF ch<fc THEN ch=fc
12190 IF ch>255 THEN ch=255
12200 IF mm=0 THEN GOSUB 13000 ELSE done
```

```
=0
12210 WEND
12220 RETURN
```

Comentário

Linhas 12040-12120: os dados de carácter para um determinado carácter são lidos do quadro C% para o quadro A% e depois utilizados para imprimir uma versão ampliada do carácter no écran. O modo de conseguir isto é, em primeiro lugar, usar BIN$ para traduzir cada *byte* dos dados de caracteres para uma cadeia de algarismos 1 e 0. Depois, é impresso um carácter circular preenchido (CHR$(231)), para corresponder à posição de um 1 na cadeia. O padrão de 8 * 8 é rodeado por uma margem compacta formada por CHR$(143). Finalmente, a variável DONE é colocada no final, para que, se o módulo for repetido sem alterações ao carácter, os dados não sejam recopiados para A%.

Linha 12140: é colocada à direita da versão ampliada uma outra versão do carácter, em tamanho normal.

Linhas 12170-12190: quando o programa é executado pela primeira vez, o carácter visualizado será o primeiro da fiada a poder ser redefinido, tal como se encontra armazenado na variável FC. Estas linhas permitem a focagem do carácter, para as deslocações deste dentro da fiada limitada por FC e 255.

Ensaio

Execute o programa e especifique 65 como o primeiro carácter definido pelo utilizador. Depois de a fiada do carácter ter sido visualizada e depois de uma pausa, para permitir a transferência dos dados, deverá ver uma versão ampliada do símbolo «A» desenhada no canto superior direito. Deverá também poder deslocar-se pelo conjunto de caracteres, examinando qualquer carácter que deseje, a partir do A. Note que ainda não pode *fazer* nada ao carácter visualizado. Isso será conseguido, eventualmente, premindo «0» para chamar um módulo subsequente. Normalmente, o programa já teria parado ao ser introduzido o modo «0», mas pode fazê-lo neste módulo premindo ESC.

Módulo 3.1.4: Um cursor definido pelo utilizador

É um módulo simples, destinado a visualizar a «imitação» de um cursor, num local escolhido.

Módulo 3.1.4: linhas 14000-14110

```
14000 REM********************************
14010 REM Display Cursor
```

```
14020 REM*******************************
14025 b$=RIGHT$(BIN$(256+a%(y)),8)
14030 LOCATE x+1,y+1:PAPER 2:PEN 1
14040 IF MID$(b$,x+1,1)="1" THEN PRINT C
HR$(231); ELSE PRINT " ";
14050 a$=""
14060 WHILE a$=""
14070 a$=LOWER$(INKEY$)
14080 WEND
14090 LOCATE x+1,y+1:PAPER 0
14100 IF MID$(b$,x+1,1)="1" THEN PRINT C
HR$(231); ELSE PRINT " ";
14110 RETURN
```

Comentário

Linhas 14030-14040: a aparência do cursor é criada através da alteração das cores de papel (PAPER) e tinta (INK), de forma a simular a presença do cursor normal, reimprimindo depois, ou um espaço, ou o pequeno círculo preenchido utilizado para visualizar versões ampliadas dos caracteres.

Linhas 14050-14080: trata-se de um estado de espera, até que seja premida uma tecla.

Linhas 14090-14100: a posição do carácter em que se encontra localizado o cursor é reimpresso em cores normais.

Ensaio

Introduza:

RUN 14000[RETURN]

e deverá ver um quadrado-cursor no canto superior esquerdo do écran. Prima uma tecla e o programa parará, com uma mensagem de erro «Unexpected RETURN».

Módulo 3.1.5: Inserção de comandos

Este módulo está organizado para apresentar um *menu,* para deslocar o cursor intermitente, para iluminar ou apagar os *pixels* ampliados e para aceitar comandos que manipulem o carácter corrente. Instruções completas quanto à sua utilização estão contidas no *menu*. O único aspecto original acerca do módulo é a forma como são aceites as instruções.

Módulo 3.1.5: linhas 13000-13220

```
13000 REM*******************************
13010 REM Redefine Character
13020 REM*******************************
13030 LOCATE 1,13:PRINT STRING$(40," ")
13040 LOCATE 1,13
13050 PRINT "'i' to invert"
13060 PRINT "'m' to mirror"
13070 PRINT "'r' to return"
13080 PRINT "'1' to ink in square"
13090 PRINT "'0' to blank out square"
13100 PRINT "'t' to turn"
13110 PRINT "'p' to place in memory"
13120 PRINT "'s' to save to tape"
13130 PRINT "'l' to load from tape"
13140 PRINT "'x' to normalise character
set"
13150 PRINT "'e' to end"
13160 PRINT "Cursor keys to move"
13170 ret=0:WHILE ret=0
13180 GOSUB 14000
13190 z=INSTR(menu$,a$)
13200 ON z GOSUB 15000,16000,17000,18000
,19000,20000,21000,22000,23000,24000,250
00,26000,27000,28000,29000
13210 WEND
13220 RETURN
```

Comentário

Linha 13170-13210: deve lembrar-se, com certeza, de que no módulo de inicialização organizámos uma cadeia bastante invulgar chamada MENU$. Esta linha é a razão para isso. Juntas, permitem que se faça uma tradução económica de uma só tecla premida para um número, através da utilização de INSTR. A posição de um carácter incluído em MENU$ é o valor que será atribuído a Z e utilizado para o ON Z GOSUB, na linha seguinte. A variável RET é usada para indicar ao módulo se toda a visualização precisa de ser redesenhada. Se RET for igual a 0 ao regressar da execução, então o carácter ampliado não será redesenhado, poupando-se tempo.

Ensaio

Execute o programa. Quando a letra «A» tiver sido desenhada, deverá poder introduzir «0» e obter a visualização do *menu*. Nenhum dos comandos terá qualquer efeito para além da paragem do programa com uma mensagem de erro.

Módulo 3.1.6: Retorno do modo de edição

Premindo «R» a partir do *menu* principal, retorna-se a execução ao módulo anterior, permitindo assim ao utilizador deslocar-se até outro carácter depois de ter tratado do corrente.

Módulo 3.1.6: linhas 17000-17030

```
17000 REM*******************************
17010 REM Return
17020 REM*******************************
17030 ret=1:RETURN
```

Ensaio

Deverá agora poder mover-se entre o *menu* principal e os módulos de movimentação de caracteres.

Módulo 3.1.7: Deslocação do cursor

Após termo-nos atribuído um cursor, acrescentamos agora a possibilidade de o deslocar, o que se faz alterando simplesmente so valores das variáveis de coordenada do cursor X e Y.

Módulo 3.1.7: linhas 26000-29040

```
26000 REM*******************************
26010 REM Up
26020 REM*******************************
26030 IF y>0 THEN y=y-1
26040 RETURN
27000 REM*******************************
27010 REM Down
27020 REM*******************************
27030 IF y<7 THEN y=y+1
27040 RETURN
28000 REM*******************************
28010 REM Left
28020 REM*******************************
28030 IF x>0 THEN x=x-1
28040 RETURN
29000 REM*******************************
29010 REM Right
29020 REM*******************************
29030 IF x<7 THEN x=x+1
29040 RETURN
```

Ensaio

Com a visualização do *menu* principal, deverá agora poder deslocar o cursor em torno do carácter ampliado, embora não possa ainda fazer qualquer alteração ao desenho.

Módulo 3.1.8: Preenchimento a tinta e apagamento

Introduzindo «1» ou «0», a partir do *menu* principal, activa-se ou apaga-se um *pixel* individual do desenho ampliado. Isto é efectuado pelo módulo corrente, através da utilização dos operadores lógicos AND e OR.

Módulo 3.1.8: linhas 18000-19060

```
18000 REM*******************************
18010 REM Ink In Square
18020 REM*******************************
18030 a%(y)=a%(y) OR 2^(7-x)
18040 LOCATE x+1,y+1
18050 PRINT CHR$(231)
18060 RETURN
19000 REM*******************************
19010 REM Blank Out Square
19020 REM*******************************
19030 a%(y)=a%(y) AND 255-2^(7-x)
19040 LOCATE x+1,y+1
19050 PRINT " "
19060 RETURN
```

Comentário

Linha 18030: a utilização de OR é um método comum de passar um, ou mais, dos dígitos binários de qualquer número para 1. O que OR faz é comparar dois números binários, produzindo depois um terceiro em que cada dígito binário é regulado para 1, caso o dígito binário em *qualquer* dos números originais fosse 1. Se, por exemplo, os dois números originais fossem:

124 = 01111100

e

3 = 00000011

o resultado seria 127, ou 01111111, visto que cada dígito binário era 1, em qualquer dos dois números.

Se desejarmos assegurar que um dígito binário específico, número N lido da di-

reita para a esquerda, é regulado para 1, tudo o que é preciso fazer é utilizar OR 2^N.

No caso do programa, a regulação de um dígito binário específico é o mesmo que activar um *pixel,* já que o estado dos *pixels* é determinado pelo estado dos dígitos binários nos oito números que constituem os dados de caracteres.

Linha 19030: o oposto de OR, quando se trata de ligar e desligar dígitos binários individuais, é AND. Quando dois números são alvo da operação lógica AND, o número resultante apenas tem dígitos regulados para 1, nas posições em que *ambos* os números originais as tinham. Assim, a aplicação de AND a 124 e 3 tal como foi descrito em cima produzirá 0, já que nenhum dos dígitos binários está estabelecido em ambos os números. A aplicação de AND a um número entre 0 e 255 com 255 não é relevante, uma vez que, no número 255, todos os oito dígitos estão regulados para 1. No entanto, qualquer um dos dígitos de 255 pode ser desligado subtraindo 2^N a 255, sendo N o número do dígito a ser alterado para 0. Em consequência, a aplicação de AND a um número, X, com 255-2^N altera o dígito binário N para 0, em X.

Em termos do programa, isto equivale ao apagamento de um *pixel.*

Ensaio

Quando em modo de edição de caracteres, você deverá poder deslocar o cursor para qualquer lado, activando ou desactivando *pixels* à vontade.

Módulo 3.1.9: Criação de um carácter em forma inversa

Passamos agora a uma série de três módulos que tornam um pouco mais fácil a tarefa da edição de caracteres, através da execução de operações em todo o carácter, tais como torná-lo na sua imagem reflectida ou rodá-lo até 90 graus. O módulo corrente apenas cria uma inversão do carácter existente. Qualquer *pixel* que estivesse activado será apagado e qualquer posição que estivesse em branco terá um *pixel* activado nesse ponto. Isto consegue-se invertendo os 1 e 0 binários dos *bytes* que constituem o carácter.

Módulo 3.1.9: linhas 15000-15060

```
15000 REM*******************************
15010 REM Invert
15020 REM*******************************
15030 FOR i=0 TO 7
15040 a%(i)=255-a%(i)
15050 NEXT i
15060 ret=1:RETURN
```

Ensaio

Execute o programa e introduza o modo de edição. Quando o *menu* for visualizado, prima «I» e deverá obter um carácter invertido. Prima novamente «I» e o carácter deverá voltar ao normal. Note que isto apenas se refere ao carácter ampliado e

não ao de tamanho normal, à direita do quadrado de 8*8. O carácter de tamanho normal apenas se alterará se você decidir introduzir o seu carácter de edição na memória, utilizando um módulo futuro.

Módulo 3.1.10: Criação de uma imagem reflexa

Este módulo pega no carácter visualizado e «vira-o», como se fosse visto através de um espelho.

Módulo 3.1.10: linhas 16000-16100

```
16000 REM********************************
16010 REM Mirror
16020 REM********************************
16030 FOR i=0 TO 7
16040 b$=RIGHT$(BIN$(256+a%(i)),8)
16050 a%(i)=0
16060 FOR j=0 TO 7
16070 a%(i)=a%(i)+2^j*VAL(MID$(b$,j+1,1)
)
16080 NEXT j
16090 NEXT i
16100 ret=1:RETURN
```

Comentário

Linhas 16060-16080: como cada *byte* do carácter é extraído de A% e transformado numa cadeia binária, acaba por ser lido da esquerda para a direita, como uma cadeia. Assim, quando o valor da variável de ciclo J é 0, o dígito efectivamente a ser lido representa 128. Por isso, se o primeiro carácter de cadeia binária for 1, este será traduzido para 2^0, ou 1 em termos decimais. Se o último carácter da cadeia for 1, então será traduzido para 2^7, ou 128 em termos decimais. Desta forma, o número binário é virado «do inverso».

Ensaio

Execute o programa e chame o *menu* de edição. Prima «M» e, após uma pausa enquanto o carácter é copiado para o quadro, deverá vê-lo impresso em forma reflexa.

Módulo 3.1.11: Viragem do carácter

Se se pensar no carácter a editar como estando impresso numa folha de plástico transparente, então, além de se poder apresentá-lo sob um certo ângulo, tudo o que se pode fazer com essa folha de plástico pode ser efectuado pela combinação do re-

flexo do carácter e/ou da sua viragem em 90 graus, uma ou mais vezes. O módulo corrente volta a utilizar o quadro T%, embora desta vez para virar o carácter no quadrado de 8*8 em 90 graus, no sentido inverso ao dos ponteiros do relógio.

Módulo 3.1.11: linhas 20000-20130

```
20000 REM*******************************
20010 REM Turn
20020 REM*******************************
20030 FOR i=0 TO 7
20040 b$=RIGHT$(BIN$(256+a%(i)),8)
20050 FOR j=0 TO 7
20060 IF i=0 THEN t%(j)=0
20070 t%(j)=t%(j)+2^i*VAL(MID$(b$,j+1,1)
)
20080 NEXT j
20090 NEXT i
20100 FOR i=0 TO 7
20110 a%(i)=t%(i)
20120 NEXT i
20130 ret=1:RETURN
```

Comentário

Linhas 20050-20090: este ciclo copia os dígitos binários de um dos oito números de A% para o quadro temporário T%. Note, no entanto, que enquanto os dígitos binários de cada número de A% são lidos da esquerda para a direita, são copiados para T% de cima para baixo — o primeiro dígito binário de cada número de A% vai para o primeiro elemento de T%, o segundo dígito binário de cada número de A% para o segundo elemento de T% e assim por diante. Desta forma, a coluna vertical do lado esquerdo de dígitos binários do carácter original torna-se na fiada superior (horizontal) de T%, virando assim o carácter 90 graus.

Linhas 20100-20120: o carácter reorganizado é lido novamente a partir de T% para A%, o quadro de trabalho principal.

Ensaio

Execute o programa, chame o *menu* de edição e depois prima «T». Após uma pausa, o carácter será reimpresso e virado 90 graus. Se premir «T» mais três vezes, o carácter deverá ser reposto na sua posição original. Experimente combinações de reflexo, viragem e inversão, até estar familiarizado com os respectivos efeitos.

Módulo 3.1.12: Inserção na memória de um carácter editado

Até aqui, foi-lhe possível manipular o carácter ampliado no quadrado de 8*8 até que, eventualmente, ele deixasse de apresentar semelhanças com o padrão original. No entanto, nada disto teve qualquer influência na versão em tamanho normal do carácter que se encontra impresso à direita do quadrado de 8*8. As alterações por si efectuadas não foram ainda introduzidas na memória de caracteres e não o serão, até você estar satisfeito com aquilo que criou. No entanto, uma vez *chegado* àquilo que pretende, este módulo tornará o padrão dentro do quadrado parte do seu conjunto de caracteres feito por medida.

Módulo 3.1.12: linhas 21000-21070

```
21000 REM*******************************
21010 REM Place In Memory
21020 REM*******************************
21030 FOR i=0 TO 7
21040 c%(ch,i)=a%(i)
21050 NEXT i
21060 SYMBOL ch,a%(0),a%(1),a%(2),a%(3),
a%(4),a%(5),a%(6),a%(7)
21070 ret=1:RETURN
```

Comentário

Linha 21040: o conteúdo de A% (o quadro em que são efectuadas todas as manipulações sobre os dados do carácter) é copiado para a posição em C% correspondente ao carácter corrente.

Linha 21060: os dados em A% são utilizados, juntamente com o poderoso comando SYMBOL, para atribuir novos valores aos oito *bytes* que registam a forma do carácter corrente.

Ensaio

Execute o programa e desloque o apontador de caracteres até ao carácter 65, que é o «A». Passe a modo de edição e vire o carácter uma vez. Prima agora «P» e observe o écran. O «A» à direita do quadrado de 8*8 é modificado, de forma a ficar de lado, pois o *464* foi recolher a informação de caracteres ao seu conjunto feito por medida. É conveniente lembrar, quando se editam caracteres, que, a menos que se pretendam letras de tipo determinado pelo utilizador, é geralmente aconselhável editar apenas caracteres gráficos. Demasiadas alterações a letras e números podem resultar numa situação em que deixa de se perceber o que o programa nos diz!

Módulo 3.1.13: Armazenamento do conjunto de caracteres

Após a edição do conjunto de caracteres, queremos agora poder guardá-lo, para que possa ser utilizado no futuro. Caso contrário, todo este exercício é inútil. Este módulo armazena em fita os caracteres feitos por medida.

Módulo 3.1.13: linhas 22000-22110

```
22000 REM*******************************
22010 REM Save To Tape
22020 REM*******************************
22030 LOCATE 1,24:OPENOUT "char data"
22040 PRINT #9,fc
22050 FOR i=fc TO 255
22060 FOR j=0 TO 7
22070 PRINT #9,c%(i,j)
22080 NEXT j
22090 NEXT i
22100 CLOSEOUT
22110 ret=1:RETURN
```

Comentário

Linhas 22040-22090: para poupar tempo no carregamento e salvaguarda, apenas são salvaguardados os caracteres redefinidos de FC para a frente. Não há vantagem em se esperar enquanto os dados dos 256 caracteres são salvaguardados, se apenas 20, ou algo como isso, foram alterados. Os dados são salvaguardados sob a forma dos números armazenados em C%.

Ensaio

Após a edição de alguns caracteres e sua colocação na memória, chame este módulo para os salvaguardar em fita. A única verificação a fazer neste momento consiste na execução do módulo sem qualquer espécie de erro. Depois da introdução do módulo seguinte, deverá poder voltar a carregar o conjunto de caracteres, para verificar se este foi correctamente armazenado.

Módulo 3.1.14: Recarregamento de um conjunto de caracteres

Após o armazenamento do conjunto de caracteres em fita, este módulo executa a tarefa do recarregamento dos dados dos caracteres. É importante que você compreenda o módulo, já que pode ser utilizado tal como se apresenta para recarregar um conjunto de caracteres para outros programas. Depois da nova concepção do seu conjunto de caracteres e do armazenamento respectivo em fita, tudo o que há a fazer

é incluir este módulo (devidamente renumerado, se necessário) no programa que vai utilizar os novos caracteres. Volte a passar os dados de caracteres e, aí está!, o conjunto de caracteres redesenhado está instalado.

Módulo 3.1.14: linhas 23000-23170

```
23000 REM********************************
23010 REM Load From Tape
23020 REM********************************
23030 LOCATE 1,24
23040 OPENIN "char data"
23050 INPUT #9,fc
23060 FOR i=fc TO 255
23070 FOR j=0 TO 7
23080 INPUT #9,c%(i,j)
23090 NEXT j
23100 NEXT i
23110 CLOSEIN
23120 SYMBOL AFTER fc
23130 FOR i=fc TO 255
23140 SYMBOL i,c%(i,0),c%(i,1),c%(i,2),c
%(i,3),c%(i,4),c%(i,5),c%(i,6),c%(i,7)
23150 NEXT i
23160 done=0:ret=1
23170 RETURN
```

Comentário

Linhas 23040-23110: o carácter de partida é lido a partir da fita e, uma vez que pode não corresponder ao valor corrente, então os dados em fita são lidos novamente para C%, da posição FC para a frente.

Linhas 23120-23150: com os dados novamente em C%, SYMBOL AFTER é utilizado para redifinir o conjunto de caracteres. Normalmente, isto poderia ser feito durante o ciclo de carregamento dos dados, sendo cada carácter redefinido depois de os seus oito *bytes* terem sido recolhidos. Na prática, devido ao que parece ser uma deficiência de funcionamento da ROM do *464,* o comando SYMBOL AFTER aparentemente não é reconhecido enquanto um ficheiro está aberto para o gravador de dados. Valeria a pena verificar se esta deficiência de funcionamento está presente na sua máquina.

Ensaio

Deverá agora poder recarregar o conjunto de caracteres que salvaguardou (SAVE) como parte do ensaio ao módulo anterior, premindo «L» em modo de edição de

caracteres. Antes de recarregar o que acabou de salvaguardar, certifique-se de que regressou ao conjunto de caracteres normal (inédito), ou será impossível dizer se foram carregados diferentes caracteres a partir da fita.

Módulo 3.1.15: Restauração do conjunto de caracteres

É muito possível que você, em qualquer ponto, decida que foi longe de mais nas suas redefinições de caracteres, desejando restaurar o conjunto de caracteres ao seu estado original. Pode fazê-lo inserindo «X», no *menu* principal. A execução é, então, enviada para o módulo de inicialização, para que o carácter de partida seja redefinido. Se desejar reenviar o *464* para a condição de ausência, o primeiro carácter deverá ser 240.

Módulo 3.1.15: linhas 24000-24040

```
24000 REM********************************
24010 REM Normalise Character Set
24020 REM********************************
24030 GOSUB 11000
24040 done=0:ret=1:RETURN
```

Ensaio

Execute o programa e altere um carácter, armazenando a definição em memória. Prima agora «E» e verificará que o carácter foi restaurado ao seu estado original.

Módulo 3.1.16: Finalização do programa

O programa termina, se se premir «E» a partir do *menu* principal, sem restaurar o conjunto de caracteres original.

Módulo 3.1.16: linhas 25000-25030

```
25000 REM********************************
25010 REM End
25020 REM********************************
25030 ret=1:RETURN
```

Ensaio

Altere um carácter, coloque a redefinição em memória e finalize o programa com «X». Quaisquer alterações que tenha feito deverão ainda fazer parte do conjunto de caracteres.

PROGRAMA 3.2: DESENHADOR

Função do programa

Sem dúvida que todos já vimos as impressionantes visualizações criadas por aquilo que se designa «*software* CAD» (*computer-aided design,* ou seja, desenho assistido por computador). Com toques lestos, o engenheiro acrescenta linhas e formas a desenhos complexos, ou apaga os já existentes. De uma forma limitada, o «Desenhador» pretende mimar esse tipo de capacidade. Ainda que não tão sofisticado, obviamente, permitir-lhe-á criar desenhos complexos, que são muito maiores do que um só écran, utilizando o écran de televisão como uma janela móvel, para examinar partes ou encolher toda a área, de forma a examiná-la na sua globalidade. Linhas, círculos e caixas podem ser adicionados à vontade ou apagados, e todo o desenho armazenado em fita para utilização posterior.

Novos conceitos apresentados neste programa incluem:

1) Um cursor intermitente definido pelo utilizador, em alta resolução.
2) Armazenamento de desenhos em fita.
3) Definição e desenho de formas geométricas.

Módulo 3.2.1: Inicialização

Trata-se de um módulo de inicialização padrão, que estabelece cores para o écran, duas janelas e uma gama de variáveis sobre a qual se falará no decurso do comentário ao programa.

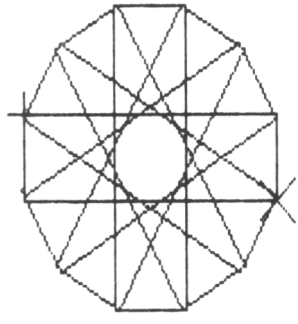


Fig. 3.3 — *Impressão de écran no «Desenhador»*

Módulo 3.2.1: linhas 11000-11250

```
11000 REM*******************************
11010 REM Initialise
11020 REM*******************************
11030 MODE 1
```

```
11040 BORDER 26
11050 INK 0,0
11060 INK 1,26
11070 INK 2,18
11080 INK 3,18
11090 PAPER 1:PEN 0
11100 CLS
11110 WINDOW 27,40,1,25
11120 ORIGIN 200,200,0,398,398,0
11130 CLG
11140 WINDOW #1,27,40,25,25:PAPER #1,1:P
EN #1,0
11150 over$=CHR$(23)+CHR$(1)
11160 normal$=CHR$(23)+CHR$(0)
11170 menu$=""
11180 FOR i=240 TO 247:menu$=menu$+CHR$(
i):NEXT i
11190 menu$=menu$+"12sc1b"+CHR$(13)+"ewt
d\"
11200 unit=1
11210 pix=2
11220 left=-100:right=99
11230 top=99:bottom=-100
11240 DIM a%(1000,4),x(1),y(1)
11250 RETURN
```

Módulo 3.2.2: Dois cursores de desenho

Em qualquer programa de desenho, uma das primeiras necessidades é que o utilizador saiba onde se encontra a posição de desenho corrente. Isto é conseguido, regra geral, através de um qualquer tipo de cursor, que marca a posição corrente no écran. Este programa utiliza dois cursores para definir, por exemplo, os dois extremos de uma linha a ser desenhada, os dois cantos opostos de uma caixa, e assim por diante. Ambos os cursores podem ser deslocados sobre o desenho utilizando as teclas de controlo de cursor, sem que tal afecte o conteúdo do écran.

Módulo 3.2.2: linhas 12000-13080

```
12000 REM*******************************
12010 REM Cursor 1
12020 REM*******************************
12030 PRINT over$
12040 PLOT x(0)*pix-16,y(0)*pix,2
12050 DRAWR 32,0
12060 PLOT x(0)*pix,y(0)*pix-16
```

```
12070 DRAWR 0,32
12080 RETURN
13000 REM*******************************
13010 REM Cursor 2
13020 REM*******************************
13030 PRINT over$
13040 PLOT x(1)*pix-16,y(1)*pix-16,2
13050 DRAWR 32,32
13060 PLOT x(1)*pix-16,y(1)*pix+16
13070 DRAWR 32,-32
13080 RETURN
```

Comentário

Linhas 12030 e 13030: OVER$ foi definido no módulo de inicialização como sendo CHR$(23) mais CHR$(1). CHR$(23) é um carácter de controlo, um carácter que não imprime nada no écran, mas que tem efeito sobre o modo como o *464* funciona.

Neste caso particular, o efeito do carácter é estabelecer o que é conhecido por uma característica «OVER» de desenho no écran, ou, de uma forma mais técnica, estabelecer um XOR (modo eXclusivo OR). Significa isto que, o que quer que se escreva no écran durante o funcionamento da característica OVER, fará passar ao estado oposto quaisquer *pixels* que aquela encontre — se os *pixels* constituíam uma cor de primeiro plano, serão mudados para cor de fundo, e se eram uma cor de fundo, passarão a cor de primeiro plano. A vantagem disto é que, se a mesma forma for desenhada *duas vezes* com OVER a funcionar, o écran ficará exactamente na condição inicial. Por outras palavras, pode ser desenhado um cursor, sendo depois redesenhado para se apagar, sem que isso tenha o mais pequeno efeito a longo prazo sobre tudo o resto que se encontra no écran.

Linhas 12040-12070 e 13040-13070: são desenhados os cursores, sendo um deles uma cruz formada por um traço vertical e horizontal, e o outro uma cruz formada por dois traços em diagonal. A posição da cruz é determinada, em ambos os casos, pelos quadros bielementares X e Y, juntamente com o valor da variável PIX. A razão por que PIX é necessária é que, numa fase posterior do programa, verificará que o desenho pode ser observado em diferentes escalas, pelo que uma distância no desenho pode variar, quando expressa no écran — PIX assegura que o cursor seja correctamente posicionado para a escala corrente de observação.

Módulo 3.2.3: Selecção do cursor

Com dois cursores, obviamente que será necessário poder seleccionar-se qual deles será endereçado. O módulo corrente é chamado a partir do *menu* principal, que introduziremos em breve, através da alteração do valor da variável CN (*Cursor Number,* ou seja, número de cursor).

Módulo 3.2.3: linhas 23000-24040

```
23000 REM*******************************
23010 REM Select Cursor 1
23020 REM*******************************
23030 cn=0
23040 RETURN
24000 REM*******************************
24010 REM Select Cursor 2
24020 REM*******************************
24030 cn=1
24040 RETURN
```

Módulo 3.2.4: Deslocação dos cursores

Finalmente, precisamos poder deslocar os cursores. Tal é conseguido através das teclas cursoras, em que as teclas cursoras não deslocadas (sem *shift*) movem o cursor uma unidade de écran na direcção apropriada e as teclas deslocadas movem o cursor dezasseis unidades. Cada uma das sub-rotinas contidas no módulo é chamada separadamente a partir do *menu,* pela respectiva tecla apropriada. Isto torna o programa mais longo, mas acelera a movimentação do cursor, se comparado com uma sub-rotina cheia de instruções IF para tratar das teclas cursoras.

Módulo 3.2.4: linhas 15000-22040

```
15000 REM*******************************
15010 REM Cursor Up 1
15020 REM*******************************
15030 y(cn)=y(cn)+unit
15040 RETURN
16000 REM*******************************
16010 REM Cursor Down 1
16020 REM*******************************
16030 y(cn)=y(cn)-unit
16040 RETURN
17000 REM*******************************
17010 REM Cursor Left 1
17020 REM*******************************
17030 x(cn)=x(cn)-unit
17040 RETURN
18000 REM*******************************
18010 REM Cursor Right 1
18020 REM*******************************
18030 x(cn)=x(cn)+unit
18040 RETURN
19000 REM*******************************
```

```
19010 REM Cursor Up 16
19020 REM*******************************
19030 y(cn)=y(cn)+unit*16
19040 RETURN
20000 REM*******************************
20010 REM Cursor Down 16
20020 REM*******************************
20030 y(cn)=y(cn)-unit*16
20040 RETURN
21000 REM*******************************
21010 REM Cursor Left 16
21020 REM*******************************
21030 x(cn)=x(cn)-unit*16
21040 RETURN
22000 REM*******************************
22010 REM Cursor Right 16
22020 REM*******************************
22030 x(cn)=x(cn)+unit*16
22040 RETURN
```

Módulo 3.2.5: Introdução de comandos

Estando dotados de dois cursores, bem como da capacidade para os movimentar, viramo-nos para o módulo que confere trabalho entre as várias partes do programa, ao ser premida uma tecla. O seu correcto funcionamento apenas pode ser ensaiado quando lhe for adicionado o módulo subsequente, o pequeno módulo de controlo. A maior parte dos comandos únicos mencionados no comentário não se tornarão operativos, obviamente, até que módulos posteriores lhes sejam acrescentados.

Módulo 3.2.5: linhas 14000-14440

```
14000 REM*******************************
14010 REM Edit Mode
14020 REM*******************************
14030 CLS
14040 PRINT "   EDIT MODE":PRINT
14050 PRINT "X1:"
14060 PRINT "Y1:":PRINT
14070 PRINT "X2:"
14080 PRINT "Y2:":PRINT
14090 PRINT "Scale:";unit
14100 PRINT "Items:";item:PRINT
14110 PRINT "'1' Cursor 1"
14120 PRINT "'2' Cursor 2"
```

```
14130 PRINT "'S' Shape"
14140 PRINT "'C' Circle"
14150 PRINT "'L' Line"
14160 PRINT "'B' Box"
14170 PRINT "'";CHR$(1);CHR$(13);"' Fix"
14180 PRINT "'E' Erase"
14190 PRINT "'W' Window"
14200 PRINT "'T' Save"
14210 PRINT "'D' Del Mode"
14220 PRINT "'\' End"
14230 t=0:WHILE t<19
14240 GOSUB 12000
14250 GOSUB 13000
14260 LOCATE 4,3:PRINT x(0);TAB(14)
14270 LOCATE 4,4:PRINT y(0);TAB(14)
14280 LOCATE 4,6:PRINT x(1);TAB(14)
14290 LOCATE 4,7:PRINT y(1);TAB(14)
14300 t=0:WHILE t=0
14310 t$="":WHILE t$=""
14320 t$=LOWER$(INKEY$)
14330 WEND
14340 t=INSTR(menu$,t$)
14350 WEND
14360 GOSUB 12000
14370 GOSUB 13000
14380 ON t GOSUB 15000,16000,17000,18000
,19000,20000,21000,22000,23000,24000,250
00,26000,27000,28000,29000,30000,31000,3
2000
14390 IF x(cn)>right THEN x(cn)=right
14400 IF x(cn)<left THEN x(cn)=left
14410 IF y(cn)>top THEN y(cn)=top
14420 IF y(cn)<bottom THEN y(cn)=bottom
14430 WEND
14440 RETURN
```

Comentário

Linhas 14030 e 14290: o *menu* do programa, que, devido à forma como o écran foi estabelecido na inicialização, é impresso na parte direita do écran — a esquerda e centro do écran são reservados para o próprio desenho. Para além do *menu,* os cursores são desenhados sobre o desenho principal e a sua localização listada.

Linhas 14300-14350: estes ciclos encaixados esperam que seja premida uma tecla, comparando-a depois com o MENU$ estabelecido no módulo de inicialização e produzindo um valor para o comando ON... GOSUB, na linha 14380.

Linhas 14390-14420: se, como resultado de um comando de movimentação do cursor, este se tenha deslocado para além dos limites da janela corrente sobre o desenho, registada nas variáveis BOTTOM, TOP, LEFT e RIGHT, as coordenadas do cursor são corrigidas por estas linhas.

Módulo 3.2.6: O módulo de controlo

O pequeno módulo de controlo deve ser introduzido nesta altura, para que a função de movimentação do cursor possa ser ensaiada, bem como módulos posteriores, à medida que forem entrando. Note que o módulo tem em consideração a recordação de dados a partir da fita — isto será amplamente discutido quando os módulos relevantes forem introduzidos.

Módulo 3.2.6: linhas 10000-10100

```
10000 REM********************************
10010 REM Control
10020 REM********************************
10030 INPUT "Load from tape (y/n)";q$
10040 GOSUB 11000
10050 IF LOWER$(q$)="y" THEN GOSUB 33000
10060 WHILE 1
10070 GOSUB 14000
10080 IF t=20 THEN MODE 1:CLS:END
10090 GOSUB 40000
10100 WEND
```

Ensaio

Execute o programa e deverá verificar que consegue movimentar qualquer dos cursores pelo écran através das teclas de controlo dos cursores. Nenhuma das outras funções do programa estarão ainda disponíveis.

Módulo 3.2.7: Desenho de uma linha

Segue-se agora uma série de três módulos, que fazem o trabalho pesado do programa desenhando, respectivamente, linhas, caixas e polígonos (incluindo círculos). Não poderá utilizá-los para isto imediatamente, visto que estes módulos não são chamados directamente do *menu,* mas por outros módulos que têm a tarefa de dirigir o trabalho de desenho/apagamento, e assim por diante. O módulo corrente apenas desenha uma linha a partir da posição do cursor 2 até à posição do cursor 1.

Módulo 3.2.7: linhas 37000-37050

```
37000 REM*******************************
37010 REM Draw Line
37020 REM*******************************
37030 PLOT x2*pix,y2*pix,c
37040 DRAW x1*pix,y1*pix
37050 RETURN
```

Ensaio

Execute o programa para montar o écran e depois pare, com « \ ». Escreva:

C = 2:X1 = 50:Y1 = 50:GOTO 37000[ENTER]

e deverá ver uma linha desenhada desde o centro da zona negra, em diagonal, para cima e para a direita.

Módulo 3.2.8: Desenho de um círculo ou polígono

Trata-se de um módulo de aplicação geral, que desenha entre pontos em torno de um círculo. O esquema do desenho aparecerá, ou como um círculo, ou como um polígono regular, dependendo do número de lados especificado — se forem usados três pontos, a forma será a de um triângulo.

Módulo 3.2.8: linhas 38000-38140

```
38000 REM*******************************
38010 REM Draw Polygon
38020 REM*******************************
38030 r=SQR((x1-x2)*(x1-x2)+(y1-y2)*(y1-
y2))
38040 z2=CINT(8*LOG(r/unit))
38050 IF NOT(z1=2 OR z1>z2) THEN z2=z1
38060 IF x1=x2 THEN a=PI/2*SGN(y1-y2)
38070 IF x1<>x2 THEN a=ATN((y1-y2)/(x1-x
2))
38080 IF x1<x2 THEN a=a+PI
38090 PLOT (x2+r*COS(a))*pix,(y2+r*SIN(a
))*pix,c
38100 FOR s=1 TO z2
38110 a1=a+2*PI*s/z2
38120 DRAW (x2+r*COS(a1))*pix,(y2+r*SIN(
a1))*pix
38130 NEXT s
38140 RETURN
```

Comentário

Linha 38030: é a distância em diagonal entre os dois cursores.

Linhas 38040-38050: Z2 será o número de pontos a traçar *(plot)* em torno da estrutura circular. A expressão da linha 38040 fornece pontos suficientes para dar um círculo aceitável — não há vantagem em se ir demasiado longe na definição do número de pontos a utilizar, já que a melhoria é dificilmente notada e o processo pode tornar-se tortuosamente lento. Quando o utilizador tiver especificado outra forma que não um círculo, a linha 38050 estabelece o número de pontos de acordo com os especificados pelo utilizador, desde que este número não seja maior do que o necessário para desenhar um círculo.

Linhas 38060-38080: estas três linhas calculam o ângulo do cursor 1, em relação ao do cursor 2.

Linha 38090: o primeiro ponto em torno do círculo, que deverá ser tudo menos idêntico à posição do cursor 2. Deverá reconhecer a equação do programa «Anarelógio», no primeiro capítulo.

Linhas 38100-38130: este ciclo calcula a posição de Z2 pontos em torno da circunferência do círculo, desenhando de um para outro como calculado. Uma vez mais, a fórmula é a mesma que foi explicada no programa «Anarelógio».

Ensaio

Tantas variáveis têm de ser estabelecidas para ensaiar este módulo e o seguinte, que é melhor esperar até que tenham sido acrescentados mais módulos, de modo a incluí-los na sequência principal do programa.

Módulo 3.2.9: Desenho de uma caixa

A forma final que o programa é capaz de desenhar é um simples rectângulo, cujos cantos opostos se encontram nos pontos 1 e 2. Este módulo é ligeiramente mais complexo do que os anteriores, já que não tem qualquer comando incluído para a forma e, para além disso, incluímos uma provisão para que ela seja rodada.

Módulo 3.2.9: linhas 36000-36210

```
36000 REM********************************
36010 REM Draw Box
36020 REM********************************
36030 a=-z1/10000
36040 cs=COS(a)
36050 sn=SIN(a)
```

```
36060 x=(x1+x2)/2
36070 y=(y1+y2)/2
36080 rx1=x+(x1-x)*cs+(y1-y)*sn
36090 ry1=y+(y1-y)*cs-(x1-x)*sn
36100 rx2=x+(x2-x)*cs+(y1-y)*sn
36110 ry2=y+(y1-y)*cs-(x2-x)*sn
36120 rx3=x+(x2-x)*cs+(y2-y)*sn
36130 ry3=y+(y2-y)*cs-(x2-x)*sn
36140 rx4=x+(x1-x)*cs+(y2-y)*sn
36150 ry4=y+(y2-y)*cs-(x1-x)*sn
36160 PLOT rx1*pix,ry1*pix,c
36170 DRAW rx2*pix,ry2*pix
36180 DRAW rx3*pix,ry3*pix
36190 DRAW rx4*pix,ry4*pix
36200 DRAW rx1*pix,ry1*pix
36210 RETURN
```

Comentário

Linhas 36030-36050: a variável A contém o ângulo, fornecido pelo utilizador, segundo o qual a caixa deve ser rodada. CS e SN serão utilizados para encurtar as fórmulas do movimento dos cantos da caixa ao ser rodada.

Linhas 36080-36150: as fórmulas para a rotatividade dos pontos são as seguintes:

X2 = XO + XD * COS ANGLE + YD * SIN ANGLE
Y2 = YO + YD * COS ANGLE − XD * SIN ANGLE

X2 e Y2 são as coordenadas X e Y que resultam da rotação do ponto X, Y.
X0 e Y0 são as coordenadas em torno das quais se dá a rotação do ponto.
XD e YD são as distâncias de X e Y em relação a X0 e Y0, respectivamente.
ANGLE é o ângulo segundo o qual é rodado o ponto definido por X e Y.

O rectângulo básico, por rodar, terá cantos de X1/Y1, X2/Y1, X2/Y2 e X1/Y2. Observando as linhas e comparando-as com as fórmulas acima, deverá ver que elas fornecem as coordenadas rodadas para os quatro pontos angulares.

Módulo 3.2.10: Atribuição de trabalho entre os módulos de desenho

Este curto módulo é um dos necessários à inclusão dos módulos de desenho no que se segue. A sua utilidade será explicada quando observarmos o próximo módulo.

PUBLICAÇÕES EUROPA-AMÉRICA
Apartado 8
2726 MEM MARTINS CODEX

IMPRESSO — RESPOSTA
Autorizado pelos CTT

A utilizar somente no Continente e Regiões Autónomas

RSF

Não carece de selo.
O porte será pago pelo destinatário.

****

****

n a única tecla premida a partir
á utilização dos verdadeiros mó-

s:

tais como «quantos lados» ou

rimindo OVER$.
ão tenha sido confirmada pelo
ualdade da variável TEMP a 1.
e Y2, as posições dos cursores.
tes.
mostrar que foi desenhada uma

****

```
25020 REM*********************************
25030 z=0:e=0:WHILE z<3 OR z>32767
25040 IF e=1 THEN x$="*OUT OF RANGE*":GO
SUB 34000
25050 INPUT #1,"Sides";z:PRINT #1
25060 e=1:WEND
25070 PRINT over$:c=2
25080 IF temp=1 THEN GOSUB 35000
25090 z1=z:x1=x(0):y1=y(0):x2=x(1):y2=y(
1)
25100 GOSUB 38000
25110 temp=1
25120 RETURN
26000 REM*********************************
26010 REM Circle
```

```
36060 x=(x1+x2)/2
36070 y=(y1+y2)/2
36080 rx1=x+(x1-x)
36090 ry1=y+(y1-y)
36100 rx2=x+(x2-x)
36110 ry2=y+(y1-y)
36120 rx3=x+(x2-x)
36130 ry3=y+(y2-y)
36140 rx4=x+(x1-x)
36150 ry4=y+(y2-y)
36160 PLOT rx1*pix
36170 DRAW rx2*pix
36180 DRAW rx3*pix
36190 DRAW rx4*pix
36200 DRAW rx1*pix
36210 RETURN
```

Comentário

Linhas 36030-36050: a va
segundo o qual a caixa deve s
fórmulas do movimento dos c

Linhas 36080-36150: as fó
tes:

X2 = XO + XD * COS ANGL
Y2 = YO + YD * COS ANGL

X2 e Y2 são as coordenadas X e Y que resultam da rotação do ponto X, Y.
X0 e Y0 são as coordenadas em torno das quais se dá a rotação do ponto.
XD e YD são as distâncias de X e Y em relação a X0 e Y0, respectivamente.
ANGLE é o ângulo segundo o qual é rodado o ponto definido por X e Y.

O rectângulo básico, por rodar, terá cantos de X1/Y1, X2/Y1, X2/Y2 e X1/Y2. Observando as linhas e comparando-as com as fórmulas acima, deverá ver que elas fornecem as coordenadas rodadas para os quatro pontos angulares.

Módulo 3.2.10: Atribuição de trabalho entre os módulos de desenho

Este curto módulo é um dos necessários à inclusão dos módulos de desenho no que se segue. A sua utilidade será explicada quando observarmos o próximo módulo.

As Publicações e as Livrarias
EUROPA-AMÉRICA

Convidam-no a juntar-se
àquele grupo de leitores exigentes
para quem um livro,
além de uma companhia agradável,
é uma companhia útil.
Assim, teremos muito gosto
em passar a enviar-lhe, regularmente,
informações sobre os livros
que publicamos.

o Editor

Nome: _____
Profissão: _____
Morada: _____ Tel.: _____
Cód. Postal: _____ Localidade: _____
Encontrei este postal no livro: _____

☐ Que adquiri numa livraria ☐ Que encomendei pelo correio

NÓS EDITAMOS O LIVRO QUE VOCÊ PROCURA!

IMPORTANTE: Se desejar alguns livros de nossa edição, também poderá usar este postal. Bastará preencher o espaço que se segue. Todas as encomendas são confidenciais.

Referência Título e Autor

Módulo 3.2.10: linhas 35000-35060

```
35000 REM*******************************
35010 REM Global Draw
35020 REM*******************************
35030 IF z1<1 THEN GOSUB 36000
35040 IF z1=1 THEN GOSUB 37000
35050 IF z1>1 THEN GOSUB 38000
35060 RETURN
```

Módulo 3.2.11: Controlo dos comandos de desenho

Chegamos agora às três sub-rotinas que traduzem a única tecla premida a partir do módulo do *menu* para os parâmetros necessários à utilização dos verdadeiros módulos de desenho.

Cada um dos módulos tem as seguintes funções:

1) Pedir, se necessário, mais informações, tais como «quantos lados» ou «grau de rotação».
2) Estabelecer a característica OVER, imprimindo OVER$.
3) Apagar qualquer forma existente que não tenha sido confirmada pelo utilizador, tal como está indicado na igualdade da variável TEMP a 1.
4) Estabelecer as variáveis Z1, X1, Y1, X2 e Y2, as posições dos cursores.
5) Chamar os módulos de desenho relevantes.
6) Estabelecer a variável TEMP de forma a mostrar que foi desenhada uma forma ainda não confirmada.

Módulo 3.2.11: linhas 25000-28110

```
25000 REM*******************************
25010 REM Shape
25020 REM*******************************
25030 z=0:e=0:WHILE z<3 OR z>32767
25040 IF e=1 THEN x$="*OUT OF RANGE*":GO
SUB 34000
25050 INPUT #1,"Sides";z:PRINT #1
25060 e=1:WEND
25070 PRINT over$:c=2
25080 IF temp=1 THEN GOSUB 35000
25090 z1=z:x1=x(0):y1=y(0):x2=x(1):y2=y(
1)
25100 GOSUB 38000
25110 temp=1
25120 RETURN
26000 REM*******************************
26010 REM Circle
```

```
26020 REM*********************************
26030 PRINT over$:c=2
26040 IF temp=1 THEN GOSUB 35000
26050 z1=2:x1=x(0):y1=y(0):x2=x(1):y2=y(
1)
26060 GOSUB 38000
26070 temp=1
26080 RETURN
27000 REM*********************************
27010 REM Line
27020 REM*********************************
27030 PRINT over$:c=2
27040 IF temp=1 THEN GOSUB 35000
27050 z1=1:x1=x(0):y1=y(0):x2=x(1):y2=y(
1)
27060 GOSUB 37000
27070 temp=1
27080 RETURN
28000 REM*********************************
28010 REM Box
28020 REM*********************************
28030 INPUT #1,"Rotation";a:PRINT #1
28040 a=a-180*INT(a/180)
28050 z=-CINT(a/180*PI*10000)
28060 PRINT over$:c=2
28070 IF temp=1 THEN GOSUB 35000
28080 z1=z:x1=x(0):y1=y(0):x2=x(1):y2=y(
1)
28090 GOSUB 36000
28100 temp=1
28110 RETURN
```

Comentário

Em vez de um comentário a cada sub-rotina, serão analisados com brevidade os pontos principais de uma delas, pontos esses que se aplicam a todas as sub-rotinas.

Linhas 25030-25060: é a informação suplementar requerida, se tiver de ser desenhada uma forma, ou seja, a quantidade de lados.

Linha 25080: se a variável TEMP estiver estabelecida, então acabou de ser desenhada uma forma e não foi confirmada (abordaremos a confirmação dentro em pouco). Antes de desenhar a forma corrente, o módulo da linha 35000 é utilizado para voltar a chamar o módulo de desenho relevante. Uma vez que OVER se encontra

estabelecido, o módulo desenhará sobre a forma anterior, apagando-a. Significa isto que, se o utilizador pretende desenhar uma forma num desenho complexo e ela não está perfeitamente certa (não se enquadra devidamente), um ou outro dos cursores pode ser deslocado, utilizando a forma insatisfatória como guia, e desenhada a forma novamente, apagando automaticamente a primeira versão.

Linha 25090: são estabelecidas as variáveis.

Ensaio

Deverá agora poder executar o programa e desenhar formas, embora não possa ainda confirmá-las nem torná-las parte integrante do desenho — cada forma desenhada apagará a anterior.

Módulo 3.2.12: Gravação de um desenho

Um programa como este é de pouca utilidade para mais do que alguns momentos passageiros de divertimento, a menos que o desenho em que se trabalhe possa ser gravado, de alguma forma. No nosso caso, a gravação do desenho num quadro permitir-nos-á, mais tarde, tanto o armazenamento dos dados em fita, como o apagamento de linhas individuais. Linhas e formas são gravadas chamando este módulo, à medida que vão sendo introduzidas. Para além disto, o módulo desenha a forma em permanência, de modo que ela não será apagada quando a forma seguinte for desenhada. O módulo é chamado premindo ENTER no *menu* principal.

Módulo 3.2.12: linhas 29000-29150

```
29000 REM*********************************
29010 REM Fix
29020 REM*********************************
29030 IF temp=0 THEN RETURN
29040 IF item=1001 THEN x$="*NO MORE ROO
M*":GOSUB 34000:RETURN
29050 PRINT normal$:c=1
29060 GOSUB 35000
29070 a%(item,0)=z1
29080 a%(item,1)=x1
29090 a%(item,2)=y1
29100 a%(item,3)=x2
29110 a%(item,4)=y2
29120 temp=0
29130 item=item+1
29140 LOCATE 7,10:PRINT item;TAB(14)
29150 RETURN
```

Comentário

Linha 29030: se TEMP não for igual a 1, não haverá forma para registar.

Linha 29050-29060: NORMAL$, que foi definido no módulo de inicialização, desliga OVER, de modo que tudo o que for desenhado, sê-lo-á de forma normal. O módulo da linha 35000 é então chamado para escolher entre os vários módulos de desenho.

Linhas 29070-29130: as variáveis necessárias ao desenho da forma são armazenadas no quadro A%. TEMP é colocado em 0, para indicar que não existe no écran qualquer forma por confirmar, e a variável ITEM é incrementada, para mostrar que foi acrescentada outra forma ao desenho.

Módulo 3.2.13: Apagamento de uma forma

Trata-se de um módulo simples, para apagamento de uma forma não confirmada, do mesmo modo como quando é desenhada uma nova forma.

Módulo 3.2.13: linhas 30000-30070

```
30000 REM*******************************
30010 REM Erase
30020 REM*******************************
30030 IF temp=0 THEN RETURN
30040 PRINT over$:c=2
30050 GOSUB 35000
30060 temp=0
30070 RETURN
```

Ensaio

Deverá agora poder apagar uma forma não confirmada premindo «E», no *menu*.

Módulo 3.2.14: Reimpressão de um desenho

Depois de nos termos proporcionado a possibilidade de registar uma forma num quadro, deparamos agora com a questão da reimpressão da totalidade do desenho a partir dos dados armazenados no quadro, colocando-os novamente no écran.

Módulo 3.2.14: linhas 39000-39130

```
39000 REM*******************************
39010 REM Redraw
39020 REM*******************************
39030 IF item=0 THEN RETURN
```

```
39040 PRINT normal$:c=1
39050 FOR i=0 TO item-1
39060 z1=a%(i,0)
39070 x1=a%(i,1)
39080 y1=a%(i,2)
39090 x2=a%(i,3)
39100 y2=a%(i,4)
39110 GOSUB 35000
39120 NEXT i
39130 RETURN
```

Ensaio

Execute o programa e desenhe algumas formas, parando depois o programa com « \ » e escrevendo:

CLS #2[ENTER]
GOTO 39000[ENTER]

Deverá ver todas as formas reimpressas no écran.

Módulo 3.2.15: Apagamento de linhas e formas

Uma vez que o desenho não é armazenado como um todo, mas antes como linhas e formas individuais, também é simples dar ao utilizador a opção de apagar itens individuais. Este módulo visualiza todo o desenho, linha por linha, com a opção de apagar qualquer linha. Pode também ser usado para redesenhar o desenho, caso o écran tenha ficado limpo devido à paragem do programa.

Módulo 3.2.15: linhas 40000-40290

```
40000 REM*******************************
40010 REM Delete Mode
40020 REM*******************************
40030 CLS
40040 CLG
40050 PRINT "  DELETE MODE":PRINT
40060 PRINT "Item # 1":PRINT
40070 PRINT "Scale:";unit
40080 PRINT "Items:";item:PRINT
40090 PRINT "'";CHR$(1);CHR$(13);"' Next
 Item"
40100 PRINT "'E' Erase"
40110 PRINT "'\' Return"
40120 t$="":i=0:WHILE i<item
```

```
40130 z1=a%(i,0)
40140 x1=a%(i,1)
40150 y1=a%(i,2)
40160 x2=a%(i,3)
40170 y2=a%(i,4)
40180 d=0:WHILE t$<>"\" AND d=0
40190 PRINT over$:c=2
40200 GOSUB 35000
40210 t$=""
40220 d=1:WEND
40230 WHILE t$=""
40240 t$=UPPER$(INKEY$)
40250 WEND
40260 IF t$="E" THEN GOSUB 41000 ELSE GO
SUB 42000
40270 WEND
40280 temp=0
40290 RETURN
```

Comentário

Linhas 40120-40220: os pormenores para uma forma são retirados do quadro A% e desenhados com OVER activado.

Linhas 40230-40280: o programa espera que o utilizador introduza «E», para apagar a forma visualizada, ou qualquer outra tecla para a confirmar. Nada disto terá ainda efeito, já que é necessário introduzir os dois módulos seguintes.

Módulo 3.2.16: Apagamento de uma forma do quadro alfanumérico

Após a reimpressão de uma forma a partir do quadro alfanumérico, o seu apagamento definitivo apenas requer que ela seja redesenhada com OVER, sendo o quadro eliminado, para remover o registo dos pormenores.

Módulo 3.2.16: linhas 41000-41120

```
41000 REM*******************************
41010 REM Erase
41020 REM*******************************
41030 PRINT over$:c=2
41040 GOSUB 35000
41050 item=item-1
41060 FOR j=i TO item-1
41070 FOR k=0 TO 4
```

```
41080 a%(j,k)=a%(j+1,k)
41090 NEXT k
41100 NEXT j
41110 LOCATE 7,6:PRINT item;TAB(14)
41120 RETURN
```

Módulo 3.2.17: Confirmação de um item em modo de apagamento

Para confirmar um item, tudo o que é preciso fazer é voltar a desenhá-lo com OVER desligado.

Módulo 3.2.17: linhas 42000-42070

```
42000 REM*********************************
42010 REM Next Item
42020 REM*********************************
42030 PRINT normal$:c=1
42040 GOSUB 35000
42050 i=i+1
42060 LOCATE 7,3:PRINT i+1
42070 RETURN
```

Ensaio

Desde que você tenha introduzido algumas formas, deverá agora premir «D», a partir da parte principal do programa, para obter este módulo. Confirme que está em condições de folhear o desenho que introduziu, apagando itens ou deixando-os intactos. Se premir « \ » no início do desenho, tal não deverá ter outras consequências que não sejam a reimpressão de todo o desenho.

Módulo 3.2.18: Mensagens de erro

Trata-se de um módulo simples, para impedir uma mensagem de erro especificada por outra parte do programa.

Módulo 3.2.18: linhas 34000-34070

```
34000 REM*********************************
34010 REM Error
34020 REM*********************************
34030 PRINT #1,x$;
34040 SOUND 1,1000,100
34050 FOR i=1 TO 1500:NEXT i
34060 PRINT #1
34070 RETURN
```

Módulo 3.2.19: Janelas e escalas

Até agora, não tocámos, nem ao de leve, numa das mais úteis capacidades do «Desenhador» — a de deslocar o écran como uma janela sobre um vasto desenho, ou a de encolhê-lo de modo que possa ser visto num único écran. Este módulo é bastante complexo, mas seria ainda pior se não fosse o facto de o *464* tornar tão fácil a deslocação da origem (ORIGIN) dos gráficos e o desenho de linhas que não aparecem no écran sem gerarem um erro.

Módulo 3.2.19: linhas 31000-32280

```
31000 REM*********************************
31010 REM Window
31020 REM*********************************
31030 x$="*OUT OF RANGE*"
31040 e=0:WHILE e=0 OR x<-16384 OR x>163
83
31050 IF e=1 THEN GOSUB 34000
31060 INPUT #1,"X";x
31070 e=1:WEND
31080 e=0:WHILE e=0 OR y<-16384 OR y>163
83
31090 IF e=1 THEN GOSUB 34000
31100 INPUT #1,"Y";y
31110 e=1:WEND
31120 e=0:WHILE e=0 OR unit<1
31130 IF e=1 THEN GOSUB 34000
31140 INPUT #1,"Scale":unit:PRINT #1
31150 e=1:WEND
31160 pix=2/unit
31170 LOCATE 7,9:PRINT unit;TAB(14)
31180 left=x-100*unit:IF left<-16384 THE
N left=-16384
31190 right=x+99*unit:IF right>16383 THE
N right=16383
31200 top=y+99*unit:IF top>16383 THEN to
p=16383
31210 bottom=y-100*unit:IF bottom<-16384
 THEN bottom=-16384
31220 x(0)=x:y(0)=y
31230 x(1)=x:y(1)=y
31240 temp=0
31250 ORIGIN 200-x*pix,200-y*pix,0,398,3
98,0
31260 CLG
```

```
31270 GOSUB 39000
31280 RETURN
```

Comentário

Linhas 31030 e 31050: são uma chamada da rotina de mensagem de erro acabada de introduzir. O módulo contém várias chamadas, para avisar o utilizador quando estão a ser introduzidos valores inválidos.

Linhas 31060-31110: o utilizador é convidado a introduzir as coordenadas X e Y do centro da nova janela. A dimensão total do desenho é de 32768*32768 unidades e a janela pode ser deslocada para qualquer parte desta área teórica.

Linhas 31120-31150: é introduzida a escala da janela. Quanto mais alto for o valor introduzido, menor será o desenho, quando colocado no écran. Isto permite a criação de um desenho numa escala que é muito maior do que o écran e a visualização do referido desenho numa escala reduzida, para que tudo possa ser visto de uma vez só.

Linhas 31180-31210: são estabelecidas as variáveis que representam os limites do écran. Se algum dos limites estiver para além da dimensão máxima para o desenho, a janela será deslocada até que os seus limites estejam dentro do permitido.

Linhas 31220-31270: ambos os cursores são deslocados para o centro da nova janela e a origem dos gráficos reposta a zero. O écran de gráficos é limpo e, então, é chamado o módulo de reimpressão, para recriar o desenho. É muito possível que, com a janela deslocada, alguma parte ou todo o desenho fique fora da nova janela e, desta forma, não esteja visível.

Ensaio

Com um desenho introduzido, deve agora poder deslocar o écran sobre o desenho, ampliando-o ou encolhendo-o.

Módulo 3.2.20: Armazenamento e carregamento em fita

Trata-se de uma secção padrão de armazenamento de dados. Para uma explicação dos métodos, veja o programa «Gráfico 3D», no capítulo anterior.

Módulo 3.2.20: linhas 32000-33130

```
32000 REM*******************************
32010 REM Save To Tape
32020 REM*******************************
32030 INPUT #1,"Name:",n$:PRINT #1
32040 OPENOUT "!"+n$
```

```
32050 PRINT #9,item
32060 FOR i=0 TO item-1
32070 FOR j=0 TO 4
32080 PRINT #9,a%(i,j)
32090 NEXT j
32100 NEXT i
32110 CLOSEOUT
32120 RETURN
33000 REM*******************************
33010 REM Load From Tape
33020 REM*******************************
33030 INPUT #1,"Name:",n$:PRINT #1
33040 OPENIN "!"+n$
33050 INPUT #9,item
33060 FOR i=0 TO item-1
33070 FOR j=0 TO 4
33080 INPUT #9,a%(i,j)
33090 NEXT j
33100 NEXT i
33110 CLOSEIN
33120 GOSUB 39000
33130 RETURN
```

Ensaio

Crie um desenho simples e utilize «T» para o armazenar. Pare o programa e depois execute-o novamente, para limpar a memória. Responda «Y» à solicitação que lhe pergunta se deseja carregar a partir da fita (TAPE). Dê o nome do ficheiro sob o qual o desenho foi armazenado e deverá ver o desenho recriado no écran.

Se este ensaio for bem sucedido, o programa estará pronto para ser utilizado.

PROGRAMA 3.3: MÚSICA

Função do programa

Além das suas muitas outras capacidades, o *464* é, sem dúvida, um dos micros musicais mais potentes da geração corrente, com uma gama de úteis características com que os possuidores de outros micros apenas podem sonhar. Não se trata apenas do facto de, contrariamente a algumas outras máquinas cujas características são igualmente poderosas, o BASIC do *464* estar concebido para permitir a qualquer pessoa, com um mínimo de esforço, retirar o máximo do poder musical fornecido pelo *hardware.*

O corrente programa é um bom exemplo do que pode ser conseguido com alguma reflexão. Ao utilizar o programa, você estará apto para organizar melodias e har-

monias complexas, manipulando-as de variadas formas. Quer você esteja musicalmente treinado ou seja apenas um curioso, os resultados que o programa porporciona surpreendê-lo-ão e, esperamos, farão as suas delícias.

```
CHANGE PARAMETERS
=================

ENTER to leave parameter unchanged

Relative octave ( 0 ) :1
Relative note ( 0 ) :3
Unit length ( 20 ) :15
Play channel 0 (Y) :
Play channel 1 (Y) :
Play channel 2 (Y) :N
```

Fig. 3.4 — *Um «menu» de «Música»*

Módulo 3.3.1: Os dados para a melodia

Tal como acontecia com os programas de gráficos vistos anteriormente, o método adoptado no programa é o de colocar a informação em que se baseará a melodia numa secção de instruções de dados, no final do programa, permitindo assim uma muito maior facilidade de examinação e edição. Neste ponto, não é importante que compreenda os dados, já que os seus vários aspectos serão explicados, à medida que formos avançando nas partes do programa que os utilizam. Os dados específicos aqui apresentados executam uma versão muito respeitável de «*Für Elise*».

Módulo 3.3.1: linhas 26000-28390

```
26000 REM ******************************
26010 REM Channel 0 Data
26020 REM ******************************
26030 DATA ENV1
26040 DATA *
26050 DATA L1,16,15
26060 DATA 16,15,16,11,14,12
26070 DATA L2,9,L1,R,0,4,9
26080 DATA L2,11,L1,R,4,8,11
26090 DATA L2,12,L1,R,4,16,15
26100 DATA 16,15,16,11,14,12
26110 DATA L2,9,L1,R,0,4,9
26120 DATA L2,11,L1,R,4,8,11
26130 DATA [1
26140 DATA L4,9
26150 DATA ]
26160 DATA *2
```

```
26170 DATA [2
26180 DATA L2,9,L1,R,11,12,14
26190 DATA *
26200 DATA L3,16,L1,7,17,16
26210 DATA L3,14,L1,5,16,14
26220 DATA L3,12,L1,4,14,12
26230 DATA L2,11,L1,R,4,16,R
26240 DATA R,16,28,R2,15
26250 DATA 16,R2,15,16,15
26260 DATA 16,15,16,11,14,12
26270 DATA L2,9,L1,R,0,4,9
26280 DATA L2,11,L1,R,4,8,11
26290 DATA L2,12,L1,R,4,16,15
26300 DATA 16,15,16,11,14,12
26310 DATA L2,9,L1,R,0,4,9
26320 DATA L2,11,L1,R,4,12,11
26330 DATA [1
26340 DATA L2,9,L1,R,11,12,14
26350 DATA ]
26360 DATA *2
26370 DATA L2,9,L1,R,0,4,9
26380 DATA 12,16,L4,21
26390 DATA end
27000 REM ******************************
27010 REM Channel 1 Data
27020 REM ******************************
27030 DATA end
28000 REM ******************************
28010 REM Channel 2 Data
28020 REM ******************************
28030 DATA ENV1,0-2
28040 DATA *
28050 DATA R2
28060 DATA R6
28070 DATA 9,16,21,R3
28080 DATA 4,16,20,R3
28090 DATA 9,16,21,R3
28100 DATA R6
28110 DATA 9,16,21,R3
28120 DATA 4,16,20,R3
28130 DATA [1
28140 DATA 9,16,21,R1
28150 DATA ]
28160 DATA *2
```

```
28170 DATA [2
28180 DATA 9,16,21,R3
28190 DATA *
28200 DATA 12,19,24,R3
28210 DATA 7,19,23,R3
28220 DATA 9,16,21,R3
28230 DATA 4,16,28,R2,00,4
28240 DATA 16,R,R,15,16,R
28250 DATA R,15,16,R3
28260 DATA R6,0-2
28270 DATA 9,16,21,R3
28280 DATA 4,16,20,R3
28290 DATA 9,16,21,R3
28300 DATA R6
28310 DATA 9,16,21,R3
28320 DATA 4,16,20,R3
28330 DATA [1
28340 DATA 9,16,21,R3
28350 DATA ]
28360 DATA *2
28370 DATA [2
28380 DATA 9,16,21,R3
28390 DATA end
```

Módulo 3.3.2: Inicialização

Trata-se de um módulo mais complexo do que é habitual, devido à necessidade de recolher e processar os dados da melodia antes de passar à parte do programa que é capaz de executar a própria melodia.

Módulo 3.3.2: linhas 10000-10160

```
10000 REM ******************************
10010 REM Initialise
10020 REM ******************************
10030 CLS:LOCATE 15,13:PRINT"Please Wait
"
10040 DIM a$(200,2),p$(2),p(2),d(2),s%(2
,500,4)
10050 FOR c=0 TO 2
10060 LET p=0
10070 READ a$(p,c)
10080 a$(p,c)=LOWER$(a$(p,c))
10090 IF a$(p,c)<>"end" THEN p=p+1:GOTO
```

```
10070
10100 p$(c)="Y"
10110 NEXT c
10120 roc=0
10130 rno=0
10140 ul=20
10150 GOSUB 25000
10160 GOSUB 15000
```

Comentário

Linha 10040: a complexidade da tarefa resultará da variedade de quadros a serem empregues no decurso do programa. Serão explicados, à medida que forem sendo utilizados.

Linhas 10050-10110: os itens de dados são lidos para as três colunas do quadro A$. O quadro, tal como aqui se encontra organizado, pode conter até três conjuntos de 200 itens de dados, correspondentes às três vozes, ou canais, do *464*. Os itens de dados são carregados, inicialmente, para a primeira coluna. Quando for encontrada a palavra «end», o ciclo continuará a ler (READ) os dados, embora começando a colocá-los na coluna seguinte do quadro. A partir disto, pode ver-se que é vital a exterminação dos dados de cada voz com a palavra «end», tal como nos módulos-espécime DATA que foram dados. Finalmente, para cada voz, o elemento relevante do quadro P$ é regulado para «Y» para indicar, pelo menos inicialmente, que o canal determinado deve ser tocado.

Linha 10120: a variável ROC será utilizada para determinar em que oitava se baseará a melodia — os dados (DATA) da melodia serão interpretados como relativos a essa oitava-base.

Linha 10130: de forma similar, RNO é a nota base. Se for alterada durante o programa, a tecla em que a melodia é tocada mudará.

Linha 10140: a variável UL é utilizada para armazenar a duração normal da nota. Dentro dos dados da melodia, cada nota será um múltiplo da duração normal.

Linhas 10150-10160: fazem apelo a duas sub-rotinas, que começam o processamento dos dados da melodia de uma forma mais conveniente para o *464*.

Ensaio

Precisará de introduzir duas linhas temporárias:

15000 RETURN
25000 RETURN

Execute agora o programa. Deverá haver uma pausa considerável antes do retorno do cursor. Não é possível ainda fazer algo com os dados recolhidos, mas, pelo menos, você ficará a saber que não existem erros de sintaxe naquilo que introduziu.

Módulo 3.3.3: O «envelope» de som

O *envelope* de uma nota é, basicamente, a forma que dita o modo como ela nasce do nada, prossegue e, eventualmente, se retira novamente para o silêncio. A maioria dos sons e instrumentos musicais possuem *envelopes* bastante distintos. O(s) comando(s) de *envelope* para o programa é colocado neste local do programa para o lembrar de que, de muitas formas, ele faz parte dos dados da melodia. A alteração dos *envelopes* irá modificar o som do que for executado quase tanto como se se tratasse de uma alteração de notas.

Não vamos aqui abordar os aspectos técnicos dos *envelopes,* pois qualquer discussão útil tomaria muitas páginas e, honestamente, você pode aprender mais com a experimentação de diferentes regulações, uma vez introduzida a totalidade do programa.

Módulo 3.3.3: linhas 25000-25040

```
25000 REM ******************************
25010 REM Put ENV and ENT Commands Here
25020 REM ******************************
25030 ENV 1,7,-1,10,8,-1,40
25040 RETURN
```

Módulo 3.3.4: Processamento dos dados da melodia

Nesta altura, muitas pessoas ficarão surpreendidas por não passarmos à execução da melodia contida nas instruções de dados. De facto, a execução da melodia está ainda a alguma distância. É necessário, primeiro, processar os dados.

A razão para isto consiste em que, na nossa busca da forma mais fácil de registar uma melodia, abandonámos o formato requerido pelos diferentes comandos de som. A notação utilizada nas instruções de dados, como poderá verificar quando chegar a analisar os diferentes comandos nelas contidos, é fácil de recordar e utilizar, mas precisa de ser traduzida antes de ser fornecida ao comando de som (SOUND) do *464.* Por rápido que o BASIC do *464* seja, a tarefa de traduzir os comandos até um máximo de três vozes e executá-los simultaneamente está para além das possibilidades daquele código. Poderia fazer-se, mas verificar-se-ia uma limitação inaceitável na velocidade a que cada melodia poderia ser tocada.

O método adoptado é, portanto, levar a efeito qualquer tradução necessária, em primeiro lugar, armazenar os parâmetros para cada uma das notas a serem tocadas no quadro S% e, só então, executar a melodia, utilizando S% como fonte.

Módulo 3.3.4: linhas 15000-15140

```
15000 REM ******************************
15010 REM Compile Data
15020 REM ******************************
15030 FOR c=0 TO 2
15040 l=1:v=12:o=0:ev=0:et=0:p=0:r=1
15050 n=0
15060 WHILE a$(p,c)<>"end"
15070 FOR i=1 TO 8
15080 IF LEFT$(a$(p,c),1)<>MID$("ovl*[]e
r",i,1) THEN NEXT i
15090 ON i GOSUB 16000,17000,18000,19000
,20000,21000,22000,23000,24000
15100 p=p+1
15110 WEND
15120 d(c)=n
15130 NEXT c
15140 RETURN
```

Comentário

Linhas 15030-15130: o processo é levado a efeito para todas as três vozes, ou canais, do *464*.

Linha 15040: as variáveis desta linha serão utilizadas com os seguintes objectivos:

L — duração da nota
V — volume
O — oitava
EV — *envelope* de volume
ET — *envelope* de tom
P — posição, dentro da sequência, de comandos para uma voz
N — número de notas tocadas

Linhas 15070-15090: semelhante à técnica MENU$ utilizada no programa dos gráficos. Aqui, o ciclo compara o comando contido em A% com os comandos permitidos registados na cadeia da linha 15080. Se a primeira letra do comando tiver correspondência com uma das letras dentro da cadeia, então a variável de ciclo I regista a posição do comando, a qual é utilizada por ON... GOSUB. O efeito dos comandos individuais será discutido a propósito dos módulos que os abordam. Se a primeira letra do comando *não* for uma das letras da cadeia de comando, parte-se do princípio de que o comando é um valor de nota.

Linha 15120: o número de notas para cada voz é armazenado no quadro D.

Ensaio

O único ensaio, e que nem é grande coisa, consiste em colocar comandos RETURN em todos os destinos especificados na linha 15090. O programa será então executado, de modo que você possa ver que não existem erros de sintaxe. A adição destas linhas RETURN permitir-lhe-á também executar o programa de cada vez que introduzir um dos módulos seguintes, que efectuam o processamento dos dados da melodia.

Módulo 3.3.5: O valor de uma nota

A maior parte de uma melodia vai ser constituída, como é perfeitamente natural, por notas, sendo este módulo o que pega na representação das notas, a partir dos dados da melodia, e a torna compreensível.

Os próprios valores de nota são introduzidos na base do sistema de doze tons. Uma oitava, para a música ocidental, divide-se em oito tons inteiros e quatro semitons, sendo doze no total, de modo que cada nota de uma oitava possa ser representada por um número de 0 a 11. Uma vez que existe uma relação matemática perfeitamente nítida entre as notas de uma oitava, ou mesmo entre as notas de oitavas diferentes, é perfeitamente possível ir além da gama de 0 a 11, sendo 12 a primeira nota da oitava acima, 24 a primeira nota da oitava seguinte e assim por diante, deixando a este módulo a tarefa da tradução.

Módulo 3.3.5: Linhas 24000-24070

```
24000 REM ******************************
24010 REM Note
24020 REM ******************************
24030 fr=440*(2^(o+(VAL(a$(p,c))-9)/12))
24040 tp=ROUND(125000/fr)
24050 s%(c,n,0)=tp:s%(c,n,1)=l:s%(c,n,2)
=v:s%(c,n,3)=ev:s%(c,n,4)=et
24060 n=n+1
24070 RETURN
```

Comentário

Linha 24030: esta fórmula traduz o valor de nota (tomando em consideração o valor de oitava, 0, que discutiremos mais adiante) para um valor de frequência.

Linha 24040: tendo chegado a valor de frequência, esta linha tradu-lo para um valor que pode ser usado para conduzir o comando SOUND — não há nada de especial quanto ao método, a não ser que o *464* está concebido para funcionar com 125000/TRUE FREQUENCY (FREQUÊNCIA VERDADEIRA).

Linha 24050: a presença de um valor de nota nos dados da melodia é assumida como comando para tocar uma nota, de modo que os variados itens que concorrem para a formação de um som (SOUND) são transferidos das variáveis em que se encontram armazenados para uma linha do quadro S%, a qual será eventualmente utilizada na execução da melodia.

Módulo 3.3.6: Alteração da oitava

Terá notado, na fórmula de extracção do valor de uma nota, que o valor da oitava (0) é tido em consideração. Isto permite uma maior flexibilidade na forma como os valores de nota podem ser introduzidos. A primeira nota da segunda oitava tanto pode ser introduzida como 12, como a oitava ser alterada para 1, sendo então o valor de nota especificado para 0. O método para especificar a alteração de uma oitava é incluir um 0, seguido de um número.

Módulo 3.3.6: linhas 16000-16040

```
16000 REM ******************************
16010 REM Octave
16020 REM ******************************
16030 o=VAL(MID$(a$(p,c),2))
16040 RETURN
```

Módulo 3.3.7: Especificação do volume

O volume é especificado incluindo V, seguido de um volume válido nos dados da melodia.

Módulo 3.3.7: linhas 17000-17040

```
17000 REM ******************************
17010 REM Volume
17020 REM ******************************
17030 v=VAL(MID$(a$(p,c),2))
17040 RETURN
```

Módulo 3.3.8: Marcação da duração da nota

A duração da nota é especificada incluindo L, seguido de um valor nos dados da melodia. A duração introduzida nos dados da melodia será multiplicada pela duração da nota de base estabelecida no módulo de inicialização.

Módulo 3.3.8: linhas 18000-18040

```
18000 REM ******************************
18010 REM Length
```

```
18020 REM ******************************
18030 l=VAL(MID$(a$(p,c),2))
18040 RETURN
```

Módulo 3.3.9: Repetição de uma secção da melodia

A maioria das melodias contêm repetições. De facto, uma melodia sem repetição seria, muito provavelmente, um pouco incómoda de ouvir. Esta repetição será, por vezes, de pequenas secções da melodia e, outras vezes, de grandes pedaços. Para poupar espaço, é razoável incluir no programa a possibilidade de marcar uma secção da melodia a ser executada duas vezes consecutivas. No programa corrente, uma tal secção é marcada no início por um asterisco e no fim por um asterisco seguido de um número, o que indica quantas vezes a secção deve ser tocada.

Módulo 3.3.9: linhas 19000-19090

```
19000 REM ******************************
19010 REM Repeat
19020 REM ******************************
19030 IF a$(p,c)="*" THEN r=1:RETURN
19040 IF VAL(MID$(a$(p,c),2))=r THEN RET
URN
19050 r=r+1
19060 WHILE a$(p,c)<>"*"
19070 p=p-1:IF p=-1 THEN RETURN
19080 WEND
19090 RETURN
```

Comentário

Linha 19030: para chegar a este módulo, o comando nos dados da melodia deve *começar* com um asterisco. Se for apenas um asterisco, a variável R, que armazena o número de vezes que a secção foi repetida, é regulada para 1. O controlo retorna então à parte principal do programa, para que possa continuar a traduzir a melodia.

Linha 19040: se o asterisco for seguido de um número, será o fim de uma secção. O valor de R, que mostra quantas vezes a secção foi repetida, é comparado com o valor a seguir ao asterisco, que mostra quantas vezes *deveria* ter sido repetida. Se os valores forem os mesmos, então o programa prossegue para lá dessa secção.

Linhas 19050-19080: é aumentado o número de repetições registado e o programa retrocede para o início da secção a ser repetida, antes de regressar à sua parte principal.

Módulo 3.3.10: Variação de secções repetidas

Acontece muitas vezes que, quando uma secção de uma melodia é repetida, são feitas pequenas alterações — a segunda repetição pode terminar em clímace de uma sucessão de notas altas, em que a primeira repetição terminara em notas baixas. O módulo corrente permite ao utilizador especificar que, dentro de uma secção repetida, certo número de notas apenas seja incluído durante *uma* das repetições. Desta forma, por exemplo, uma secção poderia ser repetida duas vezes, com um final para a primeira repetição e um final diferente para a segunda. O método neste programa consiste em marcar o início da subsecção com um símbolo [, seguido de um valor representativo do número da repetição durante a qual a subsecção será executada, e o fim da subsecção com um símbolo] apenas.

Módulo 3.3.10: linhas 20000-21030

```
20000 REM ******************************
20010 REM On nth Repeat Do
20020 REM ******************************
20030 IF VAL(MID$(a$(p,c),2))=r THEN RET
URN
20040 WHILE a$(p,c)<>"]" AND a$(p+1,c)<>
"end"
20050 p=p+1
20060 WEND
20070 RETURN
21000 REM ******************************
21010 REM End Do
21020 REM ******************************
21030 RETURN
```

Comentário

Linha 20030: para chegar a este ponto, o programa deverá ter encontrado nele incluído um símbolo [. Esta linha examina o número que se segue a [. Se o número for o mesmo do valor de R, que regista o número de vezes que a secção corrente foi repetida, o programa será autorizado a continuar as notas que se seguem a].

Linhas 20040-20060: este ciclo é utilizado se a subsecção não tiver de ser executada durante a repetição, passando simplesmente por cima da subsecção, até encontrar o símbolo de finalização].

Linha 21030: quando um] é encontrado, esta linha apenas permite a continuação do programa.

Módulo 3.3.11: Alteração de «envelopes»

Estabelecemos já um *envelope,* mas você deve saber que o *464* pode definir até 16 *envelopes* para volume e tom. Se se incluir ENV ou ENT, seguidos (sem espaço) de um número válido para um *envelope* que definimos no módulo em 25000, isso irá alterar o *envelope* utilizado pelo comando SOUND.

Módulo 3.3.11: linhas 22000-22050

```
22000 REM ******************************
22010 REM Envelopes
22020 REM ******************************
22030 IF LEFT$(a$(p,c),3)="env" THEN ev=
VAL(MID$(a$(p,c),4))
22040 IF LEFT$(a$(p,c),3)="ent" THEN et=
VAL(MID$(a$(p,c),4)
22050 RETURN
```

Módulo 3.3.12: Criação de um silêncio

Não deve esquecer-se de que o silêncio é tão importante a uma melodia como o som, pelo que precisamos de um método para parar o som durante alguns momentos. No programa corrente, é criada uma interrupção através da inclusão de R nos dados da melodia. Um único R cria uma pausa equivalente à duração de uma unidade de nota, enquanto R seguido de um número cria uma pausa igual ao número, em duração.

Módulo 3.3.12: linhas 23000-23060

```
23000 REM ******************************
23010 REM Rest
23020 REM ******************************
23030 IF a$(p,c)="r" THEN du=1 ELSE du=V
AL(MID$(a$(p,c),2))*1
23040 s%(c,n,0)=0:s%(c,n,1)=du:s%(c,n,2)
=0:s%(c,n,3)=0:s%(c,n,4)=0
23050 n=n+1
23060 RETURN
```

Ensaio

Se não tem efectuado os ensaios após a introdução dos módulos individuais, pode agora executar o programa. Cada uma das secções será chamada a actuar, à medida que se der a análise dos dados da melodia.

Módulo 3.3.13: O «menu» do programa

Trata-se de um simples *menu,* que permite o acesso às restantes funções do programa.

Módulo 3.3.13: 11000-11140

```
11000 REM ******************************
11010 REM Control
11020 REM ******************************
11030 WHILE o<>3
11040 CLS:PRINT TAB(18);"MUSIC";TAB(18);
"====="
11050 PRINT:PRINT"Options:":PRINT
11060 PRINT"1) Change parameters"
11070 PRINT"2) Play music"
11080 PRINT"3) End"
11090 PRINT:INPUT"Enter option: ",o
11100 ON o GOSUB 12000,13000
11110 WEND
11120 CLS
11130 LIST 25000-
11140 END
```

Módulo 3.3.14: Alteração dos parâmetros da melodia

Quando chega a altura de tocar a melodia, são utilizados certos parâmetros que o utilizador pode modificar, usando este módulo.

Módulo 3.3.14: linhas 12000-12110

```
12000 REM ******************************
12010 REM Change Parameters
12020 REM ******************************
12030 CLS:PRINT TAB(12);"CHANGE PARAMETE
RS";TAB(12);"================":PRINT
12040 PEN 2:PRINT"ENTER";:PEN 1:PRINT" t
o leave parameter unchanged":PRINT
12050 PRINT"Relative octave (";roc;:INPU
T") :",x$:IF x$<>"" THEN roc=VAL(x$)
12060 PRINT"Relative note (";rno;:INPUT"
) :",x$:IF x$<>"" THEN rno=VAL(x$)
12070 PRINT"Unit length (";ul;:INPUT") :
",x$:IF x$<>"" THEN ul=VAL(x$)
12080 FOR c=0 TO 2
12090 PRINT"Play channel";c;"(";p$(c);:I
```

```
NPUT") :",x$:IF x$<>"" THEN p$(c)=UPPER$
(x$)
12100 NEXT c
12110 RETURN
```

Comentário

Linha 12050: a oitava relativa é a base a partir da qual é calculada a nota a ser tocada.

Linha 12060: com a oitava básica, as notas tocadas podem ser calculadas na base de uma nota específica, alterando assim a chave da música.

Linha 12070: trata-se da unidade que é utilizada quando se toca uma nota. A música será mais lenta se se introduzir um valor superior.

Linhas 12080-12100: nem todas as vozes têm de ser tocadas e este ciclo permite a exclusão de qualquer das vozes.

Ensaio

Se o programa for executado, após uma pausa para recalcular os dados da melodia, deverá aparecer o *menu*. Você deverá poder escolher a opção 1 do *menu* e especificar quaisquer valores que queira como parâmetros.

Módulo 3.3.15: Controlo de execução da música

Passamos agora aos dois módulos que levarão a efeito o objectivo principal do programa, ou seja, tocar uma melodia. O primeiro destes dois módulos é um módulo de controlo que distribui trabalho e fornece o final da melodia; o outro módulo terá o trabalho de tocar as notas.

Módulo 3.3.15: linhas 13000-13160

```
13000 REM ******************************
13010 REM Play Music
13020 REM ******************************
13030 rtp=2^-(roc+rno/12)
13040 FOR c=0 TO 2
13050 p(c)=0
13060 NEXT c
13070 SOUND 199,0,0
13080 GOSUB 14000
13090 GOSUB 14000
13100 GOSUB 14000
```

```
13110 RELEASE 7
13120 done=0
13130 WHILE NOT done
13140 GOSUB 14000
13150 WEND
13160 RETURN
```

Comentário

Linha 13030: é a base a partir da qual será executada a melodia.

Linas 13040-13060: o quadro P de contagem, para as três vozes, é regulado para 0.

Linha 13070: esta linha não tem a função de emitir uma nota, antes se trata de um comando de gestão que faz três coisas:

1) Varrer quaisquer notas que possam permanecer na bicha para serem tocadas, pertencentes a alguma utilização anterior.
2) Especificar que devem ser enviados comandos para as três vozes.
3) Colocar uma suspensão, para que não possam ser tocadas quaisquer notas.

O número 199, por si só, não tem significado: é apenas um método para passar certos dígitos binários ou «*bits*» para binário 1. No caso de 199, são passados os *bits* 0, 1, 2, 6 e 7. São dados pormenores sobre os efeitos destes *bits* no cap. 6 do seu manual.

Linhas 13080-13110: estas três chamadas ao próximo módulo estabelecem uma fila de notas à espera de serem tocadas. Isto significa que quaisquer ligeiros pormenores introduzidos pelo ciclo que executa a parte principal da melodia não terão efeito algum sobre a regularidade da própria melodia. O comando RELEASE, na linha 13110, inicia as três vozes simultaneamente, uma vez formadas as primeiras notas da fila de espera. Esta possibilidade do *464* para enfileirar constitui uma das suas características musicais mais importantes. Em quase todos os outros micros domésticos, o problema de temporizar a execução das notas mostra-se extremamente difícil, resultando qualquer atraso no processamento de uma nota em temporização desigual. Com o *464,* tal problema é eliminado e o programa pode ser mais simples e mais eficaz.

Linhas 13120-13140: o resto do processo de execução da melodia é apenas uma questão de chamar continuamente o módulo seguinte, até que a variável DONE, que é estabelecida pelo próximo módulo, indique que foram esgotados os dados da melodia.

Módulo 3.3.16: Execução das notas

É o módulo final, que faz soar as notas das três vozes.

Módulo 3.3.16: linhas 14000-14120

```
14000 REM ******************************
14010 REM Scan Channels
14020 REM ******************************
14030 done=-1
14040 FOR c=0 TO 2
14050 IF p(c)=d(c) OR p$(c)="N" THEN GOT
O 14110
14060 done=0
14070 IF (SQ(2^c) AND 7)=0 THEN GOTO 141
10
14080 n=p(c)
14090 SOUND 2^c,s%(c,n,0)*rtp,s%(c,n,1)*
ul,s%(c,n,2),s%(c,n,3),s%(c,n,4)
14100 p(c)=p(c)+1
14110 NEXT c
14120 RETURN
```

Comentário

Linhas 14040-14050 e 14110: sem atender ao facto de todas as três vozes deverem ser tocadas ou não, este ciclo está organizado para analisar todas elas. A linha 14050, no entanto, detecta se uma determinada voz esgotou todos os seus dados, ou se o utilizador restabeleceu os parâmetros, de forma a excluir uma voz determinada. Se a parte principal do ciclo não tiver sido alcançada (isto é, se ainda houver notas para tocar durante uma voz, pelo menos), então o valor de DONE passará a 0.

Linha 14070: a função SQ é utilizada para determinar se é possível colocar outra nota na fila de determinada voz.

Linhas 14080-14100: se *for* possível acrescentar outra nota à fila, o próximo conjunto de itens de dados a partir de S% será utilizado como base para o comando SOUND — note que este não faz imediatamente soar uma nota, mas apenas a acrescenta à fila de notas que esperam para ser tocadas. Finalmente, é incrementado o contador da voz específica.

Ensaio

Finalmente pode efectuar um teste completo do programa. Para isso, basta fazê-lo «correr» e, quando o *menu* aparecer, especificar a opção 2. Logo verá se está ou não a trabalhar!

CAPÍTULO 4

Cada vez mais sério

Nesta fase do seu progresso, deve estar a tornar-se mais familiarizado com as capacidades do *464*, bem como com algumas das técnicas necessárias para o pôr a funcionar para si. Chegou o momento, portanto, de observarmos alguns programas substanciais que permitirão ao *464* fazer aquilo que os microcomputadores sabem fazer melhor — tratar, seleccionar e retirar informações para os seus donos.

Os programas neste capítulo são:

UNIFICHEIRO: um potente sistema pessoal de arquivo, capaz de armazenar uma vasta variedade de informação, para chamada imediata.

NÚMERO: é um programa que cria um dicionário de Nomes e Números para quase tudo o que desejar, permitindo ao utilizador criar facturas, avaliações de bolsa ou mesmo uma contagem das calorias para a ementa do dia.

TEXTO: um simples programa-produto para processamento de texto, que é executado em BASIC.

MULTIQ: um gerador de ensaio de escolha múltipla.

PROGRAMA 4.1: UNIFICHEIRO

Função do programa

«Unificheiro» é o corolário de um programa que foi desenvolvido ao longo dos anos nos livros da série «Micro Funcional». Leitores de livros anteriores escreveram a dizer que o utilizam nos seus negócios, no ensino de crianças em idade escolar acerca da forma como os micros tratam informações, na ajuda a clubes e a organizações voluntárias, ou simplesmente na manutenção de registos dos seus livros e discos.

Os novos conceitos apresentados neste programa incluem:

1) Pesquisa dicotómica.
2) Inserção de itens em cadeias contínuas.
3) Iniciação de um programa sem limpar quadros.

Módulo 4.4.1: Organização da estrutura do ficheiro

Muitos dos livros destinados ao possuidor de um micro pessoal oferecem programas de ficheiro inferiores, que são extremamente inflexíveis na utilização. Está incutido no programa que, de cada vez que o utilizador armazenar alguma coisa, será sob os cabeçalhos Nome, Endereço, Número de Telefone ou alguma estrutura similar. A beleza do «Unificheiro» está em que, enquanto lhe permite indiscutivelmente usar uma tal estrutura, permite-lhe igualmente criar outros ficheiros com estruturas muito diferentes — talvez com um cabeçalho, talvez com dez — sem fazer quaisquer alterações ao próprio programa. O «Unificheiro» é aquilo que eu gosto de chamar um «programa camaleão»: um programa que se adapta a uma variedade de diferentes utilizações, reagindo perante o utilizador de diferentes formas, dependendo da tarefa que estiver a executar no momento.

O objectivo deste primeiro módulo é inicializar algumas variáveis, mas, o que é mais importante é que permite ao utilizador organizar o ficheiro original da forma exactamente desejada.

```
ENTRY  1 :

NAME:BLOGGS
C/NAME:JOE
ADDRESS 1:11 ANYSTREET
ADDRESS 2:ANYTOWN
ADDRESS 3:ANYSHIRE
PHONE:01 234 5678

COMMANDS AVAILABLE:

ENTER for next item
'A' to amend
'C' to continue search
'#' followed by number to move pointer
'\' to quit function

Enter Choice:
```

Fig. 4.1 — *«Unificheiro» em modo de pesquisa*

Módulo 4.1.1: linhas 20000-20110

```
20000 REM ******************************
20010 REM Create File
20020 REM ******************************
20030 CLS:PRINT TAB(16);"NEW FILE":PRINT
 TAB(16);"========"
20040 PRINT:INPUT"Are you sure (y/n)";r$
:IF LEFT$(UPPER$(r$),1)="N" THEN RETURN
20050 CLEAR:DIM array$(1000)
20060 PRINT:INPUT"How many items in each
 entry";x
```

```
20070 DIM item$(x-1),ptr(x-1)
20080 PRINT:FOR i=0 TO x-1
20090 PRINT"Name of item #";i+1;:INPUT":
",item$(i)
20095 item$(i)=UPPER$(item$(i))
20100 NEXT i
20110 in=-1:GOTO 11000
```

Comentário

Linhas 20030-20040: a criação de novos ficheiros apagará qualquer informação correntemente em memória, pelo que é dada ao utilizador a possibilidade de parar nesta altura.

Linha 20050: a informação a ficar contida no «Unificheiro» será armazenada no quadro ARRAY$. Na linha corrente, esta é dimensionada para permitir 1001 entradas no ficheiro. Pode aumentar este número para 2000 ou mais, se o desejar. A única coisa a ter presente é que cada elemento do quadro, antes de ser utilizado, toma três *bytes* de memória. Se o quadro for dimensionado para 5000 itens, estarão a ser utilizados 15000 *bytes* de memória, um terço da memória livre, antes de ser armazenada qualquer informação. Trata-se apenas de uma questão de discernimento — se pensa que vai precisar de menos de 1000 entradas num ficheiro, então mais vale deixar a instrução DIM tal como está, economizando um pouco de memória. Note também que, por estarmos a utilizar um elemento de um quadro para armazenar cada entrada, o comprimento máximo de uma só entrada, incluindo os caracteres separadores, será de 255 caracteres — o comprimento máximo de cadeia que o *464* pode tratar.

Linhas 20060-20100: são uma parte do segredo da flexibilidade do «Unificheiro». Por cada *entrada,* que pode imaginar como uma ficha de cartão, se isso o ajuda, pode definir quantos itens aparecerem na entrada. Se você estivesse a registar a sua colecção de discos, poderia utilizar cabeçalhos como: FAIXA, ÁLBUM, COMPOSITOR, INTÉRPRETE ou DURAÇÃO. Neste caso, especificaria cinco itens, introduzindo depois os nomes dos cinco. De futuro, sempre que utilizar o «Unificheiro» para armazenar ou retirar informação sobre a sua colecção de discos, ser-lhe-á solicitada a introdução de um item de informação sob cada um dos cabeçalhos. Para aqueles que se preocupam mais com os problemas técnicos, quando se designa toda a ficha de cartão por *entrada* e os cabeçalhos individuais por *itens,* isso equivale, em linguagem informática, a *registos* e *campos,* respectivamente. A utilização das variáveis aqui definidas será explicada no decurso do comentário sobre o programa.

Módulo 4.1.2: Arranque do programa

A possibilidade de criar um novo ficheiro não é utilizada em todas as ocasiões, durante a execução do programa. Na maioria dos casos, o utilizador pretenderá recarregar dados a partir de cassete. Este módulo de arranque apenas dá ao utilizador a oportunidade de dizer qual das duas opções deseja.

Módulo 4.1.2: linhas 10000-10150

```
10000 REM ******************************
10010 REM Start
10020 REM ******************************
10030 WHILE NOT in
10040 CLS
10050 PRINT TAB(16);"UNIFILE"
10060 PRINT TAB(16);"======="
10070 PRINT
10080 PRINT"COMMANDS AVAILABLE:"
10090 PRINT
10100 PRINT"1) Create new file"
10110 PRINT"2) Load file from tape"
10120 PRINT
10130 INPUT"Enter choice: ",z
10140 ON z GOSUB 20000,21000
10150 WEND
```

Comentário

Linhas 10030 e 10150: estas duas linhas representam uma técnica muito útil, que você quererá, sem dúvida, aplicar nos seus próprios programas. Quando o programa é inicializado pelo primeiro módulo entrado, uma variável chamada IN (abreviatura de INicializado) é regulada para o valor −1. A razão disto é tornada clara por este ciclo. Se o valor de IN for −1 ao ser alcançada a linha 10030, então nenhuma parte deste ciclo será executada. O operador NOT ligado a WHILE, na linha 10030, assegura que o ciclo apenas será introduzido se o valor de IN for 0. Significa isto que, se se parar o programa e voltar a iniciá-lo com GOTO 10000 (ou GOTO 1, se foi incluído o pequeno módulo SAVE que eu recomendei no primeiro programa do livro), nenhum dos dados em memória se perderá.

Ensaio

Introduza uma linha temporária:

```
11000 STOP
```

Execute o programa e especifique que pretende organizar um novo ficheiro. Siga as solicitações dadas e introduza alguns valores e nomes razoáveis como resposta. O programa deverá então parar. Introduza agora:

```
GOTO 10000[ENTER]
```

e o programa deverá parar imediatamente — detectou que existe um ficheiro em memória, ainda que seja vazio, e passou em volta do módulo na linha 10000.

Módulo 4.1.3: O «menu»

Trata-se de um *menu* normal, mas note que não está previsto que o programa pare sob acção de ON ERROR e da tecla ESC. O «Unificheiro» é um programa complexo e pará-lo a meio fluxo poderia significar que a informação nele contida se corrompesse, por não ter sido completado um processo importante. Para terminar o «Unificheiro», deverá *sempre* regressar a este *menu* e utilizar a opção 6.

Módulo 4.1.3: linhas 11000-11210

```
11000 REM ******************************
11010 REM Main Menu
11020 REM ******************************
11030 WHILE NOT done
11040 CLS
11050 PRINT TAB(16);"UNIFILE"
11060 PRINT TAB(16);"======="
11070 PRINT
11080 PRINT"COMMANDS AVAILABLE:"
11090 PRINT
11100 PRINT"1) Create new file"
11110 PRINT"2) Load file from tape"
11120 PRINT"3) Enter information"
11130 PRINT"4) Search/Display/Change"
11140 PRINT"5) Save file to tape"
11150 PRINT"6) Stop"
11160 PRINT
11170 INPUT"Enter choice: ",z
11180 ON z GOSUB 20000,21000,12000,17000
,18000,22000
11190 WEND
11200 done=0
11210 END
```

Módulo 4.1.4: Paragem do programa

Este pequeno módulo visualiza a mensagem de finalização do programa e estabelece o valor da variável DONE de modo a informar o *menu* de que o programa deve parar.

Módulo 4.1.4: linhas 22000-22060

```
22000 REM ******************************
22010 REM Stop
22020 REM ******************************
22030 done=-1
```

```
22040 CLS:LOCATE 11,13
22050 PRINT"FILING SYSTEM CLOSED":PRINT:
PRINT
22060 RETURN
25000 OPENIN "unifile"
25010 INPUT t$
25020 PRINT t$
25030 GOTO 25010
```

Ensaio

Deverá agora poder introduzir:

GOTO 11000[ENTER]

e obter a visualização do *menu*. Especifique depois a opção 6 e terá a finalização do programa.

Módulo 4.1.5: Salvaguarda de dados em fita

À medida que vai desenvolvendo programas mais complexos, quer a partir deste livro ou por sua própria iniciativa, notará cada vez mais o desejo de introduzir o módulo de ficheiro de dados (DATA FILE) tão cedo quanto possível. A razão disto é que apenas se podem fazer ensaios convenientes do «Unificheiro» introduzindo quantidades bastante apreciáveis de dados. Uma vez que você pode cometer erros e os dados serem corrompidos, encontra-se perante a perspectiva de ter de reintroduzir esses mesmos dados uma vez após outra, durante o desenvolvimento do programa. A resposta está na gravação dos dados em fita, desde as fases mais prematuras, chamando depois as informações a partir da fita, para efectuar os seus ensaios.

O módulo corrente permitir-lhe-á armazenar dados e iniciá-lo-á igualmente numa ideia a que já fizemos referência, mas que, na verdade, não utilizámos: a de dar a um programa a capacidade de recolher ou armazenar ficheiros com uma grande variedade de nomes.

Módulo 4.1.5: linhas 18000-18080

```
18000 REM ******************************
18010 REM Save File
18020 REM ******************************
18030 CLS:PRINT TAB(16);"SAVE FILE":PRIN
T TAB(16);"=========="
18040 PRINT:INPUT"Name under which to sa
ve: ",fi$
18050 PRINT:OPENOUT fi$:PRINT#9,it:PRINT
#9,x
```

```
18060 FOR i=0 TO it-1:PRINT#9,array$(i):
NEXT i
18070 FOR i=0 TO x-1:PRINT#9,item$(i):NE
XT i
18080 CLOSEOUT:RETURN
```

Comentário

Linhas 18030-18040: o nome do ficheiro é especificado interactivamente.

Linhas 18050-18070: o número de entradas contidas no «Unificheiro» em qualquer momento é armazenado na variável IT, enquanto X regista o número de itens em cada entrada. As linhas registam todos os elementos de utilidade a partir de ARRAY$ e os nomes dos cabeçalhos a partir de ITEM$.

Módulo 4.1.6: Carregamento de dados a partir da fita

Após termos colocado os dados em fita, a tarefa é tê-los de volta, o que não é tão simples como isso, porque, enquanto no momento em que os dados foram salvaguardados todos os quadros estavam já correctamente dimensionados para o ficheiro em memória, quando um ficheiro é carregado a partir da fita todos os quadros devem ser estabelecidos de novo. É esta a razão pela qual as variáveis IT e X foram colocadas no início do ficheiro, para que possam ser lidas para a memória em primeiro lugar, sendo depois utilizadas para dimensionar os quadros exactamente do mesmo modo que o primeiro módulo, em que foram introduzidas pelo utilizador, em vez de o serem pelo gravador de dados.

Módulo 4.1.6: linhas 21000-21140

```
21000 REM ******************************
21010 REM Load File
21020 REM ******************************
21030 CLS:PRINT TAB(16);"LOAD FILE":PRIN
T TAB(16);"=========="
21040 PRINT:INPUT"Are you sure (y/n)";r$
:IF LEFT$(UPPER$(r$),1)="N" THEN RETURN
21050 CLEAR:DIM array$(1000)
21060 PRINT:INPUT"Name of file to load:
",fi$
21070 PRINT:OPENIN fi$:INPUT #9,it,x:DIM
 item$(x-1),ptr(x-1)
21080 FOR i=0 TO it-1
21090 INPUT #9,array$(i)
21100 NEXT i
21110 FOR i=0 TO x-1
```

```
21120 INPUT #9,item$(i)
21130 NEXT i
21140 in=-1:CLOSEIN:GOTO 11000
```

Comentário

Linha 21140: um certo número de pessoas podem ficar intrigadas com a presença de um GOTO neste ponto, no final de uma sub-rotina. E não pode ser substituído por um RETURN, já que, no decurso do módulo, a memória foi limpa para permitir o redimensionamento dos quadros. Ao ser limpa a memória, todo o registo do comando GOSUB original se perdeu e, desta forma, RETURN apenas produziria uma mensagem de erro.

Ensaio

Execute o programa e especifique que pretende criar um novo ficheiro. Estabeleça quatro itens, cujos nomes sejam UM, DOIS... etc. Quando o *menu* aparecer, escolha a opção 5 e especifique um nome de ficheiro como UNIDATA. Assegure-se de que a fita correcta se encontra no gravador antes de começar a gravação. Quando o programa regressar ao *menu,* faça-o parar com a opção 6 e execute-o de novo. Desta vez, especifique que pretende carregar a partir da fita, especifique também o nome do ficheiro de dados e volte a passar o ficheiro de dados sob comando (depois de ter rebobinado a fita, é claro). Quando o *menu* aparecer, faça parar o programa novamente. Introduza então:

FOR I = 0 TO X – 1:PRINT ITEM$(I):NEXT[ENTER]

e o resultado deverá ser a impressão dos seus quatro cabeçalhos.

Módulo 4.1.7: Uma forma melhor de pesquisar

Neste módulo, e nos dois que se seguem, damos atenção à forma como uma nova entrada é adicionada ao ficheiro principal, contido em ARRAY$. É aqui que se encontra o núcleo do método, no entanto, já que é este o módulo que permite ao «Unificheiro» pesquisar rapidamente através de um vasto ficheiro de entradas, procurando um local correcto para inserir uma nova entrada ou para conduzir uma pesquisa rápida sobre a presença de uma entrada de tecla no ficheiro.

Este método é conhecido por «pesquisa dicotómica» e pode ser utilizado para reduzir drasticamente o tempo de pesquisa em quaisquer programas que contenham longas listas de dados ordenados. Considere o exemplo que se segue.

Foi estabelecido um ficheiro contendo 2000 nomes. A tarefa consiste em inserir um novo nome no ficheiro, pela ordem alfabética correcta. Se fizermos um pouco de batota e olharmos para a lista de nomes, poderemos determinar que o novo nome «YOUNGERS» irá inserir-se no ficheiro na posição 1731, embora o computador não tenha meios para saber disto antecipadamente.

Uma coisa que é possível fazer é pôr o computador a examinar os nomes um por

um, desde o princípio. Assim, começará por «ADAMS», verificando que «YOUNGER» deve vir depois, passará a «ADAMSON» e assim por diante. Em certa altura, depois de ter examinado 1732 nomes, a pesquisa dará com um nome como «YOUNGMAN», o qual deverá vir *depois* de «YOUNGER», pelo que a posição correcta terá sido encontrada.

Este método é fiável, mas como seria melhor se o número de comparações feitas pudesse ser um pouco reduzido. Bem, no caso do nosso ficheiro de 2000 nomes, todo o processo pode ser efectuado com *10 comparações* apenas. Eis como isso é feito.

O computador inicia a pesquisa examinando o nome na posição 1024 do ficheiro, porque 1024 é a maior potência de dois (2^10) a caber no número total de entradas (2000). O nome na posição 1024 é considerado alfabeticamente menor do que «YOUNGER», pelo que o computador adiciona 1024/2, ou 512 ou 2^9, à posição de pesquisa original, chegando até 1536. Uma vez mais, o nome nesta posição é alfabeticamente menor que «YOUNGER», pelo que, desta vez, 256, ou 2^8, é adicionado a 1536, perfazendo 1792. Agora, algo diferente acontece, porque o nome na posição 1792 é *posterior,* pela ordem alfabética, a «YOUNGER». Assim, em vez de ser adicionado 128, ou 2^7, este valor é *subtraído* à posição de pesquisa, resultando em 1664.

A pesquisa prossegue, adicionando ou subtraindo decrescentes potências de 2 e formando um padrão de pesquisa com o seguinte aspecto:

COMPARAÇÃO N.º	POSIÇÃO	ACÇÃO
1	1024	+512
2	1536	+256
3	1792	−128
4	1644	+64
5	1728	+32
6	1760	−16
7	1744	−8
8	1736	−4
9	1732	−2
10	1730	+1

Experimente, você mesmo, com diferentes números de entradas e diferentes posições-alvo da ordem — verificará que funciona sempre.

Módulo 4.1.7: linhas 13000-13110

```
13000 REM ******************************
13010 REM Binary Search
13020 REM ******************************
13030 IF it=0 THEN ss=0:RETURN
13040 po=INT(LOG(it)/LOG(2)):ss=2^po-1
13050 FOR i=po-1 TO 0 STEP -1
13060 ss=ss+2^i*((array$(ss)>te$)-(array
$(ss)<te$))
```

```
13070 IF ss<0 THEN ss=0
13080 IF ss>it-1 THEN ss=it-1
13090 NEXT i
13100 IF array$(ss)<te$ THEN ss=ss+1
13110 RETURN
```

Comentário

Linha 13030: se não houver entradas no ficheiro, os cálculos conduzirão a um erro. Esta linha evita tal situação.

Linha 13040: estas duas expressões encontram a maior potência de 2 que caiba no número de itens do ficheiro e depois igualam o apontador de pesquisa (SS) a esse número. O « – 1» na segunda expressão toma nota do facto que o quadro se encontra numerado a partir de 0 e não a partir de 1.

Linhas 13050-13090: este ciclo conduz a pesquisa, utilizando decrescentes potências de 2. O ficheiro principal está contido em ARRAY$ e a nova entrada em TE$. O valor do apontador é estabelecido por duas condições lógicas, que indicam se TE$ é maior ou menor do que a entrada em ARRAY$(SS).

Linha 13100: em alguns casos, a posição atingida estará 1 abaixo da posição correcta — neste caso, SS será aumentado em 1.

Módulo 4.1.8: Inserção de um item

Este módulo insere a nova entrada na posição indicada pela variável SS. Para isto, apenas é necessário deslocar para a frente um lugar tudo o que estiver depois da posição SS.

Módulo 4.1.8: linhas 14000-14070

```
14000 REM ******************************
14010 REM Insert Entry
14020 REM ******************************
14030 IF it=0 THEN GOTO 14070
14040 FOR i=it TO ss+1 STEP -1
14050 array$(i)=array$(i-1)
14060 NEXT i
14070 array$(ss)=te$:it=it+1:RETURN
```

Módulo 4.1.9: Entradas no ficheiro

Após a introdução dos módulos que efectuam o trabalho real, podemos agora passar ao módulo com que o utilizador terá contacto quando colocar material no sistema de ficheiro. A função do módulo é solicitar ao utilizador a introdução do número correcto de itens para cada entrada, pela ordem certa, para combinar esses itens numa única cadeia que caiba numa linha de ARRAY$ e depois chamar os dois módulos anteriores, para inserir a entrada no ficheiro principal.

Módulo 4.1.9: linhas 12000-12320

```
12000 REM ******************************
12010 REM New Entries
12020 REM ******************************
12030 WHILE it<1000
12040 te$=""
12050 CLS
12060 PRINT TAB(15);"NEW ENTRIES"
12070 PRINT TAB(15);"==========="
12080 PRINT
12090 PRINT"Entry number";it+1
12100 PRINT
12110 PRINT"COMMANDS AVAILABLE:"
12120 PRINT
12130 PRINT"Enter item specified"
12140 PRINT"'\' to return to main menu"
12150 PRINT
12220 FOR i=0 TO x-1
12230 PRINT item$(i);:INPUT":",q$
12240 IF q$="\" THEN RETURN
12250 te$=te$+UPPER$(q$)+"\"
12260 NEXT i
12270 PRINT:PRINT"Please wait for a mome
nt ..."
12280 GOSUB 13000:GOSUB 14000
12290 WEND
12300 CLS
12310 PRINT"*** SORRY, NO MORE ENTRIES P
OSSIBLE ***"
12320 FOR i=1 TO 2000:NEXT i:RETURN
```

Comentário

Linhas 12030 e 12290-12320: as entradas apenas serão aceites se houver espaço para elas. Caso contrário, será gerada uma mensagem de erro. Note que, se preten-

der alterar o número máximo de entradas, precisará de mudar esta referência para 1000 — poderia ser substituída por uma variável, desde que você se lembre de definir essa variável quando o programa for inicializado, e de a salvaguardar quando for armazenado um determinado ficheiro.

Linhas 12220-12260: são estas as linhas que pedem a introdução dos itens individuais para a nova entrada. O nome para cada item é retirado do quadro ITEM$, o qual foi estabelecido no módulo de inicialização. Cada item será introduzido sob o nome Q$ e a entrada de novos itens terminará se Q$ se transformar em « \ », em qualquer momento. Se a entrada não for « \ », o item será adicionado a TE$, que conterá toda a entrada, sendo acrescentado um carácter « \ » ao final do item. O objectivo do carácter « \ » não tem rigorosamente nada a ver com a sua anterior utilidade de terminar a entrada de informação; ele apenas se encontra lá como indicador para partes posteriores do programa de onde cada item termina e onde o item seguinte começa. Assim, uma entrada como

SMITH
JOHN
11 ANYSTREET
ANYTOWN

seria armazenada no quadro ARRAY$ como:

SMITH\JOHN\11 ANYSTREET\ANYTOWN\

criando aquilo que se conhece por «cadeia condensada». A razão para a escolha de « \ » é a pouca probabilidade da existência de alguma razão urgente pela qual alguém desejasse vê-lo incluído numa entrada — se, no entanto, desejar incluí-lo, escolha outro carácter separador. Note que todos os itens introduzidos são traduzidos em maiúsculas. Mais tarde, verá que o mesmo acontece quando se faz a pesquisa dos itens no ficheiro. Isto evita que, inadvertidamente, a entrada de letras maiúsculas ou minúsculas torne difícil encontrar um item no ficheiro. Se precisar *obrigatoriamente* de utilizar maiúsculas e minúsculas conjuntamente, então estas provisões podem ser retiradas sem transtorno para a operatividade do programa, embora você deva estar informado de que o *464* considera qualquer letra minúscula como posterior, alfabeticamente, a uma letra maiúscula, o que pode tornar a ordem do seu ficheiro bastante estranha.

Ensaio

Estamos agora em posição de ensaiar os últimos três módulos introduzidos. Execute o programa e estabeleça um ficheiro com dois itens por entrada, denominados «ONE» e «TWO» (um e dois). Quando terminar a inicialização do programa e passar ao *menu* principal, especifique a opção 3. Será agora confrontado com a visualização criada pelo corrente módulo e solicitado a introduzir o item «ONE». Introduza «AA1». A solicitação repetir-se-á para «TWO» e você deverá introduzir «AA2». Repita o processo para «DD1», «DD2», «CC1», «CC2», «BB1», «BB2»,

em solicitações sucessivas. Introduza agora « \ » e regressará ao *menu* principal. Seleccione a opção 6 para parar o programa. Introduza agora:

FOR I = 0 TO 3:? ARRAY$(I):NEXT[ENTER]

e deverá ver:

AA1\AA2
BB1\BB2
CC1\CC2
DD1\DD2

Se tudo funcionou correctamente, talvez deseje começar de novo o programa com GOTO 10000, utilizando a opção 3 do *menu* para armazenar os dados colocados em fita. Isto tornará os ensaios subsequentes menos fatigantes.

Módulo 4.1.10: Identificação de itens numa só entrada

Antes de podermos passar aos módulos que permitem a retirada de dados do ficheiro e sua manipulação, precisamos de introduzir este, cuja tarefa é pesquisar através de uma entrada e registar a posição de cada carácter « \ », ou, melhor, o fim de cada item da entrada. Os valores são armazenados no quadro PTR e relacionar-se-ão apenas com a entrada corrente. Quando for necessária outra entrada, este módulo terá de ser chamado novamente, para analisar.

Módulo 4.1.10: linhas 19000-19080

```
19000 REM ******************************
19010 REM Analyse Record
19020 REM ******************************
19030 pp=0
19040 FOR i=0 TO x-1
19050 ptr(i)=INSTR(pp+1,array$(s1),"\")
19060 pp=ptr(i)
19070 NEXT i
19080 RETURN
```

Comentário

Linha 19050: além do módulo de pesquisa dicotómica, em que a variável SS é utilizada para indicar a entrega que estiver a ser examinada, o resto do programa faz uso da variável S1 para registar, em ARRAY$, a posição da entrada corrente. O efeito desta linha consiste em procurar cada ocorrência do carácter « \ », começando um carácter depois do local onde o último foi encontrado.

Módulo 4.1.11: Pesquisa de itens no ficheiro

Podemos agora passar ao módulo que torna o programa *útil,* permitindo a recolha dos dados que foram armazenados. O corrente módulo permitirá a recolha de entradas, segundo um de quatro métodos:

1) Um por um, ordenadamente, a partir da posição corrente.
2) Saltando para a frente, ou para trás, um número especificado de itens.
3) Introduzindo um item-chave — o primeiro item da entrada — para uma pesquisa rápida.
4) Procurando qualquer ocorrência de uma combinação de caracteres, onde quer que ela se encontre dentro de uma entrada.

Módulo 4.1.11: linhas 17000-17430

```
17000 REM ******************************
17010 REM Search
17020 REM ******************************
17030 s1=0:CLS:PRINT TAB(17);"SEARCH":PR
INT TAB(17);"======":PRINT
17040 IF it=0 THEN PRINT"*** NO DATA ENT
ERRED YET ***" ELSE GOTO 17070
17050 FOR i=1 TO 2000:NEXT i
17060 RETURN
17070 PRINT"COMMANDS AVAILABLE:"
17080 PRINT:PRINT"Input item for normal
search"
17090 PRINT"Precede with '*' for initial
 search"
17100 PEN 2:PRINT"ENTER";
17110 PEN 1:PRINT" for first item on fil
e"
17120 te$="":PRINT:INPUT"Input search co
mmand: ",te$:te$=UPPER$(te$)
17130 IF LEFT$(te$,1)="*" THEN te$=MID$(
te$,2):GOSUB 13000:te$="":s1=ss
17140 p$="C":WHILE p$="C"
17150 IF te$="" THEN GOTO 17210
17160 ff=0:FOR i=s1 TO it-1
17170 pp=INSTR(array$(i),te$)
17180 IF pp<>0 THEN ff=1:s1=i:i=it-1
17190 NEXT i
17200 IF ff=0 THEN RETURN
17210 p$=""
```

```
17220 WHILE p$="" AND it>0
17230 IF s1>it-1 THEN s1=it-1
17240 IF s1<0 THEN s1=0
17250 GOSUB 19000
17260 CLS:PRINT"ENTRY ";s1+1;":":PRINT:p
p=0
17270 FOR i=0 TO x-1
17280 PRINT item$(i);":";MID$(array$(s1)
,pp+1,ptr(i)-pp-1)
17290 pp=ptr(i):NEXT i
17300 PRINT:PRINT"COMMANDS AVAILABLE:"
17310 PRINT:PEN 2:PRINT"ENTER";:PEN 1:PR
INT" for next item"
17320 PRINT"'A' to amend"
17330 PRINT"'C' to continue search"
17340 PRINT"'#' followed by number to mo
ve pointer"
17350 PRINT"'\' to quit function"
17360 PRINT:INPUT"Enter Choice: ",p$
17370 p$=UPPER$(p$):IF p$="" OR p$="C" T
HEN s1=s1+1
17380 IF p$="C" THEN GOTO 17420
17390 IF p$="A" THEN GOSUB 15000:p$=""
17400 IF LEFT$(p$,1)="#" THEN s1=s1+VAL(
MID$(p$,2)):p$=""
17410 WEND
17420 WEND
17430 RETURN
```

Comentário

Linha 17030: S1, como foi mencionado no comentário ao último módulo, é o apontador da entrada corrente, começando em 0 de cada vez que for iniciada uma pesquisa.

Linhas 17040-17060: é gerada uma mensagem de erro se não tiverem sido ainda colocados itens no ficheiro.

Linhas 17070-17120: este é o *menu* de arranque do módulo de pesquisa, que apenas será visto uma vez por cada pesquisa, quando se começa. Utilizando este *menu,* especifica-se o tipo de pesquisa a fazer — se desejar modificar o tipo de pesquisa, será necessário abandonar a corrente pesquisa e recomeçar com este *menu*. O termo «NORMAL SEARCH» (pesquisa normal) indica uma pesquisa de uma dada combinação de caracteres — a primeira entrada retornada será a primeira do ficheiro

que contenha esses caracteres, em qualquer posição. «INITIAL SEARCH» (pesquisa inicial) refere-se a uma pesquisa de uma entrada que *comece com* a combinação de caracteres especificada pelo utilizador. Assim, a introdução de «*SMI» resultaria na descoberta de uma entrada começada pelas letras SMI, embora não necessariamente a primeira. Se a cadeia especificada não estiver presente no início de qualquer entrada, a entrada retornada será a que ocupe o local onde seria inserida, caso fosse introduzida como uma nova entrada.

Quer a pesquisa normal quer a inicial podem estender-se a mais de um item em cada entrada, incluindo um carácter « \ » na cadeia a ser pesquisada. Utilizando esta técnica num ficheiro em que o primeiro item de cada entrada fosse um sobrenome e o segundo item um primeiro nome, uma pesquisa inicial de «SMITH» \ «A» faria aparecer um qualquer SMITH com a inicial «A», embora, de novo, não necessariamente o primeiro.

Numa pesquisa normal, se a cadeia especificada não for encontrada em qualquer das entradas do ficheiro, a execução do programa regressará ao *menu* principal do programa.

Linha 17130: esta linha faz todo o trabalho necessário a uma «pesquisa inicial» retirando o asterisco inicial e chamando, simplesmente, o módulo de pesquisa dicotómica para encontrar a posição correcta.

Linhas 17140-17420: o ciclo principal, que repetirá uma pesquisa normal se o utilizador assim o pedir, num *menu* posterior. Note que a rotina de pesquisa inicial não cai no âmbito deste ciclo, uma vez que não há vantagem em repetir uma pesquisa inicial — o resultado será sempre a mesma entrada.

Linhas 17150-17210: neste ponto da execução do módulo, qualquer entrada de dados deverá corresponder a uma cadeia a ser pesquisada, utilizando a pesquisa normal. Isto é feito de forma muito simples, através da análise da entrada utilizando INSTR. Se for encontrado um item, a variável FF passará a 1, para indicar isso mesmo. A posição será registada em S1 e então, a variável de ciclo, I, será regulada para o seu valor mais elevado, para que o ciclo termine.

Linhas 17220-17410: este ciclo continuará até que P$, a entrada de dados no *menu* de pesquisa subsequente, permaneça numa «cadeia nula». A razão por que o valor de IT tem de ser incluído na condição para o ciclo, é que, dentro deste ciclo, haverá provisão para o apagamento de itens. Se todos os itens forem apagados, precisaremos de sair do ciclo, ou gerar-se-á um erro.

Linhas 17230-17240: dentro do ciclo, o utilizador tem a opção de se deslocar através do ficheiro por números — estas linhas verificam se, em subsequentes passagens pelo ciclo, o apontador não foi deslocado para fora do conjunto válido do número corrente de entradas.

Linhas 17250-17290: após a chamada do módulo anterior, estas linhas fazem uso da informação acerca da posição dos caracteres de separação para imprimir os itens que constituem a entrada corrente, juntamente com o título do item. Por cada

passagem pelo ciclo FOR, são impressos caracteres com uma posição entre o valor da variável PP e um valor contido no quadro PTR. Quando cada lote de caracteres tiver sido impresso, PP passará ao valor corrente no quadro PTR e a variável de ciclo, I, recolherá em PTR o valor seguinte.

Linhas 17300-17360: trata-se do *menu* que aparece quando uma entrada é visualizada. O utilizador tem a opção de passar à entrada seguinte, de chamar a função «AMEND» (correcção) (ainda não introduzida), de continuar a procurar a cadeia anteriormente especificada, de se deslocar no ficheiro uma quantidade especificada de entradas ou de regressar ao *menu* principal do programa.

Linhas 17370-17380: estas duas linhas dizem respeito aos ciclos que se iniciam nas linhas 17140 e 17220. Se a introdução de dados constituir uma «cadeia nula» (isto é, se ENTER for premido), então o ciclo da linha 17220 será repetido, imprimindo a entrada seguinte indicada por S1. Se «C» for introduzido, o ciclo da linha 17140 executará novamente a pesquisa, começando na entrada seguinte.

Linha 17390: é a função AMEND, que ainda não foi introduzida.

Linha 17400: a introdução de um número, precedida do sinal « # », permite ao utilizador avançar ou recuar no ficheiro. A passagem de P$ a uma cadeia nula apenas executa o ciclo da linha 17220, imprimindo a entrada alcançada.

Ensaio

Se salvaguardou previamente a série de quatro entradas feita em ensaios anteriores, execute o programa e volte a carregar as entradas. Especifique a opção 4 do *menu* principal e, então, quando o *menu* de pesquisa aparecer, passe revista às entradas premindo ENTER. Cada entrada deverá ser visualizada em duas linhas, com o nome de item apropriado. Por exemplo:

ONE: AA1
TWO: AA2

Ao atingir a quarta entrada, deverá descobrir que não pode ir mais além utilizando ENTER. Introduza agora « # – 1» e deverá regressar à entrada 3. Continue a regredir — verificará que também não pode sair do princípio do ficheiro.

Introduza « \ » para regressar ao *menu* principal e volte a especificar a opção 4. Desta vez, responda ao *menu* de pesquisa inicial com «CC». A entrada 3 deverá ser visualizada — a única a conter os caracteres «CC». Encontramo-nos agora no segundo *menu* de pesquisa, pelo que deve introduzir «C» para prosseguir a pesquisa, encontrando-se depois de volta ao *menu* principal.

Uma vez mais, escolha a opção 4, introduzindo, desta feita, «2» com alvo de pesquisa. Deverá aparecer a entrada 1, uma vez que contém o carácter «2». Introduza «C» para continuar a pesquisa e deverá ser impressa a entrada 2 — que contém igualmente o carácter 2. Continue a introduzir« C», até que todas as quatro entradas

tenham sido visualizadas e a pesquisa falhe, fazendo regressar o utilizador ao *menu* principal.

Finalmente, escolha a opção 4 a partir do *menu* principal e introduza «*B» como alvo de pesquisa. A entrada 2 deverá ser visualizada — sendo a única entrada a começar com «B». Introduza « \ » para regressar ao *menu* principal e terminar depois o programa.

Tem agora ensaiadas todas as funções de pesquisa.

Módulo 4.1.12: Apagamento de entradas

Os retoques finais ao programa são adicionados pelos dois módulos seguintes, que permitem a alteração e o apagamento de entradas. O módulo DELETE é adicionado primeiro, já que é utilizado sempre que uma entrada é alterada.

Módulo 4.1.12: Linhas 16000-16040

```
16000 REM ******************************
16010 REM Delete Item
16020 REM ******************************
16030 FOR j=s1 TO it-1:array$(j)=array$(
j+1):NEXT j
16040 it=it-1:RETURN
```

Comentário

Linhas 16030-16040: para conseguir o apagamento basta começar no princípio da entrada acima daquela que vai ser apagada, copiando aquela para um espaço abaixo. Quando cada entrada tiver sido deslocada, a contagem dos itens será reduzida em 1.

Ensaio

Execute o programa e volte a carregar os quatro itens a partir da fita. Especifique a opção 4 para parar o programa. Introduza agora:

S1 = 0[ENTER]
GOTO 16000

O programa deverá parar com um erro «Unexpected RETURN». Introduza:

GOTO 11000[ENTER]

e chame depois a opção de pesquisa. Deverá verificar que a entrada AA1 \ AA2 desapareceu do ficheiro.

Módulo 4.1.13: Alteração de entradas

O programa teria uma utilidade limitada para nós se não pudéssemos alterar dados existentes — este módulo preenche essa lacuna.

Módulo 4.1.13: linhas 15000-15190

```
15000 REM ******************************
15010 REM Change Entry
15020 REM ******************************
15030 te$="":pp=0
15040 FOR i=0 TO x-1
15050 CLS:PRINT"Entry ";s1+1;":"
15060 PRINT:PRINT item$(i);":";MID$(arra
y$(s1),pp+1,ptr(i)-pp-1)
15070 PRINT:PRINT"COMMANDS AVAILABLE:"
15080 PRINT:PEN 2:PRINT"ENTER";
15090 PEN 1:PRINT" leaves item unchanged
"
15100 PRINT"Input new item to replace on
e shown"
15110 PRINT"'\D' deletes whole entry"
15120 PRINT"'\' leaves whole entry uncha
nged"
15130 PRINT:INPUT"Which do you require:
",q$
15140 q$=UPPER$(q$):IF q$="\D" THEN GOSU
B 16000:RETURN
15150 IF q$="\" THEN RETURN
15160 IF q$<>"" THEN q$=q$+"\"
15170 IF q$="" THEN q$=MID$(array$(s1),p
p+1,ptr(i)-pp)
15180 pp=ptr(i):te$=te$+q$:NEXT i:GOSUB
16000:GOSUB 13000
15190 s1=ss:GOSUB 14000:RETURN
```

Comentário

Linhas 15040-15180: este ciclo, embora seja muito semelhante ao ciclo que imprime as entradas, no módulo 4.1.8, difere deste na medida em que imprime apenas um item de cada vez.

Linha 15140: a introdução de « \ D» quando um item é visualizado apaga toda a entrada de que esse item faz parte — note que um *item* individual não pode simplesmente ser apagado, já que o número de itens por entrada é fixo.

Linha 15150: a introdução de « \ », em resposta a qualquer item, resulta no retorno da execução ao módulo de pesquisa. Quaisquer alterações feitas aos itens anteriores da entrada serão ignoradas, permanecendo a entrada inalterada.

Linha 15160: se for feita outra introdução, que não « \ D» ou « \ », então ela será interpretada como uma substituição do item visualizado. O separador « \ » é adicionado ao final do item, tal como no módulo dos novos itens.

Linha 15170: ao premir-se ENTER, sem uma entrada de dados, copiar-se-á o item visualizado, sem alterações. Se apenas um item dever ser alterado, bastará premir ENTER para todos os outros. Note a diferença entre a expressão de cadeia aqui e aquela que foi utilizada para imprimir o item, na linha 15060. O « – 1» no final é largado, para que o carácter separador « \ » não seja eliminado.

Linhas 15180-15190: a entrada corrigida é construída em TE$. Quando a entrada estiver completa, a entrada original será apagada do ficheiro. A razão para isto está em que as alterações feitas podem ter alterado a posição correcta da entrada no ficheiro ordenado. Uma vez apagada, a entrada é enviada para o módulo de pesquisa dicotómica e reinserido, tendo a sua posição no ficheiro sido copiada para a variável S1, de modo que o módulo de pesquisa saiba qual o item que deve visualizar, se a posição tiver sido alterada.

Ensaio

Execute o programa e volte a carregar os quatro itens de dados a partir da fita. Chame a opção de pesquisa e prima ENTER para obter a visualização da entrada 1. Introduza agora «A», em resposta ao segundo *menu* de pesquisa. Deverá obter a visualização do primeiro item, «AA1», juntamente com o *menu* AMEND (correcção). Introduza «AAA1» e, quando o segundo item for visualizado, prima ENTER. Deverá agora ter retornado ao módulo de pesquisa e a entrada deverá ser visualizada como:

ONE: AAA1
TWO: AA2

Experimente fazer outras alterações e apagar entradas.

PROGRAMA 4.2: NNÚMERO

Função do programa

Nem tudo o que diz respeito aos ficheiros se relaciona com palavras. Uma das coisas que os microcomputadores fazem melhor é armazenar e manipular valores. O corrente programa, «NNúmero» (abreviatura de «Nome e Número»), permite ao utilizador armazenar os nomes de itens, as unidades em que eles são geralmente medidos e uma quantidade associada. Assim, antes que você diga que não vê utilidade

para um tal programa, imagine o vulgar encarregado de uma loja, ou mesmo a cozinheira doméstica.

O encarregado tem uma quantidade de itens chamados «*stock*». Todos os itens que o compõem têm nomes, chegam em diferentes unidades (caixas, garrafas, sacos, etc.) e todos têm uma quantia muito importante associada — o preço. Para se fazer, portanto, que um microcomputador ajude na aquisição de *stocks* ou emita uma factura, deve lembrar-se destes três factos acerca de cada item. No lar, todos e cada um dos alimentos que comemos têm um nome, apresentam-se em diferentes unidades (colheres, quilos, mão-cheias, etc.) e, se estivermos interessados no seu efeito sobre o nosso peso, todos eles têm associada uma quantidade, conhecida por «calorias».

Estes são apenas dois exemplos — muitos mais você pode descobrir, respeitando à importância de poder registar nomes, unidades e quantia associada, para toda uma variedade de itens.

O objectivo de «Nnúmero» é permitir-lhe criar um dicionário de itens — até um máximo de 200 —, juntamente com as unidades em que são medidos e os valores associados a essas unidades. Com base nesse dicionário, você poderá construir listas de itens que o programa visualizará, encontrando o total das quantidades. O «Nnúmero» é tão fácil de utilizar na adição das calorias das receitas de um dia, como o é no fornecimento de um preço total para um conjunto de bens.

Módulo 4.2.1: Inicialização

Trata-se de um módulo normal, sendo a única coisa digna de nota o facto de o utilizador ser solicitado a especificar o tipo de item com que o programa irá trabalhar, o qual poderá ser um nome como «Item de alimento» ou «Item de *stock*». A frase introduzida será usada no decurso do programa para solicitar ao utilizador a introdução de outro item.

Módulo 4.2.1: linhas 10000-10090

```
10000 REM*******************************
10010 REM Initialise
10020 REM*******************************
10030 MODE 1
10040 DIM array$(1000,1),array(1000),te$
(100,1),te(100) : cu=0 : it=0
10050 PEN 2:PRINT TAB(17);"NNUMBER":PRIN
T TAB(17);"======="
10060 WINDOW 1,40,4,25
10070 PEN 1
10080 INPUT "Load from tape (y/n)";q$
10090 IF LOWER$(q$)="y" THEN GOSUB 22000
 ELSE PRINT:INPUT "Overall name for item
s:",nn$
```

```
                    NNUMBER
                    =======

ITEM:GROMMETTS
Units:3 BOX
Quantity: 8.07
---------------------------------------
ITEM:FLANGES
Units:11 BOX
Quantity: 292.49
---------------------------------------
ITEM:WIDGETS
Units:6 BAG
Quantity: 70.5
---------------------------------------
ITEM:DONGLES
Units:35 PACK
Quantity: 178.85
---------------------------------------

Total: 549.91

Any key to return to menu
```

Fig. 4.2 — *«Nnúmero» em modo de listagem corrente*

Comentário

Linha 10040: o quadro ARRAY$ é utilizado para registar o nome do item e o nome de unidade de cada um dos itens do dicionário. A unidade associada a cada unidade é armazenada no elemento equivalente do quadro ARRAY.

TE$ e TE terão um objectivo semelhante ao ARRAY$ e ARRAY, embora relacionado com a «listagem corrente» extraída do dicionário. A variável CU registará o número de itens da «listagem corrente» — a listagem derivada do dicionário principal. IT, tal como na maioria dos programas do livro, regista o número de itens do ficheiro principal — neste caso o dicionário.

Módulo 4.2.2: «Menu»

Trata-se de um módulo de *menu* normal, sendo a única diferença em relação aos programas anteriores a forma como a linha 11160 evita o acesso a certas funções do programa, antes da introdução de quaisquer dados — ainda que esta facilidade não possa ser utilizada até o próximo módulo ser introduzido.

Módulo 4.2.2: linhas 11000-11210

```
11000 REM*******************************
11010 REM Main Menu
11020 REM*******************************
11030 WHILE z<>8
11040 CLS
11050 PRINT"COMMANDS AVAILABLE:"
```

```
11060 PRINT
11070 PRINT "1) Display current list"
11080 PRINT "2) Input to current list"
11090 PRINT "3) Start new current list"
11100 PRINT "4) Delete from current list
"
11110 PRINT "5) Add to dictionary"
11120 PRINT "6) Examine dictionary items
"
11130 PRINT "7) Save data to tape"
11140 PRINT "8) Stop"
11150 PRINT:INPUT"Enter choice:",z
11160 IF (z=1 OR z=2 OR z=4 OR z=6 OR z=
7) AND it=0 THEN x$="     ********* NO DA
TA YET *********":GOSUB 23000:z=0
11170 ON z GOSUB 12000,13000,14000,20000
,15000,18000,21000
11180 WEND
11190 CLS
11200 LOCATE 11,11:PRINT "PROGRAM TERMIN
ATED"
11210 END
```

Módulo 4.2.3: Mensagens de erro

Este pequeno módulo pode proporcionar uma útil poupança de memória, se um programa for capaz de gerar uma variedade de mensagens de erro para o utilizador. Tudo o que o módulo faz é imprimir uma cadeia designada X$, emitir um «bip» e retornar ao local a partir do qual foi chamado. Se examinar a linha 11160, verá que, para chamar o módulo, X$ é definida e emitido depois um comando GOSUB.

Módulo 4.2.3: linhas 23000-23070

```
23000 REM*******************************
23010 REM Error
23020 REM*******************************
23030 PRINT:PRINT x$
23040 SOUND 1,1000,100
23050 FOR i=1 TO 1500
23060 NEXT i
23070 RETURN
```

Ensaio

Execute o programa e responda apropriadamente às solicitações. Quando for alcançado o *menu,* especifique a opção 1 e deverá obter a mensagem de erro que assinala a inexistência, por enquanto, de dados em memória.

Módulos 4.2.4 e 4.2.5: Ficheiros de dados

São dois módulos normais.

Módulos 4.2.4 e 4.2.5: linhas 21000-22210

```
21000 REM********************************
21010 REM Save To Tape
21020 REM********************************
21030 CLS
21040 PRINT "Save data:":PRINT
21050 INPUT "Name of file:",file$
21060 OPENOUT file$
21070 PRINT #9,nn$
21080 PRINT #9,cu
21090 PRINT #9,it
21100 FOR i=0 TO cu-1
21110 PRINT #9,te$(i,0)
21120 PRINT #9,te$(i,1)
21130 PRINT #9,te(i)
21140 NEXT i
21150 FOR i=0 TO it-1
21160 PRINT #9,array$(i,0)
21170 PRINT #9,array$(i,1)
21180 PRINT #9,array(i)
21190 NEXT i
21200 CLOSEOUT
21210 RETURN
22000 REM********************************
22010 REM Load From Tape
22020 REM********************************
22030 CLS
22040 PRINT "Load data:":PRINT
22050 INPUT "Name of file:",file$
22060 OPENIN file$
22070 INPUT #9,nn$
22080 INPUT #9,cu
22090 INPUT #9,it
```

```
22100 FOR i=0 TO cu-1
22110 INPUT #9,te$(i,0)
22120 INPUT #9,te$(i,1)
22130 INPUT #9,te(i)
22140 NEXT i
22150 FOR i=0 TO it-1
22160 INPUT #9,array$(i,0)
22170 INPUT #9,array$(i,1)
22180 INPUT #9,array(i)
22190 NEXT i
22200 CLOSEIN
22210 RETURN
```

Módulo 4.2.6: Pesquisa dicotómica

Para um comentário completo sobre este módulo, veja o módulo equivalente, no «Unificheiro». A única coisa a salientar acerca do módulo é que a selecção é feita na base do nome do item na coluna zero de ARRAY$.

Módulo 4.2.6: linhas 16000-16110

```
16000 REM*******************************
16010 REM Binary Search
16020 REM*******************************
16030 IF it=0 THEN ss=0:RETURN
16040 po=INT(LOG(it)/LOG(2)):ss=2^po-1
16050 FOR i=po TO 0 STEP -1
16060 ss=ss+2^i*((array$(ss,0)>t1$)-(arr
ay$(ss,0)<t1$))
16070 IF ss<0 THEN ss=0
16080 IF ss>it-1 THEN ss=it-1
16090 NEXT i
16100 IF array$(ss,0)<t1$ THEN ss=ss+1
16110 RETURN
```

Módulo 4.2.7: Inserção de itens no dicionário principal

O princípio deste módulo é exactamente o mesmo do módulo paralelo, no «Unificheiro». A razão por que este módulo é ligeiramente mais longo é porque tem de inserir duas cadeias e um número nos dois quadros, em vez de uma só cadeia.

Módulo 4.2.7: linhas 17000-17110

```
17000 REM*******************************
17010 REM Insert Item
```

```
17020 REM*******************************
17030 FOR i=it TO ss+1 STEP -1
17040 array$(i,0)=array$(i-1,0)
17050 array$(i,1)=array$(i-1,1)
17060 array(i)=array(i-1)
17070 NEXT i
17080 array$(ss,0)=t1$
17090 array$(ss,1)=t2$
17100 array(ss)=nn
17110 RETURN
```

Módulo 4.2.8: Introdução de itens no dicionário

Consideravelmente menos complicado do que o módulo equivalente, no «Unificheiro», este módulo aceita três introduções do utilizador: *a*) o nome do item, *b*) o nome das unidades em que é medido e *c*) a quantidade associada a essas unidades.

Módulo 4.2.8: linhas 15000-15170

```
15000 REM*******************************
15010 REM Extend Dictionary
15020 REM*******************************
15030 WHILE 1
15040 CLS
15050 PRINT "New items for dictionary:":
PRINT
15060 IF it>1000 THEN x$="     *********
NO MORE ROOM *********":GOSUB 23000:RETU
RN
15070 q$="":WHILE LOWER$(q$)<>"y"
15080 PRINT nn$;:INPUT " (name or '\' to
 quit):",t1$
15090 IF t1$="\" THEN RETURN
15100 PRINT "Units";:INPUT ":",t2$
15110 PRINT "Quantity per ";LOWER$(t2$);
:INPUT ":",nn
15120 PRINT:INPUT "Are these correct (y/
n)";q$:PRINT
15130 WEND
15140 GOSUB 16000
15150 GOSUB 17000
15160 it=it+1
15170 WEND
```

Ensaio

Ao fim de bastante tempo, torna-se possível fazer um verdadeiro ensaio do que foi introduzido até aqui.

Execute o programa, especifique que não está a carregar a partir da fita e forneça o nome ITEM em resposta à solicitação para um nome genérico. No *menu* principal, escolha a opção 5, «*Extend Dictionary*» (aumentar o dicionário). Quando o écran «*new items*» (novos itens) aparecer, introduza as seguintes três entradas:

THING1/BOX/10
THING2/BOTTLE/20
THING3/BAG/40

Estes itens não têm qualquer significado especial, apenas servem objectivos de ensaio.

Quando tiver introduzido os itens, retorne ao *menu* principal introduzindo « \ ». Chame agora o módulo do ficheiro de dados (opção 7) para armazenar a informação. Pare o programa com a opção de *menu* 8 e introduza:

```
FOR I = 0 TO 2:?ARRAY$(I,0),ARRAY$(I,1),ARRAY(I):NEXT
[ENTER]
```

Deverá ver o seguinte:

```
THING1    BOX       10
THING2    BOTTLE    20
THING3    BAG       40
```

Execute agora o programa e especifique que *quer* carregar a partir da fita. Dê o nome que forneceu aquando do armazenamento dos dados. Quando a fita tiver terminado, deverá poder executar o mesmo ensaio do conteúdo dos quadros, com o mesmo resultado.

Módulo 4.2.9: A rotina de pesquisa

Tal como no «Unificheiro», este módulo proporciona ao utilizador a oportunidade de se deslocar através do ficheiro de itens do dicionário, pesquisando designados itens ou apagando itens do ficheiro. O módulo é mais simples do que o que foi dado para o «Unificheiro», já que está concebido para pesquisar apenas itens gerais, em vez de combinações de caracteres armazenados algures num item. Para além disto, a estrutura de uma entrada completa em «Nnúmero» é bastante mais simples do que a estrutura de um item no quadro principal do «Unificheiro».

Módulo 4.2.9: linhas 18000-18250

```
18000 REM*******************************
18010 REM User Search
```

```
18020 REM*******************************
18030 ss=0
18040 t1$="":WHILE it>0
18050 CLS:PRINT "Search:":PRINT
18060 PRINT "Item number:";ss+1
18070 PRINT nn$;":";array$(ss,0)
18080 PRINT "Units:";array$(ss,1)
18090 PRINT "Quantity per ";LOWER$(array
$(ss,1));":";array(ss)
18100 PRINT:PRINT "Commands available:":
PRINT
18110 PRINT "Input item to be searched f
or"
18120 PEN 2:PRINT "ENTER";:PEN 1:PRINT "
 for next item"
18130 PRINT "'#' then number to move poi
nter"
18140 PRINT "'d' to delete item"
18150 PRINT "'\' to quit"
18160 PRINT:INPUT "Which do you require"
;t1$
18170 IF t1$="" THEN t1$="#1"
18180 IF t1$="\" THEN RETURN
18190 IF LOWER$(t1$)="d" THEN GOSUB 1900
0:t1$=""
18200 IF LEFT$(t1$,1)="#" THEN ss=ss+VAL
(MID$(t1$,2)):t1$=""
18210 IF t1$<>"" THEN GOSUB 16000
18220 IF ss>it-1 THEN ss=it-1
18230 IF ss<0 THEN ss=0
18240 WEND
18250 RETURN
```

Comentário

Linha 18170: em vez de se fazer uma provisão especial para a introdução de RETURN — isto é, uma cadeia vazia — a variável T1$ é primeiro regulada para #1. Se o utilizador apenas premir RETURN, o conteúdo de T1$ permanecerá inalterado e o item seguinte do ficheiro será visualizado.

Ensaio

Execute apenas o programa, volte a carregar os três itens de dados a partir da fita e, depois, chame a opção 6, «*Display Dictionary*» (visualizar dicionário), a partir

do *menu* principal. Deverá poder paginar através dos itens, para a frente e para trás, utilizando um número precedido de « # », tal como no «Unificheiro». Deverá também poder recuperar um item introduzindo o nome desse item — não o nome da unidade.

Módulo 4.2.10: Apagamento de um item

É o equivalente directo do módulo de apagamento, no «Unificheiro».

Módulo 4.2.10: linhas 19000-19090

```
19000 REM*******************************
19010 REM Delete
19020 REM*******************************
19030 FOR i=ss TO it-2
19040 array$(i,0)=array$(i+1,0)
19050 array$(i,1)=array$(i+1,1)
19060 array(i)=array(i+1)
19070 NEXT i
19080 it=it-1
19090 RETURN
```

Ensaio

Execute o programa, volte a carregar os dados, chame a opção 7 a partir do *menu* principal e introduza «D» contra uma das entradas. Verificará que a entrada é removida do ficheiro.

Módulo 4.2.11: Cópia de itens para a listagem corrente

O objectivo de «Nnúmero» não é apenas manter um dicionário de itens e das suas quantias associadas, mas utilizar esse dicionário como base para a construção de listagens temporárias. Os módulos que se seguem são, portanto, destinados a permitir ao utilizador acrescentar itens à «listagem corrente», visualizar essa listagem, apagar itens singulares ou apagar toda a listagem numa só operação. Este módulo permite a cópia de itens a partir do dicionário principal, para a listagem «corrente».

Módulo 4.2.11: linhas 13000-13210

```
13000 REM*******************************
13010 REM Extend Current List
13020 REM*******************************
13030 WHILE 1
13040 CLS
```

```
13050 PRINT "Additions to current list:"
:PRINT
13060 IF cu>100 THEN x$="     ***** CURRE
NT LIST NOW FULL *****":GOSUB 22000:RETU
RN
13070 PRINT nn$;:INPUT " ('\' to quit):"
,t1$
13080 IF t1$="\" THEN RETURN
13090 known=0:GOSUB 16000:IF array$(ss,0
)=t1$ THEN known=1
13100 IF known=0 THEN x$="**** "+UPPER$(
nn$)+" UNKNOWN, PLEASE CHECK ****":GOSUB
 23000:RETURN
13110 PRINT:PRINT "Units:";array$(ss,1)
13120 INPUT "Quantity:",q
13130 PRINT: INPUT "Are these correct (y
/n)";q$
13140 WHILE LOWER$(q$)="y"
13150 te$(cu,0)=array$(ss,0)
13160 te$(cu,1)=MID$(STR$(q),2)+" "+arra
y$(ss,1)
13170 te(cu)=q*array(ss)
13180 cu=cu+1
13190 q$=""
13200 WEND
13210 WEND
```

Comentário

Linhas 13090-13100: estas linhas efectuam uma verificação, para ver se o item introduzido pelo utilizador, que deve ser colocado na listagem corrente, está presente no dicionário principal. Isto é feito através da chamada do módulo de pesquisa dicotómica, para obter a posição em que o item deverá ser inserido no dicionário. O actual conteúdo do dicionário, neste ponto, é então comparado com o que o utilizador introduziu. Se o item introduzido pelo utilizador estiver incluído no dicionário, então os dois itens serão os mesmos. Caso contrário, será impressa uma mensagem de erro, terminando o módulo.

Linha 13110: depois de ter sido encontrado o item no dicionário, o módulo imprime as unidades em que aquele é, normalmente, medido e pergunta quantas destas unidades devem ser incluídas.

Linhas 13120-13170: são introduzidas as quantias reais. O utilizador é então solicitado a confirmar a veracidade da entrada antes de esta ser acrescentada à listagem

corrente, contida nas variáveis TE$ e TE. Note que a quantidade armazenada em TE não é a quantidade por unidade retirada do dicionário, mas a quantidade *total* para o número de unidades especificadas pelo utilizador.

Ensaio

Execute o programa e volte a carregar os dados a partir da fita. Chame a opção 2 do *menu* principal. Introduza os seguintes itens e números de unidades:

THING1,1
THING2,2
THING3,3

Tente agora que o módulo aceite «THING4», que não se encontra presente no dicionário. Deverá receber uma mensagem de erro, que lhe pedirá para verificar o nome do item. Antes de fazer algo mais, chame a opção 7 — esta armazenará a listagem corrente que você acabou de criar, juntamente com o dicionário principal. Finalmente, escolha a opção 8 para parar o programa.

Introduza agora a seguinte linha, em modo directo:

```
FOR I = 0 TO 2:?TE$(I,0),TE0$(I,1),TE(I):NEXT[ENTER]
```

Deverá ver o que segue:

```
THING1    1 BOX        10
THING2    2 BOTTLE     40
THING3    3 BAG        120
```

Módulo 4.2.12: Visualização da listagem corrente

O único objectivo deste módulo é o de imprimir as entradas que constituem a corrente listagem, uma por uma, no écran. Depois de cada entrada, o utilizador é solicitado a premir uma tecla antes da visualização da entrada seguinte. Isto porque a listagem será, normalmente, mais longa do que o próprio écran e o utilizador não pretenderá que a listagem desenrole para fora do écran mais depressa do que consegue lê-la. No final da listagem, é dado o total das quantidades associadas ao conteúdo da listagem corrente.

Módulo 4.2.12: linhas 12000-12160

```
12000 REM*******************************
12010 REM Display Current List
12020 REM*******************************
12030 IF cu=0 THEN RETURN
12040 CLS:ct=0
12050 FOR i=0 TO cu-1
12060 PRINT nn$;":";te$(i,0)
```

```
12070 PRINT "Units:";te$(i,1)
12080 PRINT "Quantity:";te(i)
12090 PRINT "----------------------------
------------"
12100 WHILE INKEY$="":WEND
12110 ct=ct+te(i)
12120 NEXT i
12130 PRINT:PRINT "Total:";ct
12140 PRINT:PRINT "Any key to return to
menu"
12150 WHILE INKEY$="":WEND
12160 RETURN
```

Módulo 4.2.13: Apagamento de itens da listagem corrente

Trata-se de uma versão muito mais simplificada do tipo de módulo de pesquisa utilizado para o dicionário principal. Ela permite ao utilizador avançar através da listagem corrente, um item de cada vez, e apagar qualquer item introduzindo «D» junto a ele. O processo pode ser terminado em qualquer altura introduzindo « \ ».

Módulo 4.2.13: linhas 20000-20250

```
20000 REM*******************************
20010 REM Current List Deletions
20020 REM*******************************
20030 i=0
20040 WHILE i<cu
20050 CLS
20060 PRINT "Current list deletions:":PR
INT
20070 PRINT te$(i,0)
20080 PRINT te$(i,1)
20090 PRINT:PRINT "Commands available:":
PRINT
20100 PEN 2:PRINT "ENTER";:PEN 1:PRINT "
for next item"
20110 PRINT "'d' to delete"
20120 PRINT "'\' to quit"
20130 PRINT:INPUT "Which do you require"
;q$
20140 IF q$="\" THEN RETURN
20150 IF LOWER$(q$)<>"d" THEN i=i+1
20160 WHILE LOWER$(q$)="d"
20170 FOR j=i TO cu-1
```

```
20180 te$(j,0)=te$(j+1,0)
20190 te$(j,1)=te$(j+1,1)
20200 te(j)=te(j+1)
20210 NEXT j
20220 cu=cu-1
20230 q$="":WEND
20240 WEND
20250 RETURN
```

Comentário

Linhas 20150-20230: a variável que controla qual dos itens deve ser visualizado é «I». Note que ela *não* será aumentada se «D» for introduzido. «I» apontará para o item a ser apagado e o processo de apagamento copiará o item seguinte para essa posição abaixo.

Ensaio

Execute o programa e volte a carregar o ficheiro de dados que contém a listagem corrente, a qual foi salvaguardada num ensaio anterior. Chame a opção 4, «*Delete from Current List*» (apagamento na listagem corrente), a partir do *menu* principal. Deverá agora poder percorrer os três itens da listagem corrente e apagar um deles — o facto de ele ter sido apagado pode ser verificado visualizando a listagem corrente.

Módulo 4.2.14: Inicialização da listagem corrente

Para muitas aplicações, será necessário construir uma listagem corrente, obter o total envolvido e, depois, passar rapidamente a uma nova listagem. Este módulo simples limpa o conteúdo da listagem corrente numa só operação.

Módulo 4.2.14: linhas 14000-14090

```
14000 REM*******************************
14010 REM Initialise Current List
14020 REM*******************************
14030 FOR i=0 TO cu-1
14040 te$(i,0)=""
14050 te$(i,1)=""
14060 te(i)=0
14070 NEXT i
14080 cu=0
14090 RETURN
```

Ensaio

Execute o programa, volte a carregar o ficheiro de dados que inclui a listagem corrente e especifique a opção 3, «*Start Fresh List*» (iniciar nova listagem), a partir do *menu* principal. O *menu* principal deverá ser quase instantaneamente reimpresso. Experimente agora chamar a opção 1, a partir do *menu* principal. Uma vez mais, tudo o que acontece é que o *menu* principal se reimprime a ele próprio — o módulo de visualização foi chamado, mas, como não existe nada para ser visualizado, a execução retorna imediatamente.

Se este ensaio for completado satisfatoriamente, o programa estará completo e pronto a ser utilizado.

PROGRAMA 4.3: TEXTO

Função do programa

Uma das aplicações mais fascinantes do computador moderno é o processador de texto. Utilizando um processador de texto moderno e sofisticado, é eliminado muito do trabalho fastidioso que sempre acompanhou a escrita, de qualquer tipo, e mesmo documentos complexos transformam-se em tarefas realtivamente simples. Infelizmente, no entanto, um programa que execute um processador de texto com todas as especificações pode utilizar mais memória do que a disponível ou um micro pessoal ainda mais potente do que o *464*. Mais importante ainda é que, por rápido que o BASIC do *464* seja, a velocidade a que uma aplicação complexa, como é o caso do processamento de texto, precisa de ser executada, em condições normais, necessita de programação directa em código-máquina, a obscura linguagem compreendida pela pastilha *(chip)* de processamento Z80 do *464*.

O programa fornecido abaixo, «Texto», não pode ser laureado com o título de «processador de texto», mas, ainda assim, permite ao utilizador tratar o *464* como uma forma mais flexível de máquina de escrever, compondo texto, alterando-o, apagando linhas ou palavras, deslocando linhas e imprimido-as numa impressora compatível. Se procura algo para gerir um escritório, então não é aqui que vai encontrá-lo. Mas, se precisa de algo para compor pequenos documentos, então «Texto» é uma útil ferramenta para executar esse trabalho — de facto, é mesmo um dos meus maiores prazeres, como escritor, quando recebo cartas de leitores de anteriores livros, referindo que a carta foi composta e impressa utilizando anteriores versões do «Texto».

```
                                        123 Anystreet
                                        Anytown
                                        Anyshire
                                        Tel. 01 234 5678

Dear Mr. Bloggs,

Thank you for your letter with regard to developing Texted as a
commercial alternative to Wordstar.  I think it has some way to go
yet.

As you can see, the facilities are quite limited.  Nevertheless it
does the job.

I look forward to hearing from you further,


Yours sincerely,

David Lawrence
```

Fig. 4.3: *Carta-demonstração composta e impressa utilizando «Texto».*

Novos conceitos apresentados neste programa incluem:

1) Cursores controlados pelo programa e introdução de texto.
2) Manipulação de material em vastas *arrays* em cadeia.
3) Saída para uma impressora de texto complexo.

Módulo 4.3.1: Inicialização

Trata-se de um módulo sem complexidade, em que nada há para salientar excepto a utilização de uma, ou duas, das variáveis declaradas.

Módulo 4.3.1: linhas 11000-11100

```
11000 REM*******************************
11010 REM Initialise
11020 REM*******************************
11030 in=1
11040 DIM text$(100):ll=1:pl=1
11050 text$(0)=STRING$(32,CHR$(245))
11060 text$(1)=STRING$(32,CHR$(244))
11070 a$=" "
11080 INPUT "Do you wish to load from ta
pe (y/n)";q$
11090 IF LOWER$(q$)="y" THEN GOSUB 19000
11100 RETURN
```

Comentário

Linha 11040: o quadro TEXT$ será utilizado para conter o corpo principal do texto introduzido.

Linhas 11050 e 11060: estas duas linhas de símbolos gráficos constituem dois marcadores, que aparecerão no écran sempre que o topo ou o fundo do bloco de texto for encontrado.

Módulo 4.3.2: Introdução de caracteres

A primeira tarefa principal do programa será aceitar informações a partir do teclado e *fazer* algo com elas, sendo essa a tarefa do corrente módulo. A maioria das linhas do módulo estão relacionadas com a introdução de caracteres de «comando» especiais, que dizem ao «Texto» para executar uma função específica, tal como apagar um carácter, embora haja algumas linhas familiares do capítulo de gráficos, quando é necessário um cursor móvel.

A função geral do módulo é permitir ao utilizador introduzir texto na linha 23 do écran, editando-a à medida que se vai avançando. Módulos posteriores pegarão nestas linhas e incorporá-las-ão no corpo principal do texto.

Deverá também notar que este é um dos poucos programas do livro que têm separações entre as linhas, com o objectivo de uma maior clareza. Isto deve-se a que, embora curto, o programa utiliza uma tal massa de nomes de variáveis e de curtos ciclos que é muito difícil seguir o fluxo sem um guia visual. Tal como em programas anteriores, que foram apresentados desta forma, as linhas programáticas contendo apenas dois pontos (:) podem ser omitidas, se assim o desejar.

Módulo 4.3.2: linhas 12000-12430

```
12000 REM*********************************
12010 REM Edit Line
12020 REM*********************************
12030 :
12040 :
12050 WHILE 1
12060 :
12070 :
12080 t$="":WHILE t$=""
12090 LOCATE p-40*INT(p/40)+1,23+INT(p/4
0):PEN 1:PRINT CHR$(143)
12100 FOR tt=1 TO 20:NEXT tt
12110 LOCATE p-40*INT(p/40)+1,23+INT(p/4
0):PEN 2:PRINT MID$(a$,p+1,1)
12120 t$=INKEY$
12130 WEND
```

```
12140 :
12150 :
12160 IF t$=CHR$(13) OR (LEN(a$)>40 AND
t$=" ") THEN t$="":GOSUB 13000
12170 IF t$="^" THEN GOSUB 15000
12180 :
12190 :
12200 WHILE t$="\"
12210 IF p<>0 THEN t=0 ELSE t=LEN(a$)-1
12220 p=t
12230 t$="":WEND
12240 :
12250 :
12260 WHILE p>0 AND t$=CHR$(127)
12270 a$=LEFT$(a$,p-1)+MID$(a$,p+1)
12280 p=p-1
12290 t$="":WEND
12300 :
12310 :
12320 WHILE t$>=CHR$(32) AND t$<=CHR$(16
3) AND t$<>CHR$(127)
12330 a$=LEFT$(a$,p)+t$+MID$(a$,p+1)
12340 p=p+1
12350 t$="":WEND
12360 :
12370 :
12380 IF t$=CHR$(242) AND p>0 THEN p=p-1
12390 IF t$=CHR$(243) AND p<LEN(a$)-1 TH
EN p=p+1
12400 :
12410 :
12420 LOCATE 1,23:PRINT a$
12430 WEND
```

Comentário

Linhas 12080 e 12130: trata-se de um módulo para intermitência do cursor, que actua em espaço inverso sobre um carácter da linha 23. A posição do cursor é indicada pelo valor da variável P. O cursor alterna com uma letra da cadeia **A$**, a qual é utilizada para introduzir texto.

Linha 12160: esta linha chama a parte subsequente do programa, que incorpora o que é introduzido no corpo principal do texto. Isto pode ser efectuado pelo utilizador premindo ENTER, ou automaticamente, quando a extensão de texto exceda 40 caracteres e for introduzido um espaço.

Linha 12170: premindo a tecla com o símbolo da «seta para cima», na fileira superior de teclas, chama-se um módulo subsequente, que permite a execução de uma variedade de funções de edição sobre o corpo principal do texto.

Linhas 12200-12230: se « \ » for premido, o cursor intermitente deslocar-se-á para o princípio da linha ou, se já lá se encontrar, para o final da linha.

Linhas 12260-12290: desde que o cursor intermitente não esteja posicionado no primeiro carácter, premindo a tecla DEL apaga-se o carácter colocado à esquerda do cursor intermitente. Isto faz-se facilmente reconstruindo a cadeia A$ sem o carácter na posição P.

Linhas 12320-12350: estas linhas aceitam qualquer carácter com um código entre 32 e 163 (ver o «Apêndice» do seu manual para a listagem completa), *excepto* a tecla DEL, que é abordada acima, e torna-o parte da linha de texto que estiver a ser introduzida. O novo carácter é inserido no texto na posição do cursor intermitente.

Linhas 12380-12390: para deslocar o cursor intermitente para a direita ou para a esquerda bastará alterar o valor de P, de acordo com a tecla cursora que for premida. Note como, assim como no capítulo dos gráficos, a definição do nosso próprio cursor nos permite aceitar apenas o que queremos, a partir das teclas de controlo como as das setas cursoras, inserir, apagar, e assim por diante. Nenhuma destas teclas tem qualquer efeito automático, já que ficam fora do campo dos normais caracteres de impressão que o ciclo anterior aceita. Podemos, no entanto, utilizá-las quase como teclas de função, definindo o efeito de premir qualquer delas — todas excepto a tecla ESC, é claro, que fará sempre parar o programa.

Ensaio

Tal como em programas anteriores, precisará de começar a inserir linhas do ciclo de controlo. Por isso, introduza as linhas seguintes:

```
10040 IF IN = 0 THEN GOSUB 11000
10060 GOSUB 12000
```

e depois execute o programa. Especifique que não pretende carregar a partir da fita e deverá ser confrontado com um cursor intermitente no início de uma linha, na parte inferior do écran. Comece agora a escrever — aquilo que escrever deverá aparecer no écran. Deverá poder fazer apagamentos utilizando a tecla DEL, deslocando o cursor intermitente sobre aquilo que escreveu ou saltando do princípio para o fim utilizando « \ ». Se premir ENTER, ou qualquer das setas de controlo, ou ainda se introduzir mais de duas linhas de caracteres seguidas de um espaço, o programa terminará com uma mensagem de erro «Undefined line» (linha indefinida).

Módulo 4.3.3: Colocação de material no corpo principal do texto

Os próximos dois módulos funcionam conjuntamente para permitir o processamento das linhas introduzidas ao fundo do écran, bem como a sua integração no corpo principal do texto contido em TEXT$, que pode conter até 100 linhas, de 32 caracteres cada uma. Quanto ao trabalho, é feito pelo módulo corrente, mas os resultados apenas aparecerão realmente com a entrada do próximo módulo, o qual visualiza uma secção do texto.

Módulo 4.3.3: linhas 13000-13310

```
13000 REM*******************************
13010 REM Insert Line
13020 REM*******************************
13030 IF a$="* " THEN a$=SPACE$(33)
13040 x=0
13050 WHILE a$<>""
13060 :
13070 :
13080 d=0:WHILE LEN(a$)<34 AND d=0
13090 tt$(x)=LEFT$(a$,LEN(a$)-1)
13100 a$=""
13110 d=1:WEND
13120 :
13130 :
13140 WHILE LEN(a$)>33
13150 FOR i=33 TO 1 STEP -1
13160 IF MID$(a$,i,1)<>" " THEN NEXT i:i
=33
13170 tt$(x)=LEFT$(a$,i-1)
13180 a$=MID$(a$,i+1)
13190 x=x+1
13200 WEND
13210 WEND
13220 x=x+1
13230 :
13240 :
13250 FOR i=ll+x TO pl+x STEP -1
13260 text$(i)=text$(i-x)
13270 NEXT i
13280 FOR i=0 TO x-1
13290 text$(pl+i)=tt$(i)
13300 NEXT i
13310 a$=" ":p=0:ll=ll+x:pl=pl+x
```

Comentário

Linha 13030: trata-se de uma simples provisão para facilitar a saída de dados para uma impressora. O problema surge quando se pretende imprimir uma palavra, ou palavras, no lado direito da página. A visualização do écran, quando se consegue criá-la, compõe-se de linhas de 32 caracteres, enquanto uma impressora normal trabalha com linhas de 80 caracteres.

Quando imprime, o «Texto» executa duas linhas do texto em écran conjuntamente, para formar uma linha impressa. A introdução de texto no lado direito do écran resultará, portanto, na impressão do texto a meio da página ou no seu arrastamento até ao fim da linha anterior. Uma forma trabalhosa de ultrapassar este problema é introduzir primeiro uma linha vazia, premindo ENTER quando não existe texto para introduzir. Isto imprime uma linha em branco no papel e assegura que a linha a ser impressa a seguir comece no lado esquerdo da página. Depois disto, escreva uma linha completa de espaços, seguida de outra linha de espaços terminando na palavra a ser colocada do lado direito — a impressora registará todos os espaços e a frase final aparecerá sobre a direita do papel.

No programa final, contudo, a linha completa de espaços pode ser dispensada introduzindo uma linha consistindo num simples «*». O módulo corrente automaticamente traduz isto para uma linha de 33 espaços.

Linha 13040: X é a variável que será utilizada para detectar quantas linhas de texto são necessárias para a adição do novo texto.

Linhas 13050-13210: de cada vez que uma linha de caracteres for acrescentada ao corpo principal do texto, sê-lo-á retirada de A$, o novo texto introduzido através do módulo anterior. Este processo continuará até que não reste nada de A$.

Linhas 13080-13110: o novo texto é, em primeiro lugar, colocado num quadro temporário TT$. Se houver 33 caracteres, ou menos (isto é, < 34), incluindo o espaço que se encontra sempre no final, então o novo texto caberá numa linha de 32 caracteres. Tudo o que é preciso fazer é transferi-lo e, depois, limpar A$.

Linhas 13140-13200: se houver mais 33 caracteres, de modo que o novo texto não caiba numa única linha, o procedimento será pesquisar para trás, a partir do carácter 33, para encontrar o primeiro espaço a marcar o final de uma palavra. A parte de A$ situada até este ponto pode agora ser transferida, assegurando assim que não haverá palavras divididas em duas.

Esta secção de texto é agora removida da frente de A$ e o ciclo principal é executado novamente, exactamente da mesma forma, excepto no que se refere à colocação de mais texto numa linha diferente de TT$.

Linhas 13250-13300: a variável PL regista o local *(PLace)* no corpo principal do texto onde a nova linha deve ser inserida — inicia-se em 0, quando não existe texto, e permanece, normalmente, ao fundo do que foi introduzido. Módulos posteriores permitirão ao utilizador deslocar o ponto de inserção para cima e para baixo, através do corpo principal. O primeiro dos dois ciclos desta secção começa por fazer espaço,

no ponto de inserção, para o número de novas linhas registado por X. Isto faz-se deslocando tudo desde o ponto de inserção até à última linha (LL) para cima, através do número de novas linhas, representado por X. O segundo ciclo copia o conteúdo de TT$ para o espaço criado.

Ensaio

Introduza uma linha suplementar:

14130 RETURN

e depois execute o programa. Quando o cursor intermitente aparecer, introduza:

AAA[ENTER]
BBB[ENTER]
CCC[ENTER]
DDD[ENTER]

Depois de ter feito isto, prima STOP para terminar o programa. Agora, em modo directo, introduza:

FOR I = 0 TO 5:PRINT TEXT$(I):NEXT[ENTER]

Deverá ver uma linha em dente de serra, seguida das quatro linhas de texto que introduziu e de outra linha em dente de serra, marcando o final do texto.

Módulo 4.3.4: Visualização do texto

Este módulo visualiza quinze linhas do corpo principal do texto.

Módulo 4.3.4: linhas 14000-14130

```
14000 REM*******************************
14010 REM Display Text Section
14020 REM*******************************
14030 ss=pl-7
14040 IF ll-pl<8 THEN ss=ll-15
14050 IF ss<0 THEN ss=0
14060 CLS
14070 FOR i=ss TO ss+15
14080 PEN 2
14090 PRINT TEXT$(I)
14100 IF i=pl-1 THEN PEN 1:PRINT ">"
14110 NEXT i
14120 LOCATE 1,23:PEN 2:PRINT a$:PEN 1
14130 RETURN
```

Comentário

Linhas 14030-14050: a variável SS representa a linha inicial da secção a ser impressa (*Section Start,* ou seja, início da secção) e é calculada de modo a ficar, normalmente, oito linhas antes do ponto de inserção corrente. Se o ponto de inserção estiver perto do final do texto, ou mesmo no fim, isto significa que apenas serão visualizadas oito linhas, pelo que a linha 14040 desloca o ponto inicial mais para trás. Finalmente, se houver menos de quinze linhas ou o ponto de inserção se situar no início, o ponto inicial encontrar-se-á então antes do início do texto, pelo que será aumentado para 0.

Linhas 14070-14110: as quinze linhas são agora impressas. Todas as linhas terminam por um ponto e vírgula, para evitar que a posição de impressão se desloque para baixo uma linha. Isto deve-se a que, de outro modo, uma linha completa de 40 caracteres (a primeira e última linhas, consistindo em símbolos gráficos) enviaria o cursor para a linha seguinte, deixando assim uma linha em branco quando aquela passasse para baixo. Para linhas com menos de 40 caracteres, o efeito do ponto e vírgula é cancelado por um PRINT vazio. O único outro requinte consiste na marcação do ponto de inserção com um sinal « > » — quaisquer funções de edição, tais como a adição de linhas, a respectiva cópia ou apagamento, actuarão sobre a linha abaixo do marcador.

Linha 14120: finalmente, como o écran ficou limpo, qualquer coisa que se encontre em A$ será reimpresso ao fundo do écran.

Ensaio

Efectue o mesmo ensaio que para o módulo anterior, sendo a única diferença que, em vez de ser necessário introduzir um comando especial para ver o que se introduziu, cada linha deverá ser colocada no texto principal ao ser premido ENTER.

Módulo 4.3.5: Edição do texto principal

Depois de nos termos possibilitado a edição de novo texto à medida que vai sendo introduzido, precisamos agora de obter algum controlo sobre o próprio texto principal. O módulo corrente dá ao utilizador a possibilidade de deslocar o ponto de inserção ao longo de todo o texto, de voltar a copiar linhas do texto para o fundo do écran, para efectuar alterações, e de apagar linhas existentes.

Módulo 4.3.5: linhas 15000-15320

```
15000 REM*******************************
15010 REM Edit Mode
15020 REM*******************************
15030 WHILE 1
15040 p2=pl-ss
```

```
15050 :
15060 :
15070 t1$="":WHILE t1$=""
15080 LOCATE 1,p2+1:PRINT " ":FOR i=1 TO
 5:NEXT i
15090 LOCATE 1,p2+1:PRINT ">":FOR i=1 TO
 20:NEXT i
15100 t1$=LOWER$(INKEY$)
15110 WEND
15120 :
15130 :
15140 p1=p1+(t1$=CHR$(240))+10*(t1$="u")
:IF p1<1 THEN p1=1
15150 p1=p1-(t1$=CHR$(241))-10*(t1$="d")
:IF p1>11 THEN p1=11
15160 IF t1$=CHR$(13) THEN t$="":RETURN
15170 :
15180 :
15190 WHILE p1<11 AND t1$=CHR$(127)
15200 FOR i=p1 TO 11:text$(i)=text$(i+1)
:NEXT i
15210 l1=l1-1
15220 t1$="":WEND
15230 :
15240 :
15250 IF p1<11 AND t1$="c" THEN a$=text$
(p1)+" "
15260 IF t1$="p" OR t1$="v" THEN GOSUB 1
7000
15270 IF t1$="s" THEN GOSUB 18000
15280 IF t1$="f" THEN GOSUB 16000
15290 :
15300 :
15310 GOSUB 14000
15320 WEND
```

Comentário

Linha 15040: para os objectivos deste módulo, o número de linha no écran do símbolo «>», que indica o ponto de inserção corrente no texto, será armazenado na variável P2.

Linhas 15070-15110: o cursor «>» passa a intermitente, para indicar que o programa se encontra em modo de edição.

Linhas 15140-15150: o cursor de edição pode ser deslocado para cima ou para baixo, através do texto — embora apenas em modo de edição. O cursor deslocar-se-á uma linha se se premir a seta cursora para cima ou para baixo; deslocar-se-á dez linhas para cima ou para baixo se se premir «U» ou «D». Note que, na prática, não é o cursor que se desloca, mas sim o texto que o rodeia, a menos que seja encontrado o princípio ou o fim do ficheiro.

Linha 15160: o modo de edição termina se se premir ENTER.

Linhas 15190-15220: se a tecla DEL for premida durante o modo de edição, apagar-se-á a linha de texto por baixo do cursor de edição.

Linha 15250: se «C» for premido durante o modo de edição, a linha abaixo do cursor de edição será copiada para o fundo do écran, de modo que possa ser manipulada como novo texto.

Linha 15260: se «P» ou «V» forem premidos durante o modo de edição, o módulo subsequente será chamado, enviando o texto para uma impressora ou para o écran, em modo de 80 colunas.

Linha 15270: se «S» for premido durante o modo de edição, o módulo subsequente será chamado, para salvaguardar o texto para fita.

Linha 15280: se «F» for premido durante o modo de edição, o módulo subsequente será chamado a arrumar o texto e a tentar normalizar, até onde for possível, a extensão das linhas.

Ensaio

Com este módulo introduzido, efectue o mesmo ensaio que para o módulo anterior, inserindo quatro grupos de letras.

1) Prima agora «^» e o cursor de edição deverá começar a piscar.
2) Experimente movimentar o cursor de edição para cima e para baixo, utilizando as setas cursoras. A utilização de «U» e «D» deverá fazer deslocar o cursor para o topo ou para o fundo do texto, uma vez que existem menos de dez linhas de texto.
3) Desloque o cursor de edição para a linha acima da última linha de texto. Prima «C», e «DDD» deverá ser copiada para a parte mais inferior do écran.
4) Prima a tecla DEL e a linha «DDD» deverá desaparecer do texto principal.
5) Desloque o cursor de edição para o topo do texto.
6) Prima ENTER para terminar o modo de edição.
7) Quando o cursor intermitente regressar ao fundo do écran, volte a premir ENTER. Deverá obter a inserção da linha «DDD» no início do texto.

Módulo 4.3.6: Formatação do texto

Uma outra possibilidade útil é a de arrumar o corpo principal do texto introduzido. Isto é necessário porque uma das vantagens principais da edição de texto, ou do seu irmão mais velho, o processamento de texto, consiste na capacidade de pegar num texto existente e retirar-lhe frases, ou inserir-lhe novas. Este processo conduz, muitas vezes, a uma aparência esfarrapada do texto e, por essa razão, a maioria dos processadores de texto que não arrumam automaticamente o texto à medida que é inserido permitem ao utilizador solicitar uma função de «formato». Na sua orgânica mais simples, a função de formatação consiste em verificar cada linha e determinar se a primeira palavra da linha seguinte poderá caber no final da linha anterior — se assim for, a palavra é deslocada. O resultado, mesmo que não sejam inseridos espaços para fazer que todas as linhas terminem exactamente no mesmo ponto (justificação), é um texto mais arrumado e agradável. O módulo corrente executa a função de formatação no texto contido em TEXT$, utilizando as técnicas de corte de cadeias apresentadas, em primeiro lugar, no cap. 1.

Módulo 4.3.6: linhas 16000-16310

```
16000 REM*******************************
16010 REM Format Line
16020 REM*******************************
16030 FOR i=1 TO 11-2
16040 d=0:WHILE text$(i)<>"" AND text$(i
+1)<>"" AND text$(i)<>SPACE$(32) AND tex
t$(i+1)<>SPACE$(32) AND d=0
16050 :
16060 :
16070 sp=32-LEN(text$(i))
16080 wrd=INSTR(text$(i+1)," ")
16090 IF wrd=0 THEN wrd=LEN(text$(i+1))+
1
16100 IF wrd>sp THEN d=1
16110 :
16120 :
16130 d2=0:WHILE sp>=wrd AND sp<=LEN(tex
t$(i+1)) AND d2=0
16140 text$(i)=text$(i)+" "+LEFT$(text$(
i+1),wrd-1)
16150 text$(i+1)=MID$(text$(i+1),wrd+1)
16160 d2=1:WEND
16170 :
16180 :
16190 sp=32-LEN(text$(i))
16200 d2=0:WHILE LEN(text$(i+1))<sp AND
d=0 AND d2=0
```

```
16210 text$(i)=text$(i)+" "+text$(i+1)
16220 FOR j=i+1 TO ll
16230 text$(j)=text$(j+1)
16240 NEXT j
16250 ll=ll-1:pl=pl-1
16260 d2=1:WEND
16270 :
16280 :
16290 WEND
16300 NEXT i
16310 RETURN
```

Comentário

Linhas 16030-16300: o processo de formatação inicia-se na primeira linha e trabalha linha a linha através do texto.

Linhas 16040-16290: a operação apenas é efectuada sobre linhas que contenham algo. No final de um parágrafo, por exemplo, pode perfeitamente existir uma linha que tenha menos de 32 caracteres de comprimento — não desejamos ver o início do parágrafo seguinte passar para cima do final desta linha. Isto é ultrapassado com a introdução pelo utilizador de uma linha em branco após o final do parágrafo. O módulo de formatação não acrescenterá esta linha em branco ao final da linha anterior, nem copiará quaisquer palavras da linha seguinte para a linha em branco.

Linha 16070: a variável SP é utilizada para armazenar a quantidade de espaço disponível no final da linha que estiver a ser examinada.

Linha 16080: WRD é utilizado para armazenar a extensão da primeira palavra da linha seguinte, tal como indicado pela presença de um espaço.

Linha 16090: se não houver um espaço na linha seguinte, a extensão de WRD será regulada para a extensão da linha completa.

Linha 16100: se a extensão da primeira palavra da linha seguinte for maior do que o espaço disponível no final da linha corrente, não existe vantagem em ir mais além com a linha corrente, pelo que o ciclo WHILE termina e o ciclo FOR passa à linha seguinte do texto.

Linhas 16130-16160: estas linhas são chamadas à acção se houver espaço suficiente no final da linha corrente para a palavra seguinte, embora não para toda a linha seguinte. O efeito das linhas consiste em copiar a palavra para o final da linha corrente e retirá-la do início da linha seguinte.

Linhas 16190-16260: é muito possível que a linha corrente contenha espaço suficiente para guardar toda a linha seguinte. Neste caso, não se trata apenas de copiar a

linha seguinte para a linha corrente — todo o quadro tem de ter uma linha eliminada, já que, se assim não for, o programa ficará confuso com a instrução, acima mencionada, de que as linhas em branco devem ser deixadas sós.

Ensaio

Efectue o mesmo ensaio do módulo anterior introduzindo os quatro grupos de letras. Quando todos eles forem visualizados como parte do texto principal, prima «^» para introduzir o modo de edição e depois «F». Após uma pausa (que deverá ser bastante acentuada, caso já tenha introduzido muito texto), o texto principal voltará a ser visualizado, mas, desta vez, os quatro grupos de letras deverão ter sido executados conjuntamente numa só linha.

Módulo 4.3.7: Ficheiros de dados

Trata-se de um módulo normal de ficheiro de dados. Um pormenor a salientar é a utilização de LINE INPUT, que assegura que todo o texto que contenha vírgulas e outros caracteres perante os quais INPUT hesite seja devidamente recolhido.

Para além disto, o módulo parece ter esclarecido um problema bastante obscuro, pelo menos em alguns *464*. Este problema assume a forma de uma corrupção, totalmente invisível para o utilizador, da cadeia FI$ no decrso do módulo, evitando que o *464* reconheça o nome do ficheiro em fita como o mesmo que foi introduzido, ainda que ambos, quando visualizados, sejam idênticos. Na nossa própria versão do programa, a linha

por ridícula que pareça, consegue ultrapassar o problema lembrando ao computador o verdadeiro valor da cadeia. Se o programa funcionar sem aquela adição (como deverá), pode ignorar o comentário aqui feito.

Módulo 4.3.7: linhas 18000-19150

```
18000 REM*******************************
18010 REM Save To Tape
18020 REM*******************************
18030 q$="":WHILE LOWER$(q$)<>"y"
18040 CLS:INPUT "File name:",fi$
18050 PRINT "File to be saved is ";fi$
18060 INPUT "Is this correct (y/n)";q$
18070 WEND
18080 OPENOUT fi$
18090 PRINT #9,p1
18100 PRINT #9,l1
18110 FOR i=0 TO l1
18130 PRINT #9,text$(i)
18140 NEXT i
18150 CLOSEOUT
```

```
18160 RETURN
19000 REM*******************************
19010 REM Load From Tape
19020 REM*******************************
19030 q$="":WHILE LOWER$(q$)<>"y"
19040 PRINT:INPUT "File name:",fi$
19050 PRINT "File to be saved is ";fi$
19060 INPUT "Is this correct (y/n)";q$
19070 WEND
19080 OPENIN fi$
19090 INPUT #9,pl
19095 INPUT #9,ll
19100 FOR i=0 TO ll
19110 LINE INPUT #9,text$(i)
19130 NEXT i
19140 CLOSEIN
19150 RETURN
```

Módulo 4.3.8: Módulo de controlo

Trata-se de um módulo de controlo normal.

Módulo 4.3.8: linhas 10000-10060

```
10000 REM*******************************
10010 REM Control Loop
10020 REM*******************************
10030 MODE 1
10040 IF in=0 THEN GOSUB 11000
10050 GOSUB 14000
10060 GOTO 12000
```

Módulo 4.3.9: Impressão do texto

Se possuir uma impressora — e um processador de texto não lhe será de grande utilidade se assim não for —, então este é o módulo que lhe permitirá pegar em tudo o que compôs no écran e colocá-lo em papel. Além disto, o módulo permite ao utilizador visualizar primeiro o texto em modo de 80 colunas, no écran, de modo a assegurar que o texto seja exposto de uma tal forma que possa obter-se um resultado satisfatório na impressora.

Módulo 4.3.9: linhas 17000-17250

```
17000 REM*******************************
17010 REM Output To Printer
```

```
17020 REM*******************************
17030 channel=8
17040 IF t1$="v" THEN channel=0:PEN 1:MO
DE 2
17050 x=1
17060 WHILE x<11
17070 d=0
17080 :
17090 :
17100 IF text$(x)="" THEN PRINT #channel
: d=1
17110 IF d=0 THEN PRINT #channel,SPACE$(
8);text$(x);" ";
17120 x=x+1
17130 :
17140 :
17150 WHILE x<11 AND d=0
17160 PRINT #channel,text$(x)
17170 IF text$(x)="" THEN PRINT #channel
17180 x=x+1
17185 IF channel=0 THEN IF INKEY$="" THE
N GOTO 17185
17190 d=1:WEND
17200 :
17210 :
17220 WEND
17230 PRINT #channel
17240 IF channel=0 THEN IF INKEY$="" THE
N GOTO 17240
17250 MODE 1 : RETURN
```

Comentário

Linhas 17030-17040: este módulo é chamado, quer o utilizador deseje ver o texto em impressora ou no écran, em modo de 80 colunas. A diferença será que, se o utilizador desejar que a visualização se faça no écran, a saída de dados do programa irá para o canal 0 (o écran) e o MODE 2 ficará preparado.

Linha 17050: a variável X é utilizada para registar o progresso do módulo através das linhas de TEXT$.

Linha 17070: a variável D será utilizada para saltar de umas secções do módulo para outras, quando nada mais houver a fazer a uma determinada linha de texto.

Linha 17100: tal como na formatação, não é feita qualquer tentativa para interferir nas linhas em branco — quando uma é encontrada, é empregue uma instrução PRINT # vazia, para deslocar a posição de impressão uma linha para baixo. A variável D passa a 1 para indicar que terminou uma linha de texto.

Linha 17110: se a linha corrente não estiver vazia, será impressa como a primeira metade de uma linha de 64 caracteres, seguida por um espaço. Note-se o ponto e vírgula, o qual assegura que os caracteres impressos a seguir se sucedam sem mais interrupções.

Linhas 17150-17190: é impressa a segunda metade da linha. Se era uma linha vazia, é utilizada uma PRINT # extra para deslocar a impressora um espaço para baixo. Se a saída de dados estiver a ser encaminhada para o écran, então será inserida uma pausa após cada linha impressa, até que o utilizador prima uma tecla. Isto permite a visualização de documentos mais extensos do que um só écran.

Linhas 17240: uma vez mais, se o écran estiver a ser usado para visualização, espera-se que seja premida outra tecla após a conclusão do documento. Serve isto para evitar que o utilizador, inadvertidamente, passe por cima da última linha e perca a visualização do documento.

Ensaio

Se possui uma impressora, assegure-se de que ela se encontra devidamente ligada e a funcionar, introduzindo depois algum texto. Prima SHIFT/0 para introduzir o modo de edição e, finalmente, prima «P». O texto deverá passar para a impressora. Se este ensaio for efectuado com sucesso, o programa deverá estar pronto para utilização. No entanto, para o ajudar a dominar melhor o programa, é fornecida a seguir uma tabela dos vários comandos de tecla única que devem ser utilizados.

«Texto»: Tabela dos comandos de tecla única

Em modo de inserção de texto

ENTER	Introduz novo texto no texto principal.
\	Desloca o cursor para o princípio ou fim da linha.
DEL	Apaga o carácter à esquerda do cursor.
CURSOR ARROWS[1]	Deslocam o cursor para a direita ou esquerda.
^	Inserção do modo de edição.

[1] Setas cursoras. *(N. do T.)*

Em modo de edição

ENTER	Termina o modo de edição.
CURSOR ARROWS	Deslocam o cursor de edição uma linha para cima ou para baixo.
U ou D	Deslocam o cursor de edição dez linhas para cima ou para baixo.
DEL	Apaga directamente a linha por baixo do cursor de edição.
C	Copia directamente a linha por baixo do cursor de edição.
P	Imprime o texto principal corrente.
S	Salvaguarda o texto principal corrente para fita.
F	Formata o texto principal.
V	Visualiza o texto em formato de 80 colunas.

PROGRAMA 4.4: MULTIQ

Função do programa

No programa final deste capítulo, viramo-nos para o compensador assunto da educação, visando algum divertimento. Não estou perfeitamente certo de quanto é possível aprender acerca do tema escolhido, utilizando este programa, mas é divertido de utilizar e torna num hábito a resposta a perguntas. E não apenas isto: o programa é uma estrela por direito próprio, já que foi uma anterior versão do «MultiQ» o primeiro programa (pelo menos de acordo com as informações da imprensa) a gerar emissões radiofónicas, na Grã-Bretanha, em que os ouvintes se divertiam respondendo a perguntas.

«MultiQ», tal como o nome sugere, é um programa de aplicação geral que, tanto pode ser um orientador de línguas como, minutos mais tarde, um questionador sobre pontos pouco esclarecidos da história do século XIX. Tudo isto é efectuado através da criação de testes aleatórios de resposta, do tipo cada vez mais utilizado em exames públicos, em que é posta uma questão e fornecidas cinco respostas possíveis, sendo apenas uma delas a resposta correcta. É mantida uma pontuação corrente, que fornece uma determinação dos conhecimentos do utilizador sobre o tema em causa. O trabalho principal cabe, evidentemente, ao programador, já que não só o próprio programa tem de ser introduzido, como há ainda a questão menor da introdução de um corpo de perguntas suficientemente vasto para assegurar um significado coerente dos testes.

Módulo 4.4.1: Inicialização

Tal como acontece com todos os programas de aplicação múltipla deste capítulo, este módulo inclui uma provisão para a descrição do tipo de ficheiro a ser tratado na corrente sessão.

Módulo 4.4.1; linhas 10000-10100

```
10000 REM*******************************
10010 REM Initialise
10020 REM*******************************
10030 DIM name$(1),qu(4),array$(499,1),q
types$(9),ntype(1,9)
10040 right=0:total=0:it=0
10050 MODE 1
10060 PEN 2:PRINT TAB(17);"MULTIQ":PRINT
 TAB(17);"======"
10070 WINDOW 1,40,4,25
10080 PEN 1
10090 INPUT "Load from tape (y/n)";q$
10100 IF LOWER$(q$)="y" THEN GOSUB 23000
 ELSE GOSUB 24000
```

Comentário

Linha 10030: o quadro NAME$ será utilizado para registar os nomes genéricos dados pelo utilizador às perguntas e respostas; QU será utilizado no estabelecimento de testes aleatórios; ARRAY$ conterá o ficheiro principal das perguntas e respostas; QTYPES conterá os nomes de tipos que possam ser atribuídos a perguntas e respostas; NTYPE armazenará o número de cada tipo no ficheiro principal.

```
                MULTIQ
                ======

EVENT:Outbreak of World War 2

DATE:
 1 ) 1937
 2 ) 1941
 3 ) 1938
 4 ) 1940
 5 ) 1939

Which is the right answer (1-5)? 5

**********
*        *
* RIGHT! *
*        *
**********

Any more (y/n)? !
```

Fig. 4.4 — *Parte de um teste montado pelo «MultiQ»*

Módulo 4.4.2: A estrutura do teste

O «MultiQ», como já foi salientado, estabelece testes, isto é, faz perguntas e visualiza respostas possíveis. Estas linhas permitem que o utilizador dê um nome genérico a perguntas e respostas. Se o programa se destinasse a ser utilizado com objectivos de aprendizagem do francês, por exemplo, poderia chamar à pergunta «PALAVRA INGLESA» e à resposta «PALAVRA FRANCESA EQUIVALENTE».

Módulo 4.4.2: linhas 24000-24130

```
24000 REM*******************************
24010 REM Test Structure
24020 REM*******************************
24030 q$="":WHILE LOWER$(q$)<>"y"
24040 CLS
24050 PRINT "Test structure:":PRINT
24060 INPUT "Name for answer:",name$(0)
24070 INPUT "Name for question:",name$(1
)
24080 PRINT:INPUT "Are these correct (y/
n)";q$
24090 WEND
24100 qtype$(0)="No type"
24110 ntypes=1
24120 GOSUB 12000
24130 RETURN
```

Módulo 4.4.3: Mensagens de erro

Trata-se de um módulo normal para impressão de mensagens de erro fornecidas por outras partes do programa.

Módulo 4.4.3: linhas 25000-25070

```
25000 REM*******************************
25010 REM Error
25020 REM*******************************
25030 PRINT:PRINT x$
25040 SOUND 1,1000,100
25050 FOR i=1 TO 1500
25060 NEXT i
25070 RETURN
```

Módulo 4.4.4: O «menu»

Trata-se de um módulo de *menu* normal.

Módulo 4.4.4: linhas 11000-11210

```
11000 REM*******************************
11010 REM Main Menu
11020 REM*******************************
11030 WHILE z<>7
11040 CLS
11050 PRINT"COMMANDS AVAILABLE:"
11060 PRINT
11070 PRINT "1) Input new items"
11080 PRINT "2) Enter new types"
11090 PRINT "3) Search/Delete"
11100 PRINT "4) Generate questions"
11110 PRINT "5) Display or reset score"
11120 PRINT "6) Save data to tape"
11130 PRINT "7) Stop"
11140 PRINT
11150 INPUT"Enter choice: ",z
11160 IF z>2 AND z<7 AND it=0 THEN x$="
   ********* NO DATA YET *********":GOSU
B 25000:z=0
11170 ON z GOSUB 13000,12000,20000,17000
,19000,22000
11180 WEND
11190 CLS
11200 LOCATE 12,11:PRINT "CLASSROOM CLOS
ED"
11210 END
```

Módulo 4.4.5: Estabelecimento de tipos de perguntas

No decurso da introdução de módulos posteriores, descobrirá que o «MultiQ» prevê dois níveis diferentes de dificuldade nos testes que estabelece. Isso é feito com base em tipos de perguntas. Voltando novamente ao exemplo da utilização do programa como orientador da língua francesa, é claramente possível dividir os tipos de palavras visualizadas em diferentes grupos gramaticais, tais como verbos, substantivos, adjectivos e assim por diante. Se for visualizada uma palavra inglesa que seja um verbo, e das cinco possíveis respostas em francês uma for um verbo, o teste será bastante mais fácil do que se todas as cinco respostas possíveis fossem verbos.

O objectivo do corrente módulo é permitir ao utilizador definir até dez tipos, dentro dos quais irão cair as perguntas ou respostas, com a facilidade de marcar uma pergunta com um tipo, logo que seja introduzida. Quando o «MultiQ», mais tarde, chegar a estabelecer um teste, perguntará ao utilizador se pretende que respostas possíveis a perguntas sejam retiradas apenas do mesmo tipo, para as respostas correctas, ou de toda a provisão de respostas.

Módulo 4.4.5: linhas 12000-12150

```
12000 REM*******************************
12010 REM New Types
12020 REM*******************************
12030 q$="":WHILE 1
12040 CLS
12050 PRINT "Types:":PRINT
12060 PRINT "Types so far:":PRINT
12070 FOR i=0 TO ntypes-1
12080 PRINT i+1;") ";qtype$(i)
12090 NEXT i
12100 IF ntypes=10 THEN x$="     **** NO
ROOM FOR MORE TYPES ****":GOSUB 25000:RE
TURN
12110 PRINT:INPUT "Input new type ('\' t
o quit):";q$
12120 IF q$="\" THEN RETURN
12130 qtype$(ntypes)=q$
12140 ntypes=ntypes+1
12150 WEND
```

Ensaio

Deverá agora estar em posição de executar o programa e introduzir até dez tipos de perguntas. Pode confirmar a aceitação dos tipos parando o programa e escrevendo:

FOR I = 0 TO 9:PRINT QTYPE$(I):NEXT I[ENTER]

Módulo 4.4.6: Pesquisa dicotómica

Trata-se de um módulo de pesquisa normal, que trabalha alfabeticamente na base de respostas a perguntas. Note que, tal como descobrirá quando introduzir o módulo dos novos itens, dentro de alguns momentos, cada resposta é precedida por um só carácter (0-9), para indicar de que tipo se trata. O ficheiro será seleccionado, primeiro que tudo, com base nos tipos e, dentro destes, com base na ordem alfabética das respostas.

Módulo 4.4.6: linhas 14000-14110

```
14000 REM*******************************
14010 REM Binary Search
14020 REM*******************************
14030 IF it=0 THEN ss=0:RETURN
14040 po=INT(LOG(it)/LOG(2)):ss=2^po-1
```

```
14050 FOR i=po TO 0 STEP -1
14060 ss=ss+2^i*((array$(ss,0)>t1$)-(arr
ay$(ss,0)<t1$))
14070 IF ss<0 THEN ss=0
14080 IF ss>it-1 THEN ss=it-1
14090 NEXT i
14100 IF array$(ss,0)<t1$ THEN ss=ss+1
14110 RETURN
```

Módulo 4.4.7: Inserção de um item

Trata-se de um módulo de inserção normal.

Módulo 4.4.7: linhas 15000-15110

```
15000 REM*******************************
15010 REM Insert Entry
15020 REM*******************************
15030 FOR i=it TO ss+1 STEP -1
15040 array$(i,0)=array$(i-1,0)
15050 array$(i,1)=array$(i-1,1)
15060 NEXT i
15070 array$(ss,0)=t1$
15080 array$(ss,1)=t2$
15090 it=it+1
15100 GOSUB 16000
15110 RETURN
```

Módulo 4.4.8: Rastreio dos tipos

Introduzimos já o módulo que permite o registo dos tipos, mas precisamos também de dar ao programa a possibilidade de registar as quantidades de cada tipo existente no ficheiro e onde cada grupo de tipos começa.

O registo das quantidades de cada tipo é tratado pelo módulo de introdução, o qual apenas adiciona 1 ao elemento relevante do quadro NTYPE. Este módulo é chamado sempre que é feita uma nova entrada ou apagamento. O seu objectivo é registar no outro lado de NTYPE os totais acumulados de tipos de 0 a 9. As respostas, eventualmente, serão organizadas por tipos dentro do ficheiro principal, pelo que o conhecimento do total de itens que se incluem nos tipos de 0 a 2, por exemplo, indica-nos onde se iniciam os itens sob o grupo 3.

Módulo 4.4.8: linhas 16000-16080

```
16000 REM*******************************
16010 REM Update
16020 REM*******************************
```

```
16030 su=0
16040 FOR i=0 TO 9
16050 ntype(1,i)=su
16060 su=su+ntype(0,i)
16070 NEXT i
16080 RETURN
```

Módulo 4.4.9.: Inserção de um novo item

Trata-se de um módulo sem complexidade, que permite ao utilizador introduzir uma nova pergunta e resposta, ligando-a depois a um tipo.

Módulo 4.4.9: linhas 13000-13300

```
13000 REM*******************************
13010 REM New Items
13020 REM*******************************
13030 WHILE 1
13040 CLS
13050 PRINT "New items:":PRINT
13060 IF it=500 THEN LET x$="     *******
** NO MORE ROOM *********":GOSUB 25000:R
ETURN
13070 PRINT"Enter item specified"
13080 PRINT"'\' to return to main menu"
13090 PRINT
13100 PRINT name$(0);
13110 INPUT ":";t1$
13120 IF t1$="\" THEN RETURN
13130 PRINT name$(1);
13140 INPUT ":";t2$
13150 IF t2$="\" THEN RETURN
13160 PRINT:PRINT "Types:":PRINT
13170 FOR i=0 TO ntypes-1
13180 PRINT i+1;") ";qtype$(i)
13190 NEXT i
13200 PRINT:INPUT "Input number of type:
",t3
13210 t3=t3-1
13220 IF t3<0 OR t3>ntypes THEN t3=0
13230 PRINT:INPUT "Are these correct (y/
n)";q$
13240 WHILE LOWER$(q$)="y"
13250 ntype(0,t3)=ntype(0,t3)+1
```

```
13260 t1$=MID$(STR$(t3)+t1$,2)
13270 GOSUB 14000
13280 GOSUB 15000
13290 q$="":WEND
13300 WEND
```

Comentário

Linha 13250: o elemento do quadro NTYPE que representa o tipo especificado para a pergunta e resposta correntes é aumentado em 1. É sobre este quadro, como vimos, que o módulo de actualização trabalha, no que se refere ao local onde cada grupo começa, dentro do quadro.

Ensaio

Deverá agora poder executar o programa, especificar tipos e, depois, solicitar a opção 1 do *menu,* para iniciar a introdução de perguntas e respostas. Para verificar se os itens estão a ser recebidos correctamente, introduza os seguintes dados:

PERGUNTA *(Question)*	RESPOSTA *(Answer)*	TIPO *(Type)*
Q111	A111	3
Q222	A222	2
Q333	A333	1

e depois abandone o programa. Escreva agora:

FOR I = 0 TO 2:FOR J = 0 TO 1:PRINT ARRAY$(I,J):NEXT J:NEXT I[ENTER]

Deverá ver:

0A333
Q333
1A222
Q222
2A333
Q111

Módulo 4.4.10: Armazenamento de dados

Agora que podem ser introduzidos itens de dados, é altura de introduzir os dois módulos normais que armazenarão e solicitarão itens.

Módulo 4.4.10: linhas 22000-23220

```
22000 REM********************************
22010 REM Save To Tape
22020 REM********************************
22030 CLS
22040 PRINT "Save data:":PRINT
22050 INPUT "Name of file:",file$
22060 OPENOUT file$
22070 PRINT #9,it
22080 PRINT #9,ntypes
22090 FOR i=0 TO 1
22100 PRINT #9,name$(i)
22110 FOR j=0 TO ntypes-1
22120 PRINT #9,ntype(i,j)
22130 NEXT j
22140 FOR j=0 TO it-1
22150 PRINT #9,array$(j,i)
22160 NEXT j
22170 NEXT i
22180 FOR i=0 TO ntypes-1
22190 PRINT #9,qtype$(i)
22200 NEXT i
22210 CLOSEOUT
22220 RETURN
23000 REM********************************
23010 REM Load From Tape
23020 REM********************************
23030 CL
23040 PR  T "Load data:":PRINT
23050 INPUT "Name of file:",file$
23060 OPENIN file$
23070 INPUT #9,it
23080 INPUT #9,ntypes
23090 FOR i=0 TO 1
23100 INPUT #9,name$(i)
23110 FOR j=0 TO ntypes-1
23120 INPUT #9,ntype(i,j)
23130 NEXT j
23140 FOR j=0 TO it-1
23150 INPUT #9,array$(j,i)
23160 NEXT j
23170 NEXT i
23180 FOR i=0 TO ntypes-1
23190 INPUT #9,qtype$(i)
```

```
23200 NEXT i
23210 CLOSEIN
23220 RETURN
```

Módulo 4.4.11: A pesquisa do utilizador

Trata-se de um simples módulo de pesquisa, que permite ao utilizador pesquisar para a frente e para trás através do ficheiro principal, visualizando e, após a inserção do módulo seguinte, apagando itens.

Módulo 4.4.11: linhas 20000-20220

```
20000 REM ******************************
20010 REM Search
20020 REM ******************************
20030 ss=0
20040 t1$="":WHILE t1$<>"\" AND it>0
20050 CLS:PRINT "Search:":PRINT
20060 PRINT "Item number:";ss+1
20070 PRINT name$(0);":";MID$(array$(ss,
0),2)
20080 PRINT name$(1);":";array$(ss,1)
20090 PRINT "Type:";qtype$(VAL(LEFT$(arr
ay$(ss,0),1)))
20100 PRINT:PRINT "Commands available:":
PRINT
20110 PEN 2:PRINT "ENTER";:PEN 1:PRINT "
 for next item"
20120 PRINT "'#' then number to move poi
nter"
20130 PRINT "'d' to delete item"
20140 PRINT "'\' to quit"
20150 PRINT:INPUT "Which do you require"
;t1$
20160 IF t1$="" THEN t1$="#1"
20170 IF LOWER$(t1$)="d" THEN GOSUB 2100
0
20180 IF LEFT$(t1$,1)="#" THEN ss=ss+VAL
(MID$(t1$,2))
20190 IF ss>it-1 THEN ss=it-1
20200 IF ss<0 THEN ss=0
20210 WEND
20220 RETURN
```

Ensaio

Execute o programa e volte a chamar os dados que armazenou em fita. Utilizando o item 3 do *menu,* deverá poder revistar os itens, para trás e para a frente.

Módulo 4.4.12: Apagamento de um item

Trata-se de um módulo normal de apagamento.

Módulo 4.4.12: linhas 21000-21100

```
21000 REM*******************************
21010 REM Delete
21020 REM*******************************
21030 ntype(0,VAL(LEFT$(array$(ss,0),1))
)=ntype(0,VAL(LEFT$(array$(ss,0),1)))-1
21040 FOR i=ss TO it-2
21050 array$(i,0)=array$(i+1,0)
21060 array$(i,1)=array$(i+1,1)
21070 NEXT i
21080 it=it-1
21090 GOSUB 16000
21100 RETURN
```

Ensaio

Siga o mesmo procedimento tomado para o módulo anterior, mas introduza «D» contra um dos itens. Deverá verificar que o item especificado foi apagado.

Módulo 4.4.13: Organização de perguntas

Viramos agora a nossa atenção para os módulos que constituem a novidade do «MultiQ», através da organização dos testes de escolha múltipla. O módulo corrente trata a parte visível do processo, a visualização das perguntas e possíveis respostas, e a escolha feita pelo utilizador quanto à resposta correcta.

Módulo 4.4.13: linhas 17000-17430

```
17000 REM*******************************
17010 REM Questions
17020 REM*******************************
17030 CLS
17040 PRINT "Questions:":PRINT
17050 PRINT "Do you wish answers to be d
rawn from one";
```

```
17060 PRINT "type only (harder) or from
the whole"
17070 PRINT "stock (easier)?"
17080 PRINT:PRINT "1) One type only"
17090 PRINT "2) All types"
17100 PRINT:INPUT "Which:",rq
17110 IF rq<1 OR rq>2 THEN rq=2
17120 q$="":WHILE LOWER$(q$)<>"n"
17130 GOSUB 18000
17140 CLS
17150 PRINT name$(1);":";array$(qu(qpos)
,1)
17160 PRINT:PRINT name$(0);":"
17170 FOR i=0 TO 4
17180 PRINT i+1;") ";MID$(array$(qu(i),0
),2)
17190 NEXT i
17200 ra=0:WHILE ra<1 OR ra>5
17210 PRINT:INPUT "Which is the right an
swer (1-5)";ra
17220 WEND
17230 WHILE ra-1=qpos
17240 PEN 3:PRINT
17250 PRINT "**********"
17260 PRINT "*         *"
17270 PRINT "* RIGHT! *"
17280 PRINT "*         *"
17290 PRINT "**********"
17300 FOR i=800 TO 100 STEP -100
17310 SOUND 1,i,10
17320 NEXT i
17330 PEN 1
17340 right=right+1
17350 ra=0:WEND
17360 WHILE ra-1<>qpos AND ra>0
17370 PRINT:PRINT "Sorry, that's wrong.
The correct answer"
17380 PRINT "was ";MID$(array$(mainq,0),
2);"."
17390 ra=0:WEND
17400 total=total+1
17410 PRINT:INPUT "Any more (y/n)";q$
17420 WEND
17430 RETURN
```

Comentário

Linhas 17040-17100: fizemos já notar que o «MultiQ» é capaz de organizar dois níveis de teste. Estas linhas permitem ao utilizador especificar se pretende retirar respostas possíveis de todo o ficheiro ou do mesmo tipo da resposta correcta.

Linhas 17150-17190: são visualizadas a pergunta e as cinco respostas possíveis, que serão seleccionadas pelo módulo seguinte. As posições das cinco respostas possíveis estão contidas no quadro QU, e a posição da resposta correcta em QU é registada pela variável QPOS.

Linhas 17230-17350: se a resposta escolhida pelo utilizador, representada pela variável RA, corresponder à posição da resposta correcta (QPOS), então a palavra «RIGHT» (certo) será visualizada e soará um pequeno trinado. A variável RIGHT, que regista o número de respostas correctas, será aumentada em 1. Se for dada a resposta errada, o utilizador será informado de que a resposta está errada e qual a resposta correcta.

Linhas 17400: TOTAL, a variável que regista o número total de perguntas feitas, é aumentada em 1.

Módulo 4.4.14: Selecção das perguntas aleatórias

Depois de termos conseguido obter a visualização das perguntas e respostas, viramo-nos agora para a questão bem mais complexa da *selecção* das perguntas e respostas. Antes do comentário pormenorizado, daremos uma vista geral pelo método envolvido.

O que pretendemos é seleccionar uma pergunta e a sua correspondente resposta correcta, preenchendo depois o quadro QU com cinco números, representando as posições das cinco respostas potenciais dentro do ficheiro principal, incluindo a resposta correcta. A pergunta e resposta principais são escolhidas, em primeiro lugar, ao acaso, a partir da globalidade do ficheiro e o número da pergunta no quadro principal é colocado numa posição aleatória, dentro do quadro QU.

Depois de a pergunta principal ter sido colocada em QU, têm agora de ser encontradas quatro respostas alternativas. Dependendo da preferência do utilizador pela forma mais difícil ou mais fácil do teste, as quatro respostas alternativas serão seleccionadas, quer a partir da globalidade do ficheiro principal, quer a partir da secção que contém respostas cujo tipo é o mesmo da resposta principal. As quatro respostas são escolhidas aleatoriamente na secção apropriada do ficheiro, sendo verificado se a mesma resposta não é incluída duas vezes e se não é incluída qualquer resposta que pareça ser idêntica à resposta correcta.

Módulo 4.4.14: linhas 18000-18210

```
18000 REM*******************************
18010 REM Random Select
18020 REM*******************************
```

```
18030 mainq=INT(RND*(it))
18040 ctype=VAL(LEFT$(array$(mainq,0),1)
)
18050 r1=ntype(1,ctype)
18060 r2=ntype(0,ctype)
18070 IF r2<5 OR rq=2 THEN r1=0:r2=it
18080 qpos=INT(RND*5)
18090 FOR i=0 TO 4
18100 duff=mainq
18110 dup=1:WHILE dup=1 AND i<>qpos
18120 dup=0
18130 duff=r1+INT(RND*r2)
18140 IF MID$(array$(duff,0),2)=MID$(arr
ay$(mainq,0),2) THEN dup=1
18150 FOR j=0 TO i-1
18160 IF MID$(array$(duff,0),2)=MID$(arr
ay$(qu(j),0),2) THEN dup=1
18170 NEXT j
18180 WEND
18190 qu(i)=duff
18200 NEXT i
18210 RETURN
```

Comentário

Linhas 18030-18040: a pergunta e resposta principais são seleccionadas e a respectiva posição armazenada em MAINQ. O tipo da resposta é registado pela variável CTYPE.

Linhas 18050-18070: as duas variáveis, R1 e R2, registam o princípio e o fim da parte do ficheiro a partir da qual as perguntas devem ser retiradas. Se o utilizador tiver especificado a parte mais difícil do teste, então R1 e R2 serão reguladas para apontarem para o princípio e fim do grupo de perguntas do mesmo tipo da pergunta principal. Se acontecer haver menos de cinco perguntas nesse grupo, de modo que seja impossível escolher cinco respostas diferentes, ou se o utilizador tiver especificado a forma mais fácil de teste, R1 e R2 serão reguladas para o princípio do ficheiro. Se o utilizador especificar a forma mais difícil de teste, e verificar que o programa não lha proporciona, a razão provável será não existirem respostas suficientes do mesmo tipo da pergunta principal.

Linha 18080: QPOS é a posição da resposta correcta no quadro das cinco respostas possíveis.

Linhas 18090-18200: este ciclo escolhe as quatro respostas alternativas a partir da zona do ficheiro indicada por R1 e R2, estando cada uma temporariamente arma-

zenada na variável DUFF. Dentro do ciclo, são efectuadas verificações comparativas da nova resposta com a resposta correcta, que poderá encontrar-se em qualquer parte de QU, e as respostas previamente colocadas em QU. Note que não bastará verificar se a mesma resposta no ficheiro principal não se encontra duplicada. Duas perguntas de diferentes partes do ficheiro principal podem perfeitamente ter respostas idênticas. No final do ciclo, QU contém as posições de cinco respostas diferentes, dentro do ficheiro principal.

Ensaio

A única forma efectiva de ensaiar estes módulos é introduzindo um corpo suficiente de dados, para permitir a criação de testes. O melhor ensaio curto seria, em primeiro lugar, registar dois tipos de perguntas, intitulados TYPE 1, TYPE 2... TYPE 6. Introduza depois de uma série de perguntas sob a forma Q1, Q2... Q10, com respostas sob a forma A1, A2... A10. O tipo para as primeiras cinco perguntas deverá ser TYPE 1, sendo as restantes perguntas TYPE 2, ... TYPE 6. Isso proporciona um conjunto de perguntas capaz de gerar a forma mais difícil de teste e cinco outros com apenas uma pergunta cada um.

Chame o gerador de perguntas aleatórias e especifique a forma mais difícil de teste. Deverá verificar que pode continuar a responder a perguntas, sendo correctamente informado sobre se as suas respostas estão correctas ou erradas. Quando é escolhida uma pergunta das cinco primeiras, as cinco respostas deverão encontrar-se na gama de 1 a 5. Quando são escolhidas outras perguntas principais, deverá poder ver se as respostas são retiradas da globalidade do ficheiro de 10 perguntas.

Módulo 4.4.15: Cálculo da pontuação

O retoque final a dar ao programa será possibilitar-lhe o cálculo de uma pontuação significativa para o teste. Isto não é tão fácil como parece, visto que não se trata apenas de pegar no número de respostas correctas para fazer uma percentagem do total. Se o utilizador apenas especificar a primeira resposta em cada teste, isso será, em média, a resposta correcta uma vez em cada cinco perguntas. Uma pontuação imediata de 20% pode indicar que o utilizador não tem qualquer pista quanto à resposta correcta. A solução adoptada consiste em subtrair um quinto do total de perguntas (o número que poderia esperar-se acertar por pura sorte) às respostas correctas e expressar esse valor como uma percentagem de quatro quintos do número total de perguntas.

Módulo 4.4.15: linhas 19000-19110

```
19000 REM*******************************
19010 REM Score
19020 REM*******************************
19030 CLS
19040 PRINT "Score:":PRINT
19050 IF total=0 THEN x$="     *********
```

```
NO SCORE YET *********":GOSUB 25000:RETU
RN
19060 PRINT "Total answers:";total
19070 PRINT "Correct answers:";right
19080 PRINT:PRINT "Score:";INT((right-to
tal/5)/(total*0.8)*100+0.5);"%"
19090 PRINT:INPUT "Do you wish to reset
score (y/n)";q$
19100 IF LOWER$(q$)="y" THEN total=0:rig
ht=0
19110 RETURN
```

Ensaio

Execute novamente o ensaio do módulo anterior. Quando tiver respondido a algumas perguntas, regresse ao *menu* principal e chame o módulo de pontuação. Deverá verificar que a pontuação que lhe é dada faz pouco sentido, embora não seja fácil relacioná-la exactamente com o número de respostas correctas dadas. Deverá ainda ser-lhe apresentada a opção de recolocar a pontuação em zero e começar um novo teste a partir do princípio.

Se a execução deste ensaio foi satisfatória, o programa estará pronto a ser utilizado.

CAPÍTULO 5

Questões financeiras

Neste capítulo final, viramos a nossa atenção para um importante aspecto até agora negligenciado: o *464* e o dinheiro. Trata-se de um assunto que não pode, realisticamente, ser ignorado, justificado pela forma soberba como os microcomputadores lidam com os assuntos financeiros. Raramente as somas envolvidas são vastas — ou, se o são, é pouco provável que sejam tratadas por micros de preço acessível — e os cálculos envolvidos são geralmente simples, tratando-se de operações de adição e subtracção, respectivamente quando o dinheiro entra ou sai.

A verdadeira vantagem do microcomputador, no entanto, não consiste apenas no facto de tratar com dinheiro, já que isso também o cérebro humano pode fazer. É na capacidade que o microcomputador tem de armazenar informação que marca pontos, bem como na capacidade de retirar rapidamente a informação e apresentá-la de modo que possa imediatamente ser compreendida. Os dois programas deste capítulo exemplificam tais capacidades, permitindo um deles que o utilizador mantenha vigilância apertada sobre uma conta bancária e o outro permitindo a simples elaboração de um conjunto de contas.

Os dois programas incluídos no capítulo são:

BANQUEIRO: mantém um registo ordenado dos pagamentos a débito e a crédito de uma conta bancária.

CONTABILISTA: compila um conjunto de contas em formato tradicional.

PROGRAMA 5.1: BANQUEIRO

Função do programa

O objectivo deste programa é permitir ao utilizador manter um registo claro e continuamente actualizado de uma única conta bancária, as designações dos pagamentos, a respectiva data e quantias, incluindo a possibilidade de especificar não apenas pagamentos simples, mas também o escalonamento de receitas e despesas, independentemente da irregularidade dos prazos. O programa está concebido para tratar uma conta durante o período de um ano civil.

```
                    JANUARY
-------------------------------------
DAY & DETAILS              ITEM  BALANCE
-------------------------------------
   Balance forward                 0.00
 2 GROCERY                45.67-   45.67-
 4 GAS                    23.45-   69.12-
15 SALARY                567.89   498.77
17 TELEPHONE              51.11-  447.66
19 MORTGAGE              234.00-  213.66
21 GARAGE                 12.34-  201.32
26 ELECTRICITY            34.56-  166.76
```

Fig. 5.1 — *Impressão de écran do «Banqueiro»*

Módulo 5.1.1: Inicialização

Trata-se de um módulo normal de inicialização.

Módulo 5.1.1: linhas 10000-10160

```
10000 REM*******************************
10010 REM Initialisation
10020 REM*******************************
10030 MODE 1
10040 WHILE in=0
10050 in=1:cr$=CHR$(13):
10060 DIM a$(99,1),a(99,1):a(0,1)=999
10070 RESTORE
10080 DIM mo$(11)
10090 FOR i=0 TO 11
10100 READ mo$(i)
10110 NEXT i
10120 DATA January,February,March,April,
May,June
10130 DATA July,August,September,October
,November,December
10140 WEND
10150 INK 0,24:INK 1,3:INK 2,12
10160 WINDOW #1,1,40,16,25
```

Comentário

Linha 10060: o quadro A$ será utilizado para armazenar as designações de pagamentos e uma cadeia especial, mais adiante explicada, que registará os meses em que determinado pagamento deve ser feito. O quadro numérico A armazenará as

quantias de cada pagamento e o dia do mês em que este é feito. A regulação de A(0,1) para 999, um dia de mês impossível, é utilizada pela rotina de selecção posterior para apagar o final do ficheiro.

Linhas 10080-10130: este ciclo lê os nomes dos meses do ano para o quadro MO$.

Módulo 5.1.2: O «menu» do programa

Trata-se de um módulo de *menu* normal, com a possibilidade adicional de o utilizador ser informado sobre se o programa é solicitado a fornecer resultados antes de quaisquer dados terem sido introduzidos.

Módulo 5.1.2: linhas 11000-11210

```
11000 REM*********************************
11010 REM Menu
11020 REM*********************************
11030 CLS:CLS #1:PEN 2:PRINT TAB(18);"BA
NKER";TAB(18);"======":PEN 1:WINDOW 1,40
,4,25
11040 z=0:WHILE z<>6:CLS
11050 PRINT "Commands Available:":PRINT
11060 PRINT "1) New Payments"
11070 PRINT "2) Examine/Delete Payments"
11080 PRINT "3) Print Statement"
11090 PRINT "4) Save File"
11100 PRINT "5) Load File"
11110 PRINT "6) End"
11120 PRINT:INPUT "Which do you require:
",z
11130 WHILE pa=0 AND (z=2 OR z=3 OR z=4)
11140 PRINT:PRINT:PRINT"     ***** NO DAT
A ENTERED YET *****":SOUND 1,1000,100:FO
R i=1 TO 1500:NEXT i
11150 z=0
11160 WEND
11170 ON z GOSUB 12000,13000,14000,15000
,16000
11180 WEND
11190 CLS
11200 LOCATE 11,11:PRINT "CLOSED FOR BUS
INESS"
11210 END
```

Módulo 5.1.3: Introdução de novos itens

Trata-se de um módulo de introdução mais complexo do que aqueles que temos vindo a utilizar, pela simples razão de que as próprias entradas são mais complexas. Para cada item registado é preciso conhecer cinco factos: se o pagamento é um crédito ou um débito (dinheiro recebido ou gasto), a designação do pagamento, a quantia, os meses em que o pagamento é devido e o dia do mês em que o pagamento é efectuado.

Módulo 5.1.3: linhas 12000-12400

```
12000 REM*******************************
12010 REM Enter New Items
12020 REM*******************************
12030 CLS
12040 PRINT "New items:":PRINT
12050 PRINT "1) Credit":PRINT "2) Debit"
12060 PRINT:INPUT "Which do you require:
",cd:cd=cd-1
12070 q$="":WHILE q$="" OR LEN(q$)>18:PR
INT:PRINT "Name of payment (max. 18 char
s)":INPUT ":",q$:WEND
12080 PRINT:INPUT "Amount:",q
12090 en=1
12100 WHILE en
12110 PRINT:INPUT "Months (e.g.01040710)
:",r$:PRINT
12120 en=0:FOR i=1 TO LEN(r$) STEP 2
12130 m=VAL(MID$(r$,i,2))-1
12140 IF m<0 OR m>11 THEN PRINT:PRINT "
   ***** INVALID MONTH INPUT *****":SOUN
D 1,1000,100:en=1:i=LEN(r$):FOR j=1 TO 1
500:NEXT j
12150 IF en=0 THEN PRINT mo$(m);"/";
12160 NEXT i:PRINT:PRINT
12170 WEND
12180 INPUT "Day of payment:",s
12190 PRINT:INPUT "Are these correct (y/
n)";t$
12200 IF LOWER$(t$)<>"y" THEN RETURN
12210 pa=pa+1
12220 j=pa-1
12230 WHILE s<a(ABS(j),1) AND j>=0
12240 FOR k=0 TO 1
12250 a$(j+1,k)=a$(j,k)
```

```
12260 a(j+1,k)=a(j,k)
12270 NEXT k
12280 j=j-1
12290 WEND
12300 j=j+1
12310 a$(j,1)="000000000000"
12320 FOR i=1 TO LEN(r$) STEP 2
12330 m=VAL(MID$(r$,i,2))
12340 a$(j,1)=LEFT$(a$(j,1),m-1)+"1"+RIG
HT$(a$(j,1),12-m)
12350 NEXT i
12360 a$(j,0)=q$
12370 a(j,0)=q
12380 a(j,1)=s
12390 IF cd=1 THEN a(j,0)=-a(j,0)
12400 RETURN
```

Comentário

Linhas 12040-12060: o programa precisa de saber claramente se o item deve ser pago ou recebido, se se trata de um débito ou de um crédito. Isto é registado sob a forma da variável CD (Crédito/Débito).

Linhas 12090-12170: os meses em que o pagamento deve ser feito são introduzidos sob a forma de uma cadeia de números de dois dígitos, sem qualquer separação entre eles. Assim, se fosse necessário efectuar um pagamento trimestral em Fevereiro, Maio, Agosto e Novembro, a entrada de dados seria «02050811», representando os meses 2, 5, 8 e 11. O ciclo FOR que começa na linha 12120 analisa a entrada da cadeia, para se certificar de que, de facto, ela proporciona uma série de valores de meses com sentido e informa o utilizador se for cometido um erro. Como verificação suplementar, o ciclo imprime os nomes dos meses especificados, tal como se encontram registados em MO$, para que o utilizador possa determinar que não são apenas meses com sentido, mas, de facto, aqueles que se pretendem. A rotina da linha 12090 repete-se, caso tenha sido feita a introdução de um mês inválido, já que o ciclo WHILE depende do valor de EN (*Error Number,* ou seja, número errado), ciclo que será activado se for feita uma entrada incorrecta. Note a utilização de uma janela para a entrada de dados relativos a meses, a qual permite o pagamento das linhas específicas envolvidas sem que se perca de vista o que já foi introduzido.

Linha 12210: neste ponto, a informação introduzida foi já verificada e confirmada pelo utilizador, pelo que a variável encarregue de registar o número de itens do ficheiro, PA, é aumentada em 1.

Linhas 12230-12300: o objectivo deste ciclo é colocar o novo item na ordem correcta dos dias, dentro do ficheiro. Em vez de fazer duplicados dos pagamentos que se referem a mais de um mês, o sistema adoptado é o de armazenar todas as entradas

apenas uma vez, por ordem de *dias*. Quando se pretende a instrução para um mês determinado, uma parte posterior do programa passa em revista todas as entradas, verificando cada uma delas, para ver se se situam antes ou depois do mês especificado, necessitando ser tomadas em consideração aquando da determinação do balanço.

Ao ser inserido o novo item, o ciclo das linhas 12230 a 12290 começa pelo valor de dia mais alto (que é o valor fictício 999, inserido no módulo de inicialização) e vai avançando em sentido descendente, até encontrar um pagamento com um valor de dia menor do que o valor de S, o valor de dia do item acabado de introduzir pelo utilizador. Se o ciclo *não* encontrar a posição correcta, desloca o item que acabou de examinar um lugar para cima. Por outras palavras, à medida que o ciclo analisa ao longo do ficheiro, desloca também consigo uma linha sobressalente. Quando a posição correcta é encontrada, já a linha sobressalente se encontra na posição certa. Como acção final, o valor de J, que regista a posição da primeira entrada encontrada e que possuía um pagamento com valor de dia menos do que S, é aumentado em 1, para apontar para a linha sobressalente.

Linhas 12310-12350: a tarefa seguinte consiste em traduzir a lista de meses introduzida para um formato que possa facilmente ser analisado por partes posteriores do programa. O expediente simples adoptado é o de utilizar uma cadeia de 12 zeros, registando meses em que o pagamento *não* deve ser feito, usando depois um ciclo para alterar um zero para um, referente a meses em que o pagamento *deve* ser feito. Assim, no caso do nosso exemplo de pagamentos trimestrais, a cadeia eventual deveria apresentar «010010010010».

Linhas 12360-12390: a nova informação é colocada nos quadros principais. Se a variável CD registar que o item é um débito, a quantia será multiplicada por – 1, tornando-a negativa.

Ensaio

Execute o programa e solicite a opção 1 do *menu*.
Introduza um novo item, tal como segue:

Débito (opção 2 da solicitação)
Nome: TEST
Quantia: 100
Meses: 02050811
Dia: 15

Depois de uma pausa, deverá regressar ao *menu* principal. Pare o programa utilizando a opção 5 do *menu*. Introduza agora:

```
PRINT A$(0,0),A$(0,1),A(0,0),A(0,1)
```

O resultado deverá ser:

```
TEST      010010010010      -100      15
```

Módulo 5.1.4: Visualização da instrução

Embora haja ainda mais módulos a chegar, a tarefa final da parte principal do programa consiste em pegar nos itens que foram introduzidos utilizando o módulo anterior e compilá-los numa instrução para qualquer mês especificado do ano. A instrução incluirá um cálculo do balanço levado a cabo em meses anteriores e visualizará por completo todos os pagamentos do mês, bem como o balanço contínuo criado por cada pagamento.

Módulo 5.1.4: linhas 14000-14290

```
14000 REM*******************************
14010 REM Compile Statement
14020 REM*******************************
14030 sum=0
14040 CLS:PRINT "Statement:":PRINT
14050 q=0:WHILE q<1 OR q>12
14060 PRINT:INPUT "Number of month for s
tatement (1-12):",q
14070 WEND
14080 FOR j=1 TO q-1
14090 FOR i=0 TO pa-1
14100 IF MID$(a$(i,1),j,1)="1" THEN sum=
sum+a(i,0)
14110 NEXT i
14120 NEXT j
14130 PRINT:INPUT "Send statement to pri
nter (y/n)";p$:ch=0:IF LOWER$(p$)="y" TH
EN ch=8
14140 CLS:PRINT #ch,TAB (20-LEN(mo$(q-1)
)/2);UPPER$(mo$(q-1)):PRINT #ch,STRING$(
39,"-")
14150 PRINT #ch,"DAY & DETAILS";TAB(26);
"ITEM  BALANCE"
14160 PRINT #ch,STRING$(39,"-")
14170 PRINT #ch,"   Balance forward";:PE
N 2-SGN(sum+0.001):PRINT #ch,TAB (32);US
ING "####.##-";sum
14180 FOR i=0 TO pa-1
14190 d=0:WHILE MID$(a$(i,1),q,1)="1" AN
D d=0:d=1
14200 PEN 1:PRINT #ch,USING "## &";a(i,1
);a$(i,0);
14210 PEN 2-SGN(a(i,0+0.001)):PRINT #ch,
TAB (23);USING "####.##-";a(i,0);
```

```
14220 sum=sum+a(i,0)
14230 PEN 2-SGN(sum+0.001):PRINT #ch,TAB
 (32);USING "####.##-";sum
14240 WEND
14250 NEXT i:PEN 1
14260 IF ch=8 THEN RETURN
14270 PRINT:PRINT "Any key to continue:"
14280 WHILE INKEY$="":WEND
14290 RETURN
```

Comentário

Linha 14030: a variável SUM será utilizada para guardar o balanço na conta — tanto o balanço efectuado como o balanço após cada item.

Linhas 14080-14120: desde que a instrução não se destine ao primeiro mês, situação em que não haverá lugar à realização de balanço, estes dois ciclos analisam toda a lista de pagamentos uma vez em cada mês precedente do mês registado na instrução. Desta forma, cada pagamento é examinado, para ver se é feito em qualquer dos meses precedentes, caso em que a quantia apropriada é adicionada ao total, em SUM. No final dos dois ciclos, SUM contém o total completo de quaisquer alterações ao balanço desde o início do ano. (A manutenção de um balanço completo, incluindo qualquer quantia, em dinheiro, que se encontrasse na conta no início do ano, pode ser facilmente efectuada através da introdução do balanço do ano transacto como crédito no 1.º de Janeiro.)

Linha 14170: um pormenor a notar acerca da impressão de SUM, neste pronto, é que, se a quantia for negativa, a cor de impressão será alterada para vermelho. Encontrará as mesmas técnicas frequentemente utilizadas nas linhas que se seguem. Note igualmente a utilização do comando PRINT USING, que nos permite impor um formato padrão para o número a ser impresso e assegurar que a instrução será claramente apresentada, com todos os pontos decimais alinhados.

Linhas 14180-14250: este ciclo analisa toda a lista de pagamentos, enquanto o ciclo nas linhas 14190-14240 selecciona apenas aqueles que têm um «1» na posição relevante da cadeia que regista os meses em que o pagamento deve ser feito. Quando um pagamento deve ser feito no mês especificado para a instrução, o ciclo imprime o dia, A(I,1), o nome, A$(I,0), a quantia, A(I,0), e, finalmente, o balanço produzido pelo pagamento, obtido pela adição da quantia ao total anterior em SUM. Os itens de débito são impressos em vermelho, excepto o dia. O écran é mantido em colunas ordenadas, apesar do facto de os valores poderem variar em extensão, através da utilização de TAB, que inicia os itens numa posição padrão do écran, e de PRINT USING.

Ensaio

Introduza os dados do teste que utilizou para o último módulo. Ao regressar ao *menu* principal, especifique a opção 3 para imprimir a instrução.

Comece pelo mês 1 e, quando estiver satisfeiro, volte ao *menu* principal e imprima a instrução para o mês seguinte. Deverá verificar que a maioria das instruções estão em branco, já que a maioria dos meses não têm pagamentos. Os meses de Fevereiro, Maio, Agosto e Novembro, no entanto, têm todos um pagamento, que deverá estar claramente visualizado.

Módulo 5.1.5: Ficheiros de dados

Trata-se de um módulo normal de ficheiro de dados.

Módulo 5.1.5: linhas 15000-16140

```
15000 REM*******************************
15010 REM Save To Tape
15020 REM*******************************
15030 CLS:q$="":WHILE LOWER$(q$)<>"y"
15040 INPUT "Name of file to be saved:",
fi$
15050 PRINT:PRINT "File to be saved is "
;fi$
15060 PRINT:INPUT "Is this correct (y/n)
";q$
15070 WEND
15080 OPENOUT fi$
15090 PRINT #9,pa
15100 FOR i=0 TO pa-1
15110 PRINT #9,a$(i,0);cr$;a$(i,1);cr$;a
(i,0);cr$;a(i,1)
15120 NEXT i
15130 CLOSEOUT
15140 RETURN
16000 REM*******************************
16010 REM Load From Tape
16020 REM*******************************
16030 CLS:q$="":WHILE LOWER$(q$)<>"y"
16040 INPUT "Name of file to be loaded:"
;fi$
16050 PRINT:PRINT "File to be loaded is
";fi$
16060 PRINT:INPUT "Is this correct (y/n)
";q$
```

```
16070 WEND
16080 OPENIN fi$
16090 INPUT #9,pa
16100 FOR i=0 TO pa-1
16110 INPUT #9,a$(i,0),a$(i,1),a(i,0),a(
i,1)
16120 NEXT i
16130 CLOSEIN
16140 RETURN
```

Módulo 5.1.6: Alteração e apagamento de itens

Trata-se, tal como em «Nnúmero», de um módulo de alteração muito simples para o utilizador, funcionando, desta vez, na base de um ciclo FOR que analisa os itens um por um.

Módulo 5.1.6: linhas 13000-13250

```
13000 REM*******************************
13010 REM Examine/Delete Items
13020 REM*******************************
13030 FOR i=0 TO pa-1
13040 CLS
13050 PRINT "Payment:";a$(i,0)
13060 PRINT "Amount:";a(i,0)
13070 PRINT "Months:";
13080 FOR j=1 TO 12
13090 IF MID$(a$(i,1),j,1)="1" THEN PRIN
T mo$(j-1);"/";
13100 NEXT j:PRINT
13110 PRINT "Day of payment:";a(i,1)
13120 PRINT:PRINT "Commands available:":
PRINT
13130 PEN 2:PRINT "ENTER";:PEN 1:PRINT "
 next item"
13140 PRINT "'\' quit"
13150 PRINT "'\D' delete item"
13160 q$="":PRINT:INPUT "Which do you re
quire:",q$
13170 WHILE UPPER$(q$)="\D"
13180 FOR j=i TO pa-1
13190 FOR k=0 TO 1
13200 a$(j,k)=a$(j+1,k):a(j,k)=a(j+1,k)
13210 NEXT k,j
13220 pa=pa-1:q$="\"
```

```
13230 WEND
13240 IF q$="" THEN NEXT i
13250 RETURN
```

Ensaio

Com alguns dados introduzidos e salvaguardados em fita, chame esta opção. Deverá verificar que consegue passar em revista os itens e apagá-los. Se estas funções estiverem disponíveis, então o programa deverá estar perfeitamente operacional.

PROGRAMA 5.2: CONTABILISTA

Função do programa

O segundo programa deste capítulo é mais complexo do que o «Banqueiro». A sua função é manter dois lados de um simples conjunto de contas, apresentando-os em formato tradicional, com alguns itens isolados e outros claramente divididos em grupos representativos de diferentes tipos de despesa. O programa, tal como se encontra correntemente montado, pretende tratar somas até £ 9999.99[1]. Quantias acima desta são calculadas com exactidão, mas adulteram o formato das contas. Se necessitar de itens ou totais de 10 000, ou acima deste valor, será preciso fazer alterações simples aos comandos PRINT USING e ao espaçamento da visualização.

Módulo 5.2.1: Inicialização

Trata-se de um módulo de inicialização normal.

Módulo 5.2.1: linhas 10000-10080

```
10000 REM*******************************
10010 REM Initialise
10020 REM*******************************
10030 CLS
10040 WHILE in=0
10050 in=1:cr$=CHR$(13):
10060 DIM a$(1,99),a(1,99)
10070 WEND
10080 MODE 1
```

[1] Note que se trata de um valor em libras esterlinas, pelo que o valor em escudos acarretará as alterações a seguir mencionadas.

```
                        DEBIT

CAR EXPENSES
  REPAIRS              98.21
  PETROL              198.56
  TYRES                54.00
                      -------

                                   350.77
AMSTRAD CPC 464                    349.00

HOUSEHOLD
  GAS                  45.87
  ELECTRICITY          56.43
  RATES               126.75
                      -------

                                   229.05
HOLIDAY                            364.76
                                   -------
TOTAL:                            1293.58
```

Fig. 5.2 — *Impressão de écran do «Contabilista»*

Comentário

Linha 10060: os dois lados das contas, crédito e débito, incluindo os nomes associados a cada pagamento, são armazenados nos dois lados dos quadros A e A$. Podem ser armazenados até cem itens em ambos os lados, embora esta quantidade possa, evidentemente, ser aumentada com facilidade.

Módulo 5.2.2: O «menu» principal

Trata-se de um módulo de *menu* normal, com protecção contra o acesso a certas funções antes de terem sido introduzidos alguns dados.

Módulo 5.2.2: linhas 11000-11220

```
11000 REM*******************************
11010 REM Menu
11020 REM*******************************
11030 CLS:PEN 2:PRINT TAB(16);"ACCOUNTAN
T";TAB(16);"==========":PEN 1:WINDOW 1,4
0,4,25
11040 z=0:WHILE z<>6:CLS
11050 PRINT "Commands Available:":PRINT
11060 PRINT "1) New Headings"
11070 PRINT "2) Change/Delete Items"
```

```
11080 PRINT "3) Print Accounts"
11090 PRINT "4) Save File"
11100 PRINT "5) Load File"
11110 PRINT "6) End"
11120 PRINT:INPUT "Which do you require:
",z
11130 IF z<4 AND z>0 THEN GOSUB 12000
11140 WHILE c(cd)=0 AND (z=2 OR z=3 OR z
=4)
11150 PRINT:PRINT:PRINT"     ***** NO DAT
A ENTERRED YET *****":SOUND 1,1000,100:F
OR i=1 TO 1500:NEXT i
11160 z=0
11170 WEND
11180 ON z GOSUB 13000,16000,18000,19000
,20000
11190 WEND
11200 CLS
11210 LOCATE 11,11:PRINT "PROGRAM TERMIN
ATED"
11220 END
```

Módulo 5.2.3: Crédito ou débito?

De forma diferente à do «Banqueiro», várias partes deste programa precisam de saber se está a ser especificado um item de crédito ou de débito, pelo que a rotina que solicita esta informação é incluída em módulo separado.

Módulo 5.2.3: linhas 12000-12110

```
12000 REM*******************************
12010 REM Credit or Debit?
12020 REM*******************************
12030 cd=-1:WHILE cd<>0 AND cd<>1
12040 CLS
12050 PRINT "Credit or debit:":PRINT
12060 PRINT "1) Credit":PRINT "2) Debit"
12070 PRINT:INPUT "Which do you require:
",cd:cd=cd-1
12080 cd$="CREDIT"
12090 IF cd=1 THEN cd$="DEBIT"
12100 WEND
12110 RETURN
```

Módulo 5.2.4: O tipo de item

Este módulo não tem, em si mesmo, nada de especial, mas fornece uma pista quanto à razão pela qual este programa está destinado a ser mais longo do que algo parecido com o «Banqueiro». O objectivo do módulo é permitir ao utilizador especificar sob qual de três tipos recai um item a ser introduzido. Os três tipos são:

1) Um item simples *(«A Single Item»):* tudo o que é necessário, neste caso, é o nome do item e a quantia. Quando a conta eventual é impressa, os itens individuais terão os respectivos nomes impressos sobre o lado esquerdo e a quantia associada na coluna principal de valores, sobre a direita.

2) Um cabeçalho principal *(«A Main Heading»):* é este o tipo que permite especificar grupos de itens dentro da conta geral. Se estava a utilizar o programa para preparar a contabilidade doméstica, por exemplo, pode estabelecer «CARRO» como um cabeçalho principal de um grupo de itens que inclua pneus, gasolina, reparações e assim por diante. Na conta eventual, o nome do cabeçalho principal será impresso sobre o lado esquerdo, mas não haverá qualquer quantia impressa junto ao próprio cabeçalho principal.

3) Subtítulos *(«Subheadings»):* tal como foi ilustrado em 2), cada cabeçalho principal pode ter a segui-lo uma lista de itens, os quais constituem um grupo separado. Na contabilidade, os nomes de subtítulos serão impressos sob o cabeçalho relevante, inseridos a partir da esquerda, enquanto a quantia associada a cada subtítulo será impressa sob a esquerda da coluna principal de valores.

Módulo 5.2.4: linhas 13000-13120

```
13000 REM*******************************
13010 REM Input Headings
13020 REM*******************************
13030 CLS
13040 PRINT "New items:":PRINT
13050 PRINT cd$:PRINT
13060 PRINT "Is the item:":PRINT
13070 PRINT "1) A Single Item;"
13080 PRINT "2) A Main Heading or"
13090 PRINT "3) A Sub-Heading"
13100 PRINT:INPUT "(0 to quit)";type
13110 ON type GOSUB 14000,14000,15000
13120 RETURN
```

Ensaio

Após a introdução de todas as partes do programa que não implicam quaisquer cálculos, será melhor, provavelmente, que você execute o programa e ensaie rapida-

mente o *menu*. Se especificar que pretende introduzir um novo item, deverá ser-lhe perguntado se se trata de um crédito ou de um débito, seguindo-se a solicitação para especificar um tipo — embora seja impossível ir mais longe. A outra única função do *menu* que terá algum efeito é a opção 5, que pára o programa. O próprio *menu* deve impedir o utilizador de ter acesso às funções de alteração de dados ou de impressão das contas, já que não foram ainda introduzidos dados.

Módulo 5.2.5: Entrada de itens singulares e de cabeçalhos principais

Dois módulos distintos estão encarregues da introdução de subtítulos, por um lado, e de itens singulares e cabeçalhos principais, por outro. É importante, para a compreensão de partes posteriores do programa, que o utilizador tente seguir a forma como os itens são armazenados e os caracteres indicadores especiais que registam o tipo de item.

Módulo 5.2.5: linhas 14000-14120

```
14000 REM*******************************
14010 REM Single Item or Main Heading
14020 REM*******************************
14030 q=0:q$="":r$=""
14040 PRINT:INPUT "Name of item:",q$
14050 IF type=1 THEN PRINT:INPUT "Amount
 for item:",q
14060 PRINT:INPUT "Is this correct (y/n)
";r$
14070 IF LOWER$(r$)<>"y" THEN PRINT:PRIN
T "     ******** NOT REGISTERED ********"
:SOUND 1,1000,100:FOR i=1 TO 1500:NEXT i
:RETURN
14080 IF type=1 THEN q$="%"+q$ ELSE q$="
*"+q$
14090 a$(cd,c(cd))=q$
14100 a(cd,c(cd))=q
14110 c(cd)=c(cd)+1
14120 RETURN
```

Comentário

Linha 14050: tal como foi mencionado na introdução ao módulo anterior, os cabeçalhos principais não têm associado qualquer valor, pelo que esta linha apenas aceita um valor para itens singulares.

Linha 14080: não existem zonas de armazenamento separadas para os diferentes tipos de itens, além dos lados de crédito e débito dos quadros. Partes posteriores do

programa determinarão o tipo de item consultando um carácter indicador especial junto ao início do nome do item. Aquele será «%» para um item singular e «*» para um cabeçalho principal.

Linhas 14090-14120: já lhe foi apresentada a variável CD, que regista se um item é um crédito ou um débito. Aqui, CD é utilizada para decidir qual o lado dos quadros A e A$ em que o novo item deve ser colocado. Além disto, precisamos de manter um registo do número de itens do lado do crédito e do débito, já que, geralmente, estes serão diferentes. Isto é feito pelo quadro C. O quadro não foi declarado no módulo de inicialização, uma vez que apenas terá dois elementos, C(0) e C(1), correspondentes aos lados do crédito e do débito nos quadros principais. Uma vez mais, o valor de CD é utilizado para indicar qual dos dois elementos de C deve ser usado. Ao aplicar isto, podemos ver que, quando se faz referência a

```
A  ( CD ,  C(CD) )
1     2        3
```

o que isso significa é:

1) Um elemento no quadro numérico A.
2) O lado indicado pelo valor de CD, isto é, um crédito ou um débito.
3) O primeiro elemento vazio desse lado, determinado por aquilo que já se encontra armazenado.

Ensaio

Execute o programa e chame a nova opção de entrada. Especifique que pretende introduzir um cabeçalho principal no lado do crédito e, depois, introduza TEST MAIN para o nome do item — não lhe deverá ser pedido qualquer valor. Chame a opção da nova entrada, uma vez mais, e especifique um item singular do lado do crédito, com o nome TEST e o valor 100. Agora, faça exactamente a mesma coisa, mas do lado do débito das contas.

Pare o programa a partir do *menu* e introduza, em modo directo:

```
PRINT A(0,0),A(1,0),A$(0,0),A$(1,0)[ENTER]
```

Deverá ver:

```
0    0     *TEST MAIN      *TEST MAIN
```

Aplique agora o mesmo procedimento para a linha 1 dos quadros, ou seja, A(0,1), etc.

Deverá ver:

```
100      100   %TEST      %TEST
```

Finalmente, imprima o valor de C(0) e C(1) — ambos deverão ser equivalentes a 2.

Módulo 5.2.6: Introdução de um subtítulo

A questão da introdução de um novo subtítulo não é tão simples como a da introdução de um item singular. Por cada subtítulo introduzido, deve ser feita uma verificação da presença do cabeçalho principal relevante e o item deve ser colocado junto ao seu cabeçalho principal, em vez de ser, simplesmente, arrastado para o final dos itens previamente armazenados.

Módulo 5.2.6: linhas 15000-15120

```
15000 REM*******************************
15010 REM Sub Heading
15020 REM*******************************
15030 PRINT:INPUT "Main heading:",q$
15040 q$="*"+q$
15050 pl=-1:FOR i=0 TO c(cd)-1
15060 IF a$(cd,i)=q$ THEN pl=i+1
15070 NEXT i
15080 IF pl=-1 THEN PRINT:PRINT "     ***
* NO HEADING OF THAT NAME ****":SOUND 1,
1000,100:FOR i=1 TO 1500:NEXT i:RETURN
15090 PRINT:INPUT "Name of sub-heading:"
,q$
15100 PRINT:INPUT "Amount:",q
15110 PRINT:INPUT "Are these correct (y/
n)";r$
15120 IF LOWER$(r$)<>"y" THEN PRINT:PRIN
T "     ******** NOT REGISTERED ********"
:SOUND 1000,100:FOR i=1 TO 1500:NEXT i:R
ETURN
15130 q$="$"+q$
15140 FOR i=c(cd)+1 TO pl+1 STEP -1
15150 a$(cd,i)=a$(cd,i-1)
15160 a(cd,i)=a(cd,i-1)
15170 NEXT i
15180 a$(cd,pl)=q$
15190 a(cd,pl)=q
15200 c(cd)=c(cd)+1
15210 RETURN
```

Comentário

Linhas 15030-15080: é solicitado o nome do cabeçalho principal relevante e feita uma verificação dos itens já dentro do ficheiro, para conclusão sobre a existência

efectiva do cabeçalho — se assim não for, será impressa uma mensagem de erro, e o programa regressa ao *menu*.

Linhas 15140-15200: tal como foi mencionado previamente, o objectivo fundamental de um subtítulo consiste no seu aparecimento na contabilidade principal como parte de um grupo impresso sob o cabeçalho principal relevante. Para que isto se consiga facilmente, o método empregue é o de armazenar o subtítulo no ficheiro junto ao seu cabeçalho principal. A posição do primeiro item a seguir ao cabeçalho principal foi já encontrada pelo ciclo FOR, na linha 15050. Assim, tudo o que resta fazer é deslocar para cima todos os itens a partir desse ponto e colocar o novo item no quadro, na posição correcta.

Ensaio

Introduza os itens especificados para o ensaio ao módulo 5.2.5 e, depois, volte a chamar o módulo das novas entradas, para colocar um novo subtítulo no lado dos créditos, cujo nome será TEST SUB, com um valor de 200. Faça o mesmo para o lado dos débitos.

Introduza o seguinte, em modo directo:

```
FOR I=0TO 2:PRINT A(0,I),A(1,I),A$(0,I),A$(1,I):NEXT I[ENTER]
```

Deverá ver:

```
0       0       *TEST MAIN      *TEST MAIN
200     200     $TEST SUB       $TEST SUB
100     100     %TEST           %TEST
```

Imprima os valores de C(0) e C(1), devendo ambos ser 3.

Módulo 5.2.7: Ficheiros de dados

Posto que os dados para o «Contabilista» são bastante complexos, talvez seja sensato introduzir neste ponto o módulo do ficheiro de dados, de modo a eliminar a necessidade de uma constante reentrada de dados durante os ensaios. Uma vez introduzido o módulo, introduza e salvaguarde os dados especificados para o ensaio ao módulo anterior.

Módulo 5.2.7: linhas 19000-20160

```
19000 REM*******************************
19010 REM Save To Tape
19020 REM*******************************
19030 CLS:q$="":WHILE LOWER$(q$)<>"y"
19040 INPUT "Name of file to be saved:";
fi$
```

```
19050 PRINT:PRINT "File to be saved is "
;fi$
19060 PRINT:INPUT "Is this correct (y/n)
";q$
19070 WEND
19080 OPENOUT fi$
19090 FOR i=0 TO 1
19100 PRINT #9,c(i)
19110 FOR j=0 TO c(i)-1
19120 PRINT #9,a$(i,j);cr$;a(i,j)
19130 NEXT j
19140 NEXT i
19150 CLOSEOUT
19160 RETURN
20000 REM*********************************
20010 REM Load From Tape
20020 REM*********************************
20030 CLS:q$="":WHILE LOWER$(q$)<>"y"
20040 INPUT "Name of file to be loaded:"
;fi$
20050 PRINT:PRINT "File to be loaded is
";fi$
20060 PRINT:INPUT "Is this correct (y/n)
";q$
20070 WEND
20080 OPENIN fi$
20090 FOR i=0 TO 1
20100 INPUT #9,c(i)
20110 FOR j=0 TO c(i)-1
20120 INPUT #9,a$(i,j),a(i,j)
20130 NEXT j
20140 NEXT i
20150 CLOSEIN
20160 RETURN
```

Módulo 5.2.8: Alterações aos itens

Trata-se de um módulo simples, com algumas características adicionais, que contempla o facto de alguns itens não se encontrarem sós, antes fazendo parte de grupos de itens sob um cabeçalho principal comum.

Módulo 5.2.8: linhas 16000-16290

```
16000 REM*********************************
16010 REM Changes and Deletions
```

```
16020 REM*******************************
16030 FOR i=0 TO c(cd)-1
16040 d=0:WHILE d=0:d=1
16050 CLS
16060 PRINT "Change or Delete:":PRINT
16070 IF LEFT$(a$(cd,i),1)<>"$" THEN PRI
NT MID$(a$(cd,i),2);
16080 IF LEFT$(a$(cd,i),1)="*" THEN hh$=
MID$(a$(cd,i),2):PRINT
16090 IF LEFT$(a$(cd,i),1)="$" THEN PRIN
T hh$:PRINT "  ";MID$(a$(cd,i),2);
16100 IF a(cd,i)<>0 THEN PRINT TAB(32);U
SING"####.##";a(cd,i)
16110 PRINT:PRINT "Commands available:":
PRINT
16120 PRINT "1) Next Item"
16130 PRINT "2) Change Amount"
16140 PRINT "3) Return to Menu"
16150 PRINT "4) Delete Item"
16160 PRINT:PRINT "Which do you require?
"
16170 q$="":WHILE q$=""
16180 q$=INKEY$
16190 WEND
16200 IF q$="4" THEN GOSUB 17000:RETURN
16210 IF q$="3" THEN RETURN
16220 WHILE q$="2" AND LEFT$(a$(cd,i),1)
<>"*"
16230 PRINT:INPUT "Amount to be added:",
q
16240 PRINT:INPUT "Is that correct (y/n)
";r$
16250 IF LOWER$(r$)="y" THEN a(cd,i)=a(c
d,i)+q:q$=""
16260 WEND
16270 WEND
16280 NEXT i
16290 RETURN
```

Comentário

Linhas 16070-16090: se o item chamado a partir do ficheiro for um item singular, então este será impresso — embora despojado do carácter indicador que se encontra junto ao início do nome. Se o item for um cabeçalho principal, não só será

impresso, como o respectivo nome será armazenado em HH$, para que possa ser impresso acima de qualquer dos seus subtítulos seguintes.

Linhas 16220-16260: para além do apagamento de itens, podem ser feitas alterações ao valor associado a um título. Isto consegue-se pela inserção de um número positivo ou negativo, através do qual o valor de um item pode ser alterado — não um valor absoluto que o item deve assumir. A vantagem disto é que a maioria das alterações resultarão na necessidade de adicionar quantias a itens existentes, já que despesas ou receitas suplementares são feitas sob itens anteriormente existentes. Assim, se for necessário gastar 10 000 escudos extra em reparações ao automóvel, para as quais existe já um título, tudo o que é preciso fazer é passar o ficheiro em revista até esse título e introduzir «10 000».

Ensaio

Execute o programa e chame os dados que armazenou previamente em fita. Chame agora a opção 2 a partir do *menu* e verifique se pode passar em revista os três itens, regressando finalmente ao *menu*. Volte a chamar a opção 2, experimentando, desta vez, adicionar ou subtrair dos dois totais anteriormente introduzidos. A repetição da revista aos itens deverá revelar que você conseguiu alterar com sucesso os respectivos valores.

Módulo 5.2.9: Apagamento de itens

Uma característica final a ser acrescentada, com relação aos itens existentes, é o apagamento. No caso do «Contabilista», o módulo de apagamento é mais complexo do que anteriores exemplos do mesmo tipo. A razão para isto está na existência dos grupos formados em torno de cabeçalhos principais. Enquanto não se registam dificuldades relacionadas com o apagamento de um item singular ou de um subtítulo, que acontece quando um cabeçalho principal é apagado? A resposta, obviamente, é que o cabeçalho principal deve, não só ser retirado, como sê-lo juntamente com os subtítulos que lhe estão associados — de outro modo, a contabilidade tornar-se-ia emperrada por subtítulos ligados de um cabeçalho principal, tornando incompreensíveis as contas.

Módulo 5.2.9: linhas 17000-17140

```
17000 REM*******************************
17010 REM Deletions
17020 REM*******************************
17030 pl=i:gr=1
17040 d=0:WHILE LEFT$(a$(cd,pl),1)="*" A
ND d=0:d=1
17050 WHILE LEFT$(a$(cd,pl+gr),1)="$"
17060 gr=gr+1
17070 WEND
```

```
17080 WEND
17090 FOR k=p1 TO c(cd)-gr-1
17100 a(cd,k)=a(cd,k+gr)
17110 a$(cd,k)=a$(cd,k+gr)
17120 NEXT k
17130 c(cd)=c(cd)-gr
17140 RETURN
```

Comentário

Linha 17030: a posição em que o apagamento deve ter lugar é enviada do módulo anterior, sob a forma da variável I. Esta é transferida para PL, por razões que se prendem com o módulo corrente. A variável GR (GRupo) regista a quantidade de itens que devem ser apagados. É regulada, inicialmente, para 1 e apenas será aumentada se o item a ser apagado for um cabeçalho principal, com subtítulos adjuntos.

Linhas 17040-17080: estes dois ciclos incorporados apenas serão activados se o item especificado para apagamento for um cabeçalho principal. O ciclo interior analisa as entradas seguintes, contando quantos daqueles itens a seguir são precedidos por «$», indicando que se trata de subitens do cabeçalho principal. O resultado da contagem é guardado em GR.

Linhas 17090-17120: trata-se de um ciclo típico, destinado a eliminar um quadro e a apagar um item. A diferença, aqui, está em que, em vez de o item X ser copiado para o espaço X-1 e, portanto, cada elemento ser copiado um espaço para baixo, os itens são transferidos GR posições, apagando assim GR itens, ou o número de itens no grupo baseado em torno de um item principal.

Ensaio

Seguindo o procedimento de ensaio usado para o módulo anterior, o utilizador deverá não apenas ser capaz de passar os itens em revista e alterá-los como também apagá-los. Se apagar o item etiquetado MAIN TEST, deverá verificar que SUB-TEST desaparece com ele.

Módulo 5.2.10: Visualização da contabilidade

Depois de todos os preparativos, este é o módulo que dá sentido ao conjunto, através da visualização da contabilidade na sua forma final. Tal como o módulo equivalente no «Banqueiro», este também parece complexo, mas, uma vez obtida a visualização, depressa perceberá por que razão tudo está disposto como está.

Módulo 5.2.10: linhas 18000-18310

```
18000 REM*******************************
18010 REM Print Accounts
18020 REM*******************************
18030 tt=0:ss=0
18040 CLS:PRINT "Print Accounts:":PRINT
18050 PRINT:INPUT "Send accounts to prin
ter (y/n)";p$:ch=0:IF LOWER$(p$)="y" THE
N ch=8
18060 CLS
18070 PRINT #ch,TAB(20-LEN(cd$)/2);cd$:P
RINT
18080 FOR i=0 TO c(cd)-1
18090 tt=tt+a(cd,i)
18100 IF LEFT$(a$(cd,i),1)="*" THEN PRIN
T #ch
18110 IF LEFT$(a$(cd,i),1)="$" THEN PRIN
T #ch,"  ";
18120 PRINT #ch,MID$(a$(cd,i),2);
18130 d=0:WHILE LEFT$(a$(cd,i),1)<>"*" A
ND d=0:d=1
18140 PRINT #ch,TAB(18);
18150 IF LEFT$(a$(cd,i),1)="%" THEN PRIN
T #ch,TAB(29);
18160 PRINT #ch,USING "####.##";a(cd,i);
18170 IF LEFT$(a$(cd,i),1)="$" THEN ss=s
s+a(cd,i)
18180 WEND
18190 PRINT #ch
18200 WHILE ss<>0 AND LEFT$(a$(cd,i+1),1
)<>"$"
18210 PRINT #ch,TAB(18);"--------"
18220 PRINT #ch,TAB(29);USING "####.##";
ss
18230 ss=0
18240 WEND
18250 NEXT i
18260 PRINT #ch,TAB(29);"--------"
18270 PRINT #CH,"TOTAL:";TAB(29);USING "
####.##";tt
18280 IF ch=8 THEN RETURN
18290 PRINT:PRINT "Press any key to cont
inue"
```

```
18300 WHILE INKEY$="":WEND
18310 RETURN
```

Comentário

Linha 18090: a variável TT será utilizada para armazenar o total corrente das contas, à medida que são impressas.

Linhas 18100-18120: estas linhas imprimem o nome do item. Para um item principal, é impressa primeiro uma linha em branco, para o separar do que está para trás, enquanto, para um subtítulo, os dois espaços englobam o nome do item.

Linhas 18130-18180: estas linhas tratam da impressão das quantias para itens singulares e subtítulos. Os subtítulos serão impressos na 18.ª posição ao longo da linha e os itens singulares na 29.ª posição. Se o item sob tratamento for um subtítulo, então o subtotal dos itens do grupo corrente será armazenado temporariamente em SS.

Linhas 18190-18230: este ciclo apenas opera quando um grupo está em processamento, tal como é indicado pelo facto de SS não ser equivalente a 0 e de o item seguinte não fazer parte do grupo — isto é, o grupo está completo. O efeito do ciclo consiste em imprimir o total do grupo na coluna principal de valores, na posição 29 ao longo da linha.

Ensaio

Volte a carregar os dados que previamente armazenou em fita e chame a opção 3 do *menu* principal. Se este ensaio for executado com sucesso, o programa estará pronto para ser utilizado.

NÃO SÃO LIVROS PARA PRINCIPIANTES

PROGRAMAÇÃO AVANÇADA PARA O *AMSTRAD*

DAVID LAWRENCE

Este livro pretende demonstrar como se desenvolvem programas de aplicação sérios para utilização no *Amstrad.* O autor destaca a importância de os programas obedecerem a uma concepção e planeamento cuidados e ilustra os pontos principais com grande profusão de exemplos.



A obra descreve as vantagens da programação modular, seguindo-se capítulos sobre métodos adequados de introdução da informação, manuseamento de cadeias, como evitar erros, armazenamento e recuperação de informação, estruturas de dados, ordenação e busca.

Valiosas sugestões e novas ideias com aplicação nos diversos modelos do *Amstrad.* Um valioso instrumento a ter sempre à mão.

AVENTURAS COM O *ATARI*

TONY BRIDGE

Texto de aventuras; Scott Adams e Infocon; Aventuras de galerias labirínticas; Masmorras e Dragões; Vedetas do *software;* Apresentação do elenco; Escolha de uma aventura.



Esta é a primeira parte do livro, que aborda a fase inicial da aventura — a concepção do jogo, versão somente de texto, no qual o jogador terá de solucionar muitos *puzzles* de forma a encontrar o caminho certo.

A lenda; Criação da sua própria masmorra; Monstro, Monstro!; Ataque e defesa; Figuras divertidas; Vamos dar um passo; O *menu,* por favor. Esta a segunda parte, que apresenta uma secção de gráficos, uma outra de texto da aventura *O Olho do Guerreiro das Estrelas.* Cada linha completamente discutida. Muitas rotinas podem ser usadas nos seus próprios programas.

EUROPA-AMÉRICA...
...a memória no futuro

Este livro é uma recolha de programas consistentes para aplicação. As matérias abordadas pelos programas incluem as finanças domésticas e os impostos, o armazenamento de informações e respectiva utilização, gestão doméstica e quotidiana, gráficos criativos e técnicas de visualização efectiva, música, educação e um conjunto de programas mais pequenos que exploram as possibilidades do *Amstrad* como máquina do tempo.

Todos os programas do livro são claramente explicados e escritos em módulos identificáveis, para que os métodos neles incluídos possam ser copiados para os seus próprios programas.

O *Amstrad* abre novos caminhos por entre os micros pessoais no que se refere a potência e custos, oferecendo perspectivas de novas e entusiásticas áreas de aplicação. Os livros de David Lawrence iniciaram milhares de pessoas na escrita dos seus próprios programas de aplicação. Agora, ele mostra como toda a potência do *Amstrad* pode ser liberta através do potente e novo BASIC.

David Lawrence é o autor de vários livros *best-seller* para computadores, entre os quais *O SPECTRUM FUNCIONAL*. Simon Lane é um conceituado programador, cujo *software* de jogos tem sido vendido em muitos países pelo mundo fora.



Informática 4

12500487 1

12509487 1

O AMSTRAD Funcional

David Lawrence
e Simon Lane

Informática