11

Esta obra destina-se a todos aqueles
que se interessam pelas infinitas possibilidades
do *Spectrum,* mas especialmente aos divulgadores
de vendas e aos homens do *marketing*
que já alguma vez se interrogaram:

— Como poderá o meu computador pessoal evitar
que eu gaste uma fortuna em programas?
Poderei eu próprio elaborá-los de modo a fazer
previsões, fichas de clientes, gráficos,
criar uma base de dados e reduzir o volume
da papelada com que me debato?
A resposta do autor deste livro
é afirmativa e categórica.

Peter Jackson é um homem de negócios
de invejável experiência
que desempenhou cargos importantes
de direcção e gestão de empresas.
Possui actualmente uma firma de *software*
e é leitor convidado
da London Business School.

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

BIBLIOTECA VERBO DE INFORMÁTICA

O SPECTRUM NA EMPRESA P. JACKSON

O SPECTRUM NA EMPRESA

PETER JACKSON

VERBO

BIBLIOTECA DE INFORMÁTICA

O Spectrum na empresa

PETER JACKSON e PETER GOODE

O Spectrum na empresa

Verbo

ÍNDICE

	Introdução	7
1	**COMPUTADORES ... Amigos ou inimigos?**	9
2	**Programação em BASIC**	17
3	**PRINCÍPIOS de programação**	33
4	**AJUSTADOR ... Ajustamento de tendências de vendas**	58
5	**GRÁFICOS ... Traçado de gráficos e mapas**	73
6	**PREVISOR ... Previsão de vendas**	91
7	**CONTACTOS ... Registo de clientes**	107
8	**QUEM vende o quê?**	120
9	**TENDÊNCIA DE VENDAS, programa para o director de vendas**	161

Título do original inglês:
Business programming on your Spectrum

Tradução de
Raul Monteiro de Sousa Machado

N.º Ed. 1719
Composto por Fotocompográfica, Ld.ª
Impresso por Empresa Litográfica do Sul
em Outubro de 1986
Depósito Legal n.º 12 802/86

Introdução

Este livro destina-se aos interessados na aplicação de computadores pessoais a verdadeiros problemas de negócios. Foca essencialmente aspectos de direcção de vendas e de *marketing*, sobre os quais, apesar da relativa falta de literatura de microcomputadores, existe enorme campo para a procura de melhores soluções para problemas bem conhecidos. Como, em qualquer ramo da actividade comercial, se verifica sempre uma componente de vendas, este livro reveste-se ainda de interesse para todos os gerentes ou directores de departamento, bem como para os que, de alguma forma, trabalham nos ramos de vendas e de *marketing*.

Os primeiros capítulos, que compreendem uma introdução à programação em BASIC e se debruçam sobre alguns dos princípios de programação, conduzem o leitor à parte principal da obra. Esta última está estruturada em cinco capítulos, cada um dos quais apresenta um programa relacionado com um aspecto específico da direcção de vendas ou de *marketing*. Todos estes capítulos seguem uma organização comum, isto é, à introdução do problema a analisar seguem-se a descrição genérica do programa, a explicação pormenorizada e as referências à listagem completa que encerra o capítulo.

A ordenação dos diversos capítulos está feita de modo a que, sempre que isso seja lógico, se incorporem facilmente nos programas mais longos algumas secções de programas anteriores. Esta estruturação evita duplicações desnecessárias de texto, mas obriga a que o livro tenha de ser lido pela ordem natural dos vários capítulos, pelo menos na primeira abordagem da matéria.

Os programas estão escritos no BASIC da Sinclair, para correrem no microcomputador *Spectrum*. Os dois últimos são bases de dados e requerem a utilização de uma *microdrive*.

O último capítulo descreve em termos genéricos um bloco de gestão integrada, em que cada subsecção foi já objecto de estudo em capítulos anteriores. As explicações focam o modo de interligação dessas subsecções, de forma a que o programa constitua um todo coerente. O leitor é então convidado a prosseguir para lá do fim do livro.

Para se obter um rendimento máximo das listagens aqui apresentadas, recomendamos o uso de uma *microdrive ZX* nos programas «Contactos» e «Quem». O primeiro destes pode correr no *Spectrum* de 48 K sem qualquer periférico, mas neste caso o volume de informação a armazenar ficará bastante limitado.

Capítulo 1

Computadores Amigos ou inimigos?

Qual a nossa atitude perante os computadores? Os técnicos de vendas terão quase de certeza opiniões discordantes. O principal contacto de um vendedor com um computador será qualquer coisa do género «Vendas efectivas *versus* objectivo. Zona 12. Representante João Sabão», que lhe colocam à frente na reunião mensal para análise do cumprimento das tarefas. Este encontro verifica-se entre o vendedor, o director de vendas e uma pilha de cinco centímetros de altura de papel às riscas verdes e brancas. O director dá voltas e reviravoltas ao monte de papel, tentando descobrir onde começa o texto, incapaz de se desenvencilhar das páginas, que normalmente estão todas ligadas e em que metade está de pernas para o ar. Naturalmente, começará por dizer «O computador afirma que você está abaixo do objectivo em...».

Pode suceder que a fábrica seja gerida por um computador. Nesse caso, o cliente que o vendedor conhece por Companhia de Cestos Palhinha será listada como COMCESPAL, e terá um número parecido com E67043263. Se pretender descobrir o que se passa com a última encomenda do cliente, ninguém lhe explicará coisíssima nenhuma a não ser que o vendedor se lembre do número de computador dessa firma. Ultrapassada esta dificuldade, o pessoal da informática vai à procura dos dados noutra enorme pilha de papel (ou, se a empresa estiver tecnologicamente avançada, num monitor vídeo). Radiantes, informá-lo-ão de que a encomenda será despachada no fim da semana 621. Acorre imediatamente à memória do vendedor que, uma vez, já prestou informação semelhante a um cliente, tendo recebido a seca resposta de que, normalmente, o ano só tem 52 semanas. Ficou assim a saber que 621 significa, na realidade, 21 de Junho, pois é assim que os Americanos escrevem as datas!

E ao vendedor já apareceram, de certeza, listas de cessação de fornecimentos e lembra-se como se divertiu quando teve de informar um cliente de que a linha de abastecimento fora cancelada por falta de pagamento dos artigos enviados nos últimos três meses. «Tenho a certeza de que o departamento de contabilidade não cometeu qualquer erro... Mas, para seu descanso, vou verificar de novo... Talvez se trate de um erro do computador...»

Pois é, a maior parte dos vendedores ainda não confia totalmente nos computadores...

Temos de considerar, no entanto, os aspectos positivos. Repare-se, por exemplo, que dos gabinetes de desenho desapareceu a maior parte dos estiradores e prateleiras. Os desenhadores sentam-se agora em frente de terminais de vídeo de computador, e os traçados e esquemas saem de impressoras automáticas. Mais abaixo, nas oficinas, os operários introduzem *cassettes* nas máquinas-ferramentas, que trabalham sozinhas e na perfeição. Os contabilistas já não contabilizam nada, pois é o computador que trabalha. Será que uma parte de nós mesmos ainda tem a estranha sensação de que está a ser rapidamente ultrapassado, só porque ainda escrevemos à mão o nome e a morada dos clientes naquele velho fichciro que se parece com uma caixa de sapatos?

Se se tratar de um director de vendas, o problema ainda é mais sério, porque são os colegas quem se encarrega de todos esses novos equipamentos. O director financeiro tem números sobre tudo, e com certeza correctos, porque saíram do computador. O director de produção também dispõe do seu próprio computador, que não está em total concordância com o do director financeiro quando se trata de orçamentos, mas que está 100% correcto no que respeita a quantidades físicas, porque estas são indicadas pelo programa de controle de produção. Até o próprio director de pessoal tem uma coisa a que chama *package*, que lhe proporciona informações sobre a produtividade da firma e sobre um misterioso assunto chamado «valor acrescentado».

Na reunião mensal da direcção, o tópico, em geral, versa o porquê de os resultados mensais não coincidirem com as previsões (feitas por toda a equipa de *management*, salienta o administrador). A surpresa das surpresas surge quando se observa que a previsão aponta para uma diminuição das encomendas, e quando se verifica que a produção não consegue acompanhar a necessidade de redução das vendas. Os preços têm tendência a baixar, e o director financeiro está convencido de que esse facto tem a ver com o errado estabelecimento dos descontos em grandes encomendas.

O director de vendas replica com factos, como o de não ter ainda visto os relatórios de vendas do último mês, com a habitual quebra de encomendas em Agosto, e que é pena que o orçamento não tome em consideração estas realidades. Pergunta o que se passa com a satisfação das encomendas em atraso, e refere o facto de ter de ouvir as queixas de alguns clientes cujas encomendas ainda não estão satisfeitas. Mesmo assim, os seus veementes «por exemplo» e «lembro que» não conseguem sobrepor-se às montanhas de papel cheias de dados sólidos, à disposição dos dedos cheios de razão dos colegas. Mais uma vez, a reunião é dada por encerrada com mais um claro sinal do administrador de que ninguém está satisfeito com o departamento de vendas.

VENDEDORES E COMPUTADORES

Há um certo número de dados que poderá explicar o facto de a actividade das vendas ter sido ultrapassada pelo processamento automático de dados (ADP, de *automatic data processing*).

— Até há muito pouco tempo, os computadores eram grandes e caros, e só cabiam em grandes escritórios. Para se servir deles, o vendedor tinha de passar o dia num gabinete, e na realidade os bons vendedores são os que trabalham sempre longe das instalações da firma.

— A maior parte dos dados úteis aos vendedores são fundamentalmente diferentes dos necessários aos contabilistas, que

tradicionalmente constituem a «casta» que tende a dominar o processamento de dados. As vendas estão quase sempre relacionadas com tendências. Os vendedores vêem os produtos mais como mercados a servir do que como unidades que têm forçosamente de passar por todo um processo de produção. Procuram, sobretudo, descobrir padrões de compra e, por exemplo, tentam separar em categorias os clientes que compram todos os meses dos que só compram uma vez por ano. O pessoal das vendas necessita de dados que vêm de há muito tempo atrás, de modo a poderem descobrir indícios de que a procura do mercado está em alteração, de que existem novos produtos na concorrência e de que os clientes estão a deixar de comprar os artigos já muito vistos. Os contabilistas, pelo contrário, normalmente só se interessam pelos dados do último mês. Podem servir-se de números dos meses transactos, ou mesmo de dados acumulados dos últimos anos, mas só para fins de comparação, e, desde que um dado mês esteja terminado, os números respectivos passam a figurar nos arquivos históricos da empresa, pelo menos até ao fim do ano económico.

— Os computadores são geralmente instalados para reduzir os custos, especialmente os de trabalho de secretaria. Se, adicionalmente, vêm a obter-se melhores informações, esta realidade é bem-vinda, se bem que não constitua razão para justificar o investimento em material informático. Além do departamento de registo de encomendas, um escritório de vendas não constitui campo ideal para a redução de custos através da informatização.

— O pessoal das vendas não está normalmente muito relacionado com os números. Durante a sua carreira, não tem tantas oportunidades de seguir cursos de especialização em informática como os colegas de outras especialidades, nem de se servir de dados processados por computador.

Tudo isto está, porém, a mudar. Como toda a gente sabe, os computadores são cada vez mais baratos e, facto importante para os vendedores, há um número crescente de modelos completamente portáteis. As direcções das empresas estão mais preparadas psicologicamente para comprar computadores com o fim de aumentar a produtividade do que para reduzir os custos, e a ideia de dotar as pessoas-chaves com computadores próprios cada vez é mais vulgar. Com certeza que se verificam problemas na área do *software* (programação), mas estes nunca são insuperáveis, e não há dúvida de que hoje em dia se podem escrever programas para todos os fins que se queiram.

Mesmo assim, tem de se considerar a grande família dos vendedores ligeiramente atrasada no campo da informática. A razão principal reside na chamada «ignorância da matéria» do vendedor normal. Os engenheiros podem dispor dos seus computadores pessoais porque parecem ser o género de pessoas especialmente vocacionadas para o processamento de dados; no entanto, e se encararmos cruamente a realidade, pouquíssimos engenheiros especializados são capazes de se servir de um computador, a não ser que lhes dêem as suas próprias máquinas. O mais importante é que todos eles sabem o que fazer quando se sentam em frente de um computador, enquanto os vendedores nunca tiveram essa oportunidade.

AMADORES DE INFORMÁTICA CONTRA PROFISSIONAIS

Os computadores ganharam, desde a sua invenção, uma aura de mistério, e permaneceram acessíveis somente a alguns, poucos, privilegiados. Ninguém fora do círculo restrito dos peritos pôde dizer de sua lavra sobre assuntos informáticos, sem que tivesse de vencer impenetrável barreira de calão incompreensível. Mesmo os mais temíveis executivos, sempre prontos a cair em cima dos subordinados, procuram afastar-se quando encontram um grupo deles a discutir «ADP». Mesmo que suceda — e é mais frequente do que se pensa — que um novo sistema computorizado demore o dobro a registar uma encomenda do que o antigo processo manual, tal facto é imediatamente aceite como o preço do progresso, e como justificação para um novo investimento. Pode suspeitar-se de incompetência, mas quem estará à

altura de o provar? E porquê? A explicação é óbvia. Ao contrário de quase todas as actividades comerciais, a maior parte das pessoas não faz a mínima ideia de quais as verdadeiras possibilidades de um computador. De resto, quase ninguém conhece a fundo as técnicas de venda, o que não constitui factor impeditivo de generalizada emissão de opiniões altamente abalizadas sobre a eficiência das realidades do mundo dos negócios.

Talvez o benefício mais valioso do emprego cada vez mais generalizado de computadores pessoais resida no facto de até a menos abalizada das pessoas começar agora a tecer considerações sobre o trabalho dos profissionais da informática. A revolução já começou: todos ouvimos já comparações sobre determinada apresentação em *écran* feita por um profissional, confrontada com os vulgares jogos de computador... Começamos a interrogar-nos sobre o motivo por que o nome de cada cliente tem de ser comprimido em doze caracteres, ou por que razão os títulos das colunas não poderão ser colocados no topo de cada página, e por aí adiante...

Quanto maior a familiaridade, maior a propensão dos directores e gerentes para se dedicarem à análise de sistemas — isto é, quais e como serão os dados a obter de determinado programa, antes da respectiva concepção. A abordagem tradicional deste problema é verdadeiramente democrática: o perito em computadores realiza entrevistas com os futuros operadores, mas, como nenhum destes faz ideia das possibilidades que serão colocadas ao seu dispor, as respostas dadas não têm o mínimo sentido. Para o perito, a grande vantagem desta iniciativa reside no facto de que, à partida, já sabe que se entrevistar vinte pessoas encontrará vinte pontos de vista diferentes. Deste modo, pode em consciência instalar aquilo que ele próprio julga ser mais adequado para toda a gente. Além disto, se o sistema instalado falhar, pode rever as suas notas e descobrir facilmente que a causa principal das dificuldades foi a necessidade de satisfazer um pedido obscuro qualquer. Ninguém em consciência é capaz de iniciar uma actividade desta forma amadora, a não ser o pessoal dos computadores, e qualquer gerente que se preze terá de refrear o seu entusiasmo antes que o negócio vá por água abaixo.

O VENDEDOR E OS COMPUTADORES PESSOAIS

Para além dos possíveis benefícios obtidos do emprego de computadores, e das possibilidades de influenciar aqueles que com eles trabalham, os que se dedicam ao ramo de vendas estão particularmente bem colocados para se servirem do que se costuma designar por «computação pessoal». A distinção entre computação «pessoal» e «processamento de dados» nada tem a ver com as máquinas em si — refere-se, sim, ao que se faz com o computador. Qualquer pequena empresa que se sirva de um microcomputador para calcular o pagamento de salários está a realizar «processamento de dados». A computação «pessoal» refere-se mais ao emprego de computadores para o melhoramento do trabalho de cada um, especialmente se as tarefas implicam tomada de decisões e avaliação de situações.

Os vendedores têm largamente facilitado o seu trabalho se souberem usar acertadamente os dados disponíveis. Qualquer curso de vendas dá o devido realce ao conhecimento do mercado, às características dos possíveis clientes, à competição a enfrentar, etc. Do mesmo modo, os objectivos de uma acertada gestão de vendas residem no estabelecimento de um criterioso programa de visitas, na obtenção de um equilíbrio óptimo entre cobertura de mercado e custos, na análise de padrões de encomendas, na detecção de eventuais alterações do mercado, etc.

Existem muitos sistemas manuais de registo de dados concebidos para auxiliar o vendedor no seu trabalho do dia-a-dia, mas todos eles são repetitivos e verdadeiros consumidores de tempo. Por outro lado, este tipo de actividade é precisamente a que se presta a ser feita por qualquer pequeno computador.

Com o recente aparecimento de verdadeiros minicomputadores portáteis, o vendedor ficou armado com o equipamento necessário à determinação de soluções para complexos pro-

blemas típicos do mercado, e fá-lo em pleno escritório do cliente, à frente deste e com respostas imediatas. Como é óbvio, ainda se verificam lacunas, mas os benefícios obtidos compensam em larga escala os poucos problemas que surjam.

Num nível mais social, é hoje possível escrever qualquer carta em casa com qualidade semelhante (ou igual) à obtida num escritório, através de um microcomputador que trate os conhecidos «processadores de palavra». Podem igualmente elaborar-se relatórios com um «toque» profissional, que, se forem gravados em *floppy discs*, serão enviados pelo correio para a sede da empresa, para serem analisados depois de impressos. Para os mais arrojados, existem dispositivos capazes de enviar os dados através do telefone, por ligação a um número no escritório, atendido directamente pelo computador principal.

Capítulo 2

Programação em BASIC

INTRODUÇÃO

Antes de começar a programar, é fundamental a familiarização com dois grupos diferentes de instruções:

— a linguagem de programação

e

— o sistema operativo

As linguagens de programação (entre as quais se conta o BASIC) são sistemas de instruções que permitem ao computador executar tarefas complexas. O programa é o elo de ligação entre as possibilidades relativamente limitadas do computador e os complicados assuntos do mundo real. Na verdade, é a capacidade de ser programado que distingue o computador do seu parente próximo, a calculadora electrónica.

O segundo grupo de instruções, o sistema operativo, tem uma finalidade completamente distinta: controla as ligações entre o computador e os periféricos que lhe sejam ligados. Por outras palavras, está relacionado com o arranque e a paragem da máquina, o carregamento e a gravação de dados, o encaminhamento da saída para uma impressora ou para um monitor, etc. Tem de haver uma ligação entre o sistema operativo e a linguagem de programação, de forma a que as instruções do primeiro sejam dadas directamente pelo programa, mas para além deste elo os dois sistemas são muito diferentes entre si. Normalmente, o sistema operativo é concebido para uma máquina específica, enquanto a linguagem de programação é essencialmente a mesma, sem ter em conta o computador que a executará.

Nos primeiros contactos com o computador, pode muito bem suceder que tenhamos mais dificuldades com o sistema opera-

tivo do que com a linguagem de programação. Em parte porque a linguagem já existe há muito mais tempo que o sistema operativo, tendo sido objecto de sucessivas alterações que a levaram à quase perfeição, enquanto o sistema operativo é quase sempre de concepção recente. Por outro lado, e talvez a razão principal seja esta, o sistema operativo lida com as conexões puramente electrónicas do computador e o mundo híbrido electromecânico dos teclados, impressoras, gravadores, sistemas de discos, monitores, e por aí adiante. No que se refere aos computadores pessoais, os problemas agravam-se, porque muitas das máquinas periféricas não foram originalmente concebidas para serem ligadas a um computador. A resposta para estas dificuldades técnicas não é fácil, pois não se trata de as resolver através de raciocínios matemáticos ou de puro discernimento lógico. É muito mais fácil aprender a trabalhar com uma nova máquina de lavar roupa, caso em que os problemas a resolver dependem essencialmente de quem a concebeu e de quem escreveu o manual de instruções.

Com as linguagens de programação, o problema é bem diferente. No caso do BASIC *(Beginners All-purpose Symbolic Instructions Code)*, verificamos logo que milhões de pessoas de todo o mundo se servem dele, tanto em pequenos computadores como nas máquinas de grandes dimensões. Esta linguagem está bem apoiada em vasta literatura, e fizeram-se esforços consideráveis para conceber métodos rápidos e efectivos de ensino e treino de BASIC. Para além das instruções que acompanham o computador, encontram-se à venda variadíssimas obras e manuais com descrições minuciosas dos comandos necessários, com exemplos claros da forma de os empregar e referências sobre as consequências do seu emprego.

Este capítulo não pretende ser um novo manual completo de BASIC, mas, sim, realçar algumas partes desta linguagem de programação de relevância nos programas descritos nos capítulos posteriores.

CONCEITOS FUNDAMENTAIS

A comparação entre computadores e calculadoras, atrás mencionada, proporciona-nos adequada base de partida, em terreno bem conhecido de toda a gente. No que se refere à capacidade para executar operações aritméticas, há poucas diferenças entre as duas famílias de máquinas; qualquer delas executa somas, subtracções, multiplicações e divisões, aceita decimais e números negativos, etc. Calculadoras há que possuem memórias que guardam números posteriormente chamados sempre que seja necessário. Os computadores também têm memórias, mas, em lugar de um único número, podem armazenar milhares de valores diferentes. Imagine-se uma calculadora com muitas memórias diferentes — podendo nós guardar um número, armazenar um outro, executar um cálculo, memorizar o resultado noutra memória, voltar a chamar o primeiro valor, fazer novo cálculo... às tantas, como nos lembraríamos ainda dos locais onde guardámos tantos números? A solução não oferece grande dificuldade, desde que se invente um sistema qualquer de organização e controle das diversas memórias e dos respectivos conteúdos. Os computadores possuem tal sistema, as calculadoras não.

É esta grande capacidade de memória dos computadores que permite não só armazenar os valores que se lhe introduziram como guardar uma lista de instruções (o programa), que nada mais faz do que indicar como devem ser processados os dados memorizados. Mais ainda, é esta larga capacidade que possibilita a transferência, nos dois sentidos, de dados e de programas entre o computador e periféricos tais como gravadores, sistemas de discos e impressoras.

Em resumo, é a existência de uma memória de dimensões consideráveis que distingue um computador de uma vulgar calculadora electrónica; tudo o que envolva programação e aplicação de instruções ao sistema operativo nada mais é do que organização e controle do vasto número de memórias disponíveis.

VARIÁVEIS

A memória é composta por milhares de elementos de armazenagem individuais (imaginemo-los como caixas de correio) em que podemos guardar dados. O sistema de armazenagem é controlado dando a cada «caixa» um nome, que a define e que, uma vez dado, não pode ser alterado, a não ser que se desligue o sistema e comece tudo de novo. Para se baptizar e dar o valor a uma variável, servimo-nos da instrução **LET:**

```
LET K=20
```

Esta ordem designa (ou etiqueta) um receptáculo até aí vazio com o nome K, e atribui-lhe o valor 20. Se daqui em diante, ao falar com o computador, fizer referência a K, ele actua indo à procura do receptáculo chamado K.

Se, por exemplo, lhe dissermos:

```
PRINT K
```

Obteremos a resposta:

```
20
```

O computador também está perfeitamente à vontade se se pretender atribuir uma letra à «caixa», ou receptáculo, mas tem de haver um meio de distinguir essa letra colocada num local de outra idêntica que dá o nome a outro ponto da memória. A distinção faz-se por meio de uma convenção que determina que quaisquer letras a memorizar têm de ser escritas entre aspas, tal como: "K". Além disto, o nome da variável tem de ser seguido pelo sinal **$**, de forma a distingui-la das variáveis que possuem valores numéricos. As variáveis literais, normalmente conhecidas pela designação inglesa **STRING**, são definidas da seguinte forma:

```
LET K$="A"
```

Como tal, se introduzirmos a instrução

```
PRINT K$
```


Obteremos a resposta:

```
A
```

O que prova que **K** não é a mesma coisa que **K$**.

ARITMÉTICA E ÁLGEBRA

Tanto os computadores como as calculadoras tratam das matemáticas mais ou menos da mesma forma. Para que uma calculadora multiplique 100×50 e divida o resultado por 25 teremos de introduzir em primeiro lugar o valor 100, seguido do símbolo X, depois 50, depois o símbolo de divisão (/) e finalmente 25, após o que aparecerá o resultado.

Com o computador, todas as instruções são introduzidas antes da efectivação dos cálculos, pois o conjunto das ordens de operação tem de ser colocado antes de se dar execução aos cálculos propriamente ditos e, após o início destes, nada é alterável. Nada pode ficar por dizer, isto é, todas as instruções, da primeira à última, têm de ser metidas em memória; as variáveis têm de ser definidas, os valores iniciais têm de ser estabelecidos, e é forçoso que exista uma lista sequencial com todas as operações a executar. Um programa para os cálculos acima descritos apareceria como:

```
10 LET A=100
20 LET A=A×50 (em que×=multiplicação)
30 LET A=A/25 (em que /=divisão)
40 PRINT A
```

Se tivéssemos inserido uma instrução de impressão *(print)* depois de cada operação, isto é, **PRINT A** como linhas 15 e 25, o computador ter-se-ia comportado exactamente como uma calculadora, apresentando no monitor as respostas intermédias correspondentes a cada uma das operações aritméticas efectuadas. No entanto, uma apresentação deste tipo é redundante em computadores, pois todas as instruções necessárias já foram previamente introduzidas no programa, e uma vez este exe-

cutado nada se ganha em se apresentar as operações intermédias, ou dados parcialmente processados.

Analise-se a linha 20 com atenção. Para começar, não se trata de uma equação convencional. A não é igual a A×50, no sentido a que estamos habituados (se assim fosse, 1 seria igual a 50, o que é errado). A linha 20 é uma instrução que determina LET (deixe que) **o receptáculo A** (que no momento armazena o valor 100) **seja mudado de modo a que passe a guardar um resultado qualquer que represente a multiplicação de A por 50.** Por outras palavras, ela substitui o conteúdo do receptáculo A pelo valor que estiver do lado direito do sinal de igual. Este pormenor é importante, pois significa que não se podem intermutar o lado direito e o esquerdo de um sinal de igual, como é corrente na álgebra convencional. As duas instruções a seguir apresentadas não são a mesma coisa:

```
10 LET A=B
20 LET B=A
```

Na linha 10, o conteúdo de A é substituído por aquilo que for o valor de B. Na linha 20, passa-se o contrário, ou seja, B é substituído por A.

Através das «substituições» de umas variáveis por outras, juntamente com todo o leque de operações algébricas oferecidas pelo BASIC, podemos executar cálculos complexos com toda a facilidade, se soubermos compilar o programa adequado.

Há uma regra muito importante que nunca devemos esquecer: os computadores não conseguem entender-se com ambiguidades. Limitam-se a interpretar estritamente as instruções fornecidas, nunca fugindo às regras estabelecidas pela linguagem que estão a aplicar. É, pois, de importância vital que se torne perfeitamente claro quais as instruções a executar. Um calculador humano pode entender ordens como as duas a seguir indicadas, pois consegue aperceber-se do contexto em que o problema é colocado e escolhe a fórmula correcta:


```
10 LET A=A×B-C
20 LET B=B-C/A
```

Os computadores não conseguem discernir aqui o que se pretende. A sua interpretação da álgebra é regulada pela chamada «regra da precedência», que diz como observar com precisão as instruções recebidas. O programador tem de se certificar de que as instruções que vai escrevendo não fogem a essas regras. Para tal, dispõe de duas opções: 1) aprende as regras de precedência, ou 2) serve-se de parêntesis para tornar perfeitamente claro aquilo que pretende. Esta última solução é a mais fácil — é uma coisa menos a aprender, mesmo que à custa de mais trabalho de dactilografia! A adição de um par de parêntesis às duas expressões acima indicadas libertam-nas de toda e qualquer ambiguidade:

```
10 LET A=(A×B)-C
```

ou

```
20 LET B=B-(C/A)
```

LÓGICA

Além das funções aritméticas normais, os computadores têm a possibilidade de tratar com declarações lógicas, ou «tabelas de verdades». Isto é, avaliam declarações lógicas e dizem se estas são «verdadeiras» ou «falsas». A resposta é dada pelo valor atribuído à expressão; se este for 1, é verdadeira, mas é falsa se o valor for igual a 0. A declaração (D>2) tem o valor 1 se D for igual a 3 ou superior, e o valor 0 se D for igual ou inferior a 2. O símbolo > significa «maior que».

As declarações lógicas podem ser combinadas com a aritmética convencional, como no exemplo que se segue:

```
10 LET P=P×(D>2)
```

Neste caso, P=P×1 se D for maior que 2, e P=P×0 se D for igual ou inferior a 2. Evitaram-se quaisquer ambiguidades colo-

cando parêntesis, o que também é tão importante como no caso das simples instruções algébricas.

ARITMÉTICA DE VARIÁVEIS LITERAIS

As variáveis literais ou de texto *(strings)* podem ser adicionadas mais ou menos como as variáveis numéricas, como se vê no seguinte exemplo:

```
10 LET A$="RAPAZ": LET B$="E RAPARIGA"
20 LET C$=A$+B$
30 PRINT C$
```

O resultado é «RAPAZ E RAPARIGA».

As variáveis literais têm o seu conjunto próprio de instruções, que permite manipulá-las sob formas que não têm equivalente na forma numérica. O pequeno programa que se segue tira as iniciais de um nome (desde que este tenha sempre duas iniciais!):

```
10 LET A$="R.S.MACHADO"
20 LET L=LEN (A$): REM L é o número de caracteres em A$.
30 LET A$=A$ (5 TO L): REM A$ são os caracteres à direita (L-4) de A$. Lembre-se de que os pontos também são caracteres.
40 PRINT A$
```

O resultado é «MACHADO».

Por mais estranho que pareça, uma das aplicações mais úteis da aritmética de variáveis literais é o tratamento de números, como, por exemplo, no caso em que precisamos de transformar um número decimal no seu equivalente inteiro.

```
10 LET A=46.3467
20 LET A$=STR$(A): REM A$ é a variável equivalente a 46.3467.
30 LET A$=A$ (1 TO 2)
40 LET A=VAL (A$)
```

A linha 40 converte A$ na variável numérica A, que tem o valor 46.

GRUPOS DE VARIÁVEIS («ARRAYS»)

Um grupo de variáveis é uma lista sistemática que dá os valores de um número de variáveis pertencentes a uma mesma "família". Por exemplo, as vendas de uma "família" de produtos pode ser colocada num grupo, da seguinte forma:

```
Vendas do produto 1=20
Vendas do produto 2=30
Vendas do produto 3=45
Vendas do produto 4=50
```

Os quatro números que formam a coluna à direita dos sinais de igual são o *array*. Isto é:

```
20
30
45
50
```

Em BASIC, os grupos de variáveis têm nomes, de forma semelhante ao que se faz com as variáveis — este pode-se chamar «grupo A». Um *array* distingue-se das outras variáveis pela colocação entre parêntesis do número de elementos que ele contém, a seguir ao nome. Desta forma, o grupo A escreve-se A(4). Ao número entre parêntesis chama-se «dimensão do grupo», em que cada elemento é definido pelo seu «subscrito» — ou seja, o seu valor em ordem decrescente desde o cimo do grupo. Os elementos do *array* A definem-se como se segue:

```
A(1)=20
```

A(2)=30
A(3)=45
A(4)=50

Os grupos permitem juntar variáveis que irão ser sujeitas às mesmas operações, dentro do mesmo programa. Deste modo, em vez de perdermos precioso tempo a escrever a mesma instrução para o produto 1, depois para o 2, depois para o produto 3 e assim por diante, só temos de dar uma única instrução para todo o grupo, que será encarada de igual modo para todos os seus componentes.

Tal como sucede com as variáveis, os grupos tanto são constituídos por números como por variáveis literais, mas temos de ter presente que não se podem incluir ambos num mesmo grupo. Um *array* de variáveis de texto tem sempre o sinal designativo $ depois do nome, por exemplo **A$(4)**.

MATRIZES

Uma matriz não é mais do que uma série de grupos colocados lado a lado, esquema que se torna muito útil quando queremos processar grupos que em si fazem parte de uma «família» maior. Por exemplo, em vez de um simples grupo de venda de produtos, poderemos ter um conjunto de dados de vendas para vários períodos de tempo.

Vendas dos produtos 1 a 4 em Jan.
Vendas dos produtos 1 a 4 em Fev.
Vendas dos produtos 1 a 4 em Mar.

Temos aqui 12 valores ao todo — 4 produtos multiplicados por 3 meses. Na matriz, esta realidade é representada sob a forma **A(4,3)**, que acaba por mostrar uma «grelha» formada por 4 «filas» de 3 «colunas», cada uma das quais contém um número. Pense-se nisto como se fosse, por exemplo, uma caixa de ovos — quatro filas e três colunas suportando uma dúzia de ovos. As vendas do produto 1 no mês 1 (Jan.) são dadas por **A(1,1)**; as vendas do produto 3 no mês 2 serão indicadas por **A(3,2)**.

Como já deve ter calculado, um grupo de variáveis literais é escrito como **A$(,)**.

CÍRCULOS E DERIVAÇÕES

Como já vimos, as instruções para pôr um computador a executar cálculos aritméticos são muito semelhantes às que usamos na álgebra convencional. A diferença principal reside no facto de que essas instruções são colocadas num conjunto — o programa — antes de se iniciarem os cálculos.

No entanto, este aspecto é um dos vários pormenores da programação informática. A álgebra convencional, quando foi inventada, destinou-se a acompanhar as capacidades do homem, que são bem diferentes das reais capacidades do computador. Como tal, não será surpresa o facto de termos de enfrentar técnicas de programação que não têm equivalente na álgebra a que estamos habituados.

Essas mesmas técnicas destinam-se fundamentalmente à obtenção de melhores resultados do esforço de programação, em vez de simplesmente «automatizarem» a sequência de instruções que seriam usadas por qualquer ser humano.

Se considerarmos que um programa de computador é como um mapa de estradas no qual podemos planear uma viagem à volta do país, poderemos afirmar que a rota traçada terá de ser aquela que permita ver o máximo da paisagem com o mínimo de quilómetros percorridos. Um computador, ao contrário, tentaria maximizar o número de vezes que teria de passar pelo mesmo troço de estrada, porque a justificação para tal seria o «custo por quilómetro» (o tempo gasto a efectuar um cálculo), já que é muito pior para o computador o «custo por cada novo conjunto de instruções» (o espaço de memória ocupado por cada novo grupo de instruções), para não mencionarmos o custo das horas de trabalho do programador.

Uma das consequências desta maximização de resultados é o aparecimento dos *loops*, isto é, «cálculos em círculo» — desconhecidos na álgebra convencional mas de importância primor-

dial na programação de computadores. Aparecem em diversas formas, as mais importantes das quais vamos passar a descrever.

CÍRCULOS «FOR/NEXT»

Este tipo instrui o computador para que repita um cálculo um determinado número de vezes. É composto por duas instruções, **FOR**, que se situa no início do círculo, e **NEXT**, colocada no fim. A sua utilidade é exemplificada no simples programa que se segue, destinado a imprimir os números de **0** a 9:

```
10 FOR N=0 TO 9
20 PRINT N
30 NEXT N
40 STOP
```

O círculo inicia-se na linha **10**, que determina que «não sais deste cálculo até que algo suceda que faça com que o valor de N seja maior que 9». A linha **20** imprime o valor de N, que é **0** na primeira volta, e a linha **30** diz que «de cada vez que aqui chegares, somarei 1 ao número representado por N, e mandar-te-ei de volta para o início do círculo, na linha **10**». Este processo vai-se repetindo até que N percorra o trajecto de **0** a 9, em saltos de 1 unidade. Quando N fica igual a **10**, a linha **10** detecta que N já está fora dos limites **0**-9, o que dá por finalizado o círculo, através da declaração **NEXT N**, neste caso localizada na linha **40**, pelo **STOP**. Repare-se em que o valor de N que determina o fim do círculo é **10**, e não 9. Poderá verificar-se este facto introduzindo uma linha 35, **PRINT N**.

Uma das aplicações principais de círculos como este destina-se à manipulação de matrizes e grupos. O exemplo seguinte multiplica cada um dos elementos do grupo A(4) por 2:

```
10 FOR N=1 TO 4
20 LET A(N)=A(N)×2
30 NEXT N
```


Os mesmos princípios aplicam-se às matrizes. O programa abaixo indicado multiplica cada elemento da fila de cima (fila 1) em A(4,3) por 4:

```
10 FOR M=1 TO 3
20 LET A(1,M)=A(1,M)×4
30 NEXT M
```

Para multiplicarmos toda a matriz por 4, teremos:

```
10 FOR N=1 TO 4
20 FOR M=1 TO 3
30 LET A(N,M)=A(N,M)×4
40 NEXT M
50 NEXT M
```

Esta última listagem contém dois círculos «fechados» — o círculo N nas linhas **10** e **50** e o M nas linhas **20** e **40**.

Não se pode permitir que dois círculos se entrecruzem. Quando uma declaração **FOR** coloca uma variável em círculo, a concomitante declaração **NEXT** que se segue tem sempre de se referir a essa mesma variável. O programa seguinte não respeitou esta particularidade, pelo que nunca será cabalmente executado:

```
10 FOR N=10 TO 20
20 FOR J=30 TO 40
30 PRINT A(N,J)
40 NEXT N
50 NEXT J
```

As capacidades dos círculos podem ser ampliadas através da

instrução **STEP**, que permite o estabelecimento de uma contagem sequencial. Por exemplo:

```
FOR N=1 TO 9 STEP 2
```

faz com que N assuma os valores 1, 3, 5, 7 e por fim 9, em saltos de duas unidades de 1 a 9.

```
FOR N=9 TO 1 STEP -2
```

faz com que N desça de 9 a 1 em intervalos de −2, ou seja, 9, 7, 5, 3, 1.

GOTO

A instrução **FOR/NEXT** determina o número de vezes que o programa tem de passar pelo círculo. O comando **GOTO** pode ser usado para estabelecer um *loop* em que a saída (o fim) pode ser controlada por uma declaração de comparação lógica. O que se segue adiciona os espaços em branco necessários para que uma variável tenha sempre um comprimento uniforme de nove caracteres:

```
10 INPUT A$: REM A$ não pode exceder 9 caracteres
20 IF LEN (A$)=9 THEN GOTO 50
30 LET A$=" "+A$
40 GOTO 20
50 PRINT A$
```

O *loop* inicia-se na linha 20, que faz a diferenciação entre as variáveis com menos de nove caracteres e as que já possuem este número. Se por acaso uma pertencer a este último grupo, o programa salta imediatamente para a linha 50, em que se processa a impressão. Se a variável tiver menos de nove caracteres, o computador coloca um espaço em branco à frente dela, e regressa depois à linha 20 para ver se o comprimento já está correcto. Se ainda não for o caso, o processo vai-se repetindo. Mal A$ fique com o tamanho pretendido (nove caracteres),

efectua-se a saída do *loop*, e o A$ acaba por ser normalmente impresso através da linha 50.

GOSUB ..... RETURN

O comando **GOSUB** é muito parecido com o **GOTO**, pois provoca um desvio imediato para uma determinada linha de cálculo. Contudo, **GOSUB** vai além da outra instrução; mal o desvio tenha sido estabelecido e resolvido, o computador regressa ao ponto de partida original, desde que encontre a instrução **RETURN**. Por outras palavras, **GOSUB/RETURN** dá-nos a possibilidade de, por instantes, abandonar o programa principal e executar uma espécie de programa subsidiário (chamado **SUB-ROTINA**), com regresso automático ao programa que se estava a seguir. O exemplo que se segue serve-se de uma sub-rotina para criar variáveis de comprimento uniforme:

```
10 IF LEN (A$) < 9 THEN GOSUB 1000
20 PRINT A$
30 STOP
1000 REM Estabelecer variáveis uniformes
1010 LET A$=" "+A$
1020 IF LEN (A$)=9 THEN GOTO 1040
1030 GOTO 1010
1040 RETURN
```

As linhas 10 e 20 fazem parte do programa principal. A linha 10 estabelece a ligação à linha 1000, desde que A$ tenha menos de nove caracteres de comprimento. De 1000 a 1030, o computador faz com que a variável fique com o tamanho pretendido, e a linha 1040 provoca o retorno para a corrente principal de processamento, que começa na linha 20.

IF ... THEN GOTO

A simples instrução **IF ... THEN** só suporta dois estados, isto é, **IF PRETO E BRANCO GOTO 100** abrange duas hipóteses, preto é branco e preto não é branco. Contudo, se usarmos mais do que uma destas instruções, teremos acesso a escolhas múltiplas. Tal possibilidade fica demonstrada no exemplo seguinte, em que diferentes valores da variável R representam as cores de um semáforo de controle de trânsito. O programa converte esses números nas cores verde, amarelo e vermelho.

```
10 IF R=1 THEN GOTO 50
20 IF R=2 THEN GOTO 60
30 IF R=3 THEN GOTO 70
40 GOTO 80: REM o programa acaba se R não for 1, 2 ou 3
50 PRINT "VERMELHO": GOTO 80
60 PRINT "AMARELO": GOTO 80
70 PRINT "VERDE"
80 STOP
```

Imaginemos que R é a resposta de um qualquer sistema automático de controle, que assume os valores de 1, 2 ou 3. As linhas 10 a 30 dizem que, conforme o valor de R, o programa terá de se desviar para a correspondente linha secundária. R=1 faz com que a execução passe para a linha 50; R=2 passa o programa para a linha 60, e R=3 passa-o para a linha 70. Em cada uma destas, o programa imprime a cor apropriada. Se R não corresponder a nenhum dos números especificados, qualquer coisa está errada, de modo que a linha 40 passa a execução para a linha 80, **STOP**. As instruções **IF ... THEN GOSUB** podem ser usadas em lugar de **IF ... THEN GOTO**. Tenhamos presente, porém, que o programa executará um **RETURN** para a instrução seguinte depois de qualquer **GOSUB**, o que poderá vir a interferir com as escolhas que apresentaremos no capítulo seguinte.

Capítulo 3

Princípios de programação

INTRODUÇÃO

No seu nível mais elevado, a programação de computadores é uma forma de arte, uma elegante mistura de matemática, lógica e economia. Num bom programa, cada parte é um elo perfeito de uma cadeia de lógica, que não vale a pena melhorar com mais passos e em que o simples retirar de uma instrução fará ruir toda a estrutura. Em programação, a meta principal consiste no uso do mínimo possível de passos (declarações), desde que com eles se atinjam os fins em vista, evitando toda e qualquer falha lógica (o vulgar «gato») que prejudique o todo coerente.

Felizmente, tal grau de perfeição não é necessário se se quizer escrever simplesmente um programa que funcione. Nos programas escritos sem aquelas preocupações, o único preço a pagar pela falta de elegância é uma ínfima diminuição da velocidade de execução e um exagerado uso da capacidade de memória do computador, que se não for excedida não causará problemas de importância relevante.

No entanto, abaixo de certo nível, todo o trabalho de programação torna-se numa tarefa irritante, frustrante e positivamente não compensadora. É claro que não existe um padrão de comparação que possamos seguir, mas, em muitas áreas, a aplicação de certa disciplina e de planeamento prévio facilita enormemente toda e qualquer actividade de programação. Focamos neste capítulo os pontos mais importantes do que aqui ficou dito.

DEFINIÇÃO DE VARIÁVEIS

É quase impossível escrever um programa se não nos certificarmos da definição de cada variável. Imagine-se que estamos perante uma listagem que começa com a variável Z, destinada

a representar o número de dias do mês; mais adiante, Z passa a conter as vendas diárias; e no fim, Z passa a ser um contador dentro de um *loop*. Tal sucessão pode muito bem acontecer se se estiver a escrever um programa comprido, em que as horas de trabalho acabam por ocupar vários serões. Para evitar este problema, o melhor é referenciar cada definição de uma variável através das declarações «REM», logo no início do programa. Outra possível solução consiste em definir as variáveis à medida que vão aparecendo no programa. Por muito que esta técnica afecte o nosso estilo, não devemos fugir muito dela, especialmente se temos de distinguir entre variáveis destinadas a servirem de contadores num círculo e todas as outras (por exemplo, poderemos servir-nos de uma letra para as primeiras e de duas letras para as verdadeiras variáveis).

BANDEIRAS

As bandeiras são variáveis que desempenham o papel de indicadores de determinados estados específicos. Suponhamos ser necessário um programa que calcule preços em várias moedas; a lógica do programa é a mesma, independentemente do câmbio em si, mas em determinadas partes temos de aplicar constantes diferentes, que dependem da moeda que no momento está a ser processada.

Torna-se prático estabelecer no início do programa uma bandeira que indique a moeda que está a ser utilizada, por exemplo a variável FL estabelecida em 1 para escudos, em 2 para marcos, e assim por diante. Quando o programa atinge o ponto em que se verifica tratamento diferente para cada moeda, a bandeira serve para identificar quais os valores a considerar. Por exemplo, teremos **Se FL=1 então Preço=X; Se FL=2 então Preço=1.2×X.**

Em qualquer programa, as bandeiras têm de ser rigidamente controladas, e é muito importante reestabelecê-las em **0** mal acabem de executar as funções para que foram criadas. Caso contrário, e em particular se o programa se serve de numerosos círculos e sub-rotinas, é bem possível que uma bandeira redundante acabe por interferir com a lógica pretendida.

Tal como com os contadores, é boa ideia atribuir nomes claros às bandeiras — a prática corrente consiste em baptizá-las com duas letras, em que a primeira é sempre F (de *flag*), ou seja, FA, FB... FL, etc.

SUB-ROTINAS

A linguagem BASIC não se preocupa muito com sub-rotinas, limitando-se a defini-las por um número de linha, ao contrário do que sucede em outras linguagens, que lhes atribuem nomes. Por outro lado, as sub-rotinas podem ser inseridas em qualquer parte do programa. Como consequência, podem muito bem ser relegadas para um estatuto de «impecilho», que por qualquer razão pouco clara não faz parte do fluxo principal, o que não é propriamente o que se pretende de uma verdadeira sub-rotina. Estas têm de ser blocos autónomos de lógica perfeita, retirados do programa principal pela simples razão de que serão utilizados mais do que uma vez. Para reforçar este princípio, é aconselhável iniciar cada sub-rotina com uma declaração **REM**, que diga claramente o que ela irá fazer.

De forma geral, é prática errada sair de uma sub-rotina a meio caminho desta (isto é, antes de se chegar à instrução **RETURN**), porque *a*) torna-se difícil seguir a lógica estabelecida e *b*) o computador pode ficar confuso por não saber dessa saída extemporânea da sub-rotina, o que só vem a descobrir quando chega à declaração **RETURN**.

NUMERAÇÃO DE LINHAS

Torna-se muito mais fácil seguir a listagem de um programa se o sistema de numeração de linhas for tal que permita diferenciar com clareza as várias partes constituintes, pela simples mudança dos números. Por exemplo, poderemos usar as linhas **10** a **100** para a definição das variáveis, as linhas **500** a **1000** para o estabelecimento das variáveis literais, as linhas **2000** em diante para o programa principal, e iniciarmos as sub-rotinas nas linhas **6000**, **7000**, etc. Qualquer pessoa pode comprar (ou

escrever) um programa que execute a renumeração automática das linhas, de modo que pode poupar tempo começando a escrever a listagem e deixar que o utilitário se encarregue de ir colocando a numeração adequada.

Um pequeno pormenor, fonte de grandes complicações caso seja ignorado, é que o programa principal deve sempre acabar por uma instrução **STOP.** Se assim não for, o programa ultrapassará quase de certeza o fim e iniciará as sub-rotinas, com os resultados mais que inesperados e por vezes desagradáveis.

TRATAMENTO DE VARIÁVEIS LITERAIS

À excepção dos programas simples e curtos, não é boa ideia misturar longas declarações de variáveis literais *(strings)* com as instruções do programa, pois torna-o difícil de seguir e pode provocar lacunas se se pretender corrigir partes da listagem.

Este tipo de problemas pode ser ultrapassado pela utilização de variáveis literais no próprio programa, definindo-as separadamente quer no início quer à medida que forem surgindo. Como benefício adicional desta técnica, o programador pode ir compilando um certo número de frases capazes de virem a ser utilizadas em mais do que uma ocasião.

TRATAMENTO DE NÚMEROS

Normalmente, é necessário formatar números tanto nos estágios de entrada como de saída de um programa. Operações como esta podem revelar-se incómodas, se forem realizadas com variáveis numéricas, caso em que teremos de considerar uma mudança para as variáveis literais. A aritmética aplicada a estas últimas proporciona-nos um conjunto diferente de instruções, que muitas das vezes justificam o esforço implícito à conversão de números em variáveis literais. Por exemplo, quando os dados resultantes de um cálculo complexo têm de ser apresentados sob a forma de tabela, o comprimento dos algarismos tem normalmente de ser restringido a determinado máximo de caracteres. O comprimento total pode ser limitado retirando os primeiros caracteres **L** através de uma declaração **A$ (1 TO L).**

Outra necessidade muito comum é a justificação à direita de uma coluna de números. Para se conseguir esta apresentação, é preciso em primeiro lugar converter esses números em variáveis literais, e depois determinar qual delas é a mais comprida **(L=LEN (A$))**. Para os restantes números, atribui-se-lhes o comprimento da maior das variáveis, colocando espaços em branco à sua frente: **A$=" "+A$.**

TRATAMENTO DE MATRIZES E GRUPOS

As operações com matrizes e grupos *(arrays)*, de maneira geral, tratam todos os elementos da mesma forma, e são executadas através da utilização de círculos. O exemplo que se segue calcula as alterações percentuais de vendas entre dois períodos.

Dois conjuntos de dados de vendas para **20** produtos e **10** zonas são representados pela matriz **S(20,10)** para o primeiro período e pela matriz **T(,)** para o segundo período. A alteração em % é de ainda outra matriz, **I(,)**. Para começar, considere um elemento **(1,1)**, que contém as vendas do produto 1 na zona 1. A variação percentual de vendas é calculada da seguinte forma:

I(1,1)=((T(1,1)−S(1,1))×100)/S(1,1)

Para se realizarem estes mesmos cálculos para todas as combinações possíveis de **200** produtos/zonas, a fórmula é encaixada em dois *loops* que, em conjunto, percorrem as matrizes extraindo cada elemento de sua vez:

```
10 FOR M=1 TO 10
20 FOR N=1 TO 20
30 LET I(N,M)=((T(N,M)−S(N,M))×100)/S(N,M)
40 NEXT N
50 NEXT M
```

Podem surgir problemas nos círculos se os contadores fi-

carem positivos, negativos ou iguais a 0 em resposta aos valores atribuídos às variáveis.

O programa seguinte, que filtra todos os números superiores a 10 para um grupo numérico, C(N), apresenta-os em ordem ascendente:

```
10 INPUT A
20 FOR N=1 TO 9: LET A=A+1: LET C(N)=A: NEXT N
30 FOR N=1 TO 9
40 IF C(N)>10 THEN GOTO 60
50 NEXT N
60 PRINT N
```

Se não houver números inferiores a N (isto é, se A não for colocado em 0 na linha 10), o círculo N conta de 1 a 9, com a consequência de que a linha 60 imprime N, o número de elementos inferiores a 10, como 10, enquanto a resposta deveria ser 9. Se existirem 5 números inferiores a 10 (isto é, colocando A em 5), o círculo pára na linha 40, e a resposta correcta, N=5, é impressa pela linha 60.

Agora, estudemos o próximo programa, que determina a alteração em vendas, D( ), de um mês para o seguinte. O período sobre o qual recaem os cálculos é definido por A e B:

```
10 LET S(4)=16: LET S(3)=13: LET S(2)=4: LET S(1)=1
20 INPUT A: REM Início da série
30 INPUT B: REM Fim da série
40 FOR N=A TO B
50 LET D(N)=S(N)-S(N-1)
60 NEXT N
70 FOR N=1 TO 4: PRINT D(N): NEXT N
```


Se A assumir o valor 0 ou 1, o programa "rebenta" ao tentar descobrir um elemento de S( ) negativo ou igual a 0. Deste modo, se os meses forem numerados 1, 2, 3, etc., o programa não aceitará uma entrada do tipo A=1.

Como se torna evidente, este simples programa será facilmente corrigido realinhando os subscritos e os contadores. Em programas mais complexos, o perigo reside no facto de que um realinhamento para se resolver um problema numa secção criará outras dificuldades em locais bem distintos do programa.

CORRENTES

Por vezes, a lógica de um programa determina que as matrizes ou grupos têm de ser processados numa sequência que não pode ser estabelecida por qualquer *loop*. Por exemplo, em lugar de se listar um grupo na sequência 1 a 6 em saltos de 1 unidade, pode ser necessário contar os diversos elementos em ordem irregular, como 6, 2, 5, 4, 3, 1.

Em casos como este, temos de nos servir de uma técnica completamente diferente, em que a sequência de contagem é definida por outro grupo, utilizado como dispositivo de indexação. Este procedimento é ilustrado pelo exemplo seguinte, que extrai e imprime elementos do grupo Z$(6) numa sequência irregular:

```
10 LET C(6)=2
20 LET C(2)=5
30 LET C(5)=4
40 LET C(4)=3
50 LET C(3)=1
60 LET C(1)=0
70 LET H=6
80 IF H=0 THEN GOTO 120
```

```
90 PRINT Z$(H)
100 LET H=C(H)
110 GOTO 80
120 STOP
```

Na primeira passagem, a linha 90 imprime Z$(6). A linha 100 encarrega-se então de estabelecer H=C(6), que tem o valor 2. Na segunda passagem, executa-se a impressão de Z$(2), também pela linha 90. Este circuito é repetido até que Z$(3) seja por sua vez imprimido, e H é ligado a C(1), cujo valor é zero, na linha 100. Este processo provoca a finalização do círculo, na linha 80.

O grupo C(6) que normalmente apareceria numa ordem subscrita começando em C(1) e terminando em C(6) é um exemplo de cadeia.

Aos valores de cada elemento de C dá-se o nome de «ponteiros». O início da cadeia é dado pela linha 60, que estabelece H em 6, e o fim da mesma acontece quando H=0 em C(1).

O conceito dos ponteiros servirá como ferramenta deveras útil para manipular a ordem segundo a qual pretendemos listar matrizes e grupos. Tem ainda a particular virtude de permitir que, em lugar de se alterar a posição de cada elemento — através do incómodo processo anteriormente descrito — se possa única e simplesmente alterar o grupo em cadeia, caso se queira rearranjar a sequência de toda a matriz ou grupo. A armazenagem primária dos dados, isto é da própria matriz, permanece a mesma.

As cadeias tornam-se de importância vital quando temos de executar um certo número de operações independentes de saída de dados. Exemplo típico é a criação de uma série de relatórios tabulados a partir de dados básicos comuns, com as respostas listadas em diferentes ordens em cada relatório. Em lugar de se resolver toda a matriz, imprimir os resultados, voltar a trabalhá-la e novamente imprimir os dados obtidos, e assim por diante... dão-se instruções de impressão directas através de

uma série de grupos encadeados, que vão definir a ordem segundo a qual os dados de cada relatório têm de ser extraídos da matriz, para posterior impressão.

FACILIDADE DE TRABALHO E PREVENÇÃO DE ERROS

O critério mais óbvio — e mais importante — no julgamento da qualidade de qualquer programa reside no facto de ele ser fácil de utilizar e estar liberto de falhas. No peculiar calão da informática, estes conceitos são descritos como "amigo do utilizador" e "à prova de erros".

Os exemplos seguintes ilustram o que pretendemos dizer com estas expressões:

— O programa pede que se introduza uma informação, a data por exemplo. Escrevemo-la e o *écran* fica em branco. Espera-se um bocado, mas nada sucede. Que se pode fazer? Ou esperar mais ou partir do princípio de que os dados introduzidos não foram aceites, pelo que só resta reiniciar o programa. A resposta para este insucesso é: não sei o que se passou. O programa não é "amigo do utilizador".

— O programa diz que se escreva o número de clientes de determinada zona. Por qualquer motivo, pensamos dever escrever primeiro o nome da zona, e só depois o número de clientes. Digitamos N.W.34 e carregamos em ENTER. Pouco tempo depois, a máquina emite um *beep* e surge uma mensagem de erro. O que sucedeu foi que o computador estava à espera da introdução de um número, mas introduziu-lhe letras; estas foram aceites, mas mais adiante o programa "rebentou" porque tentou processar uma variável literal como se fosse uma variável numérica. Em resumo, este programa não é "à prova de erros".

O mais vulgar é haver erros deste tipo quando introduzimos dados num computador, e a única forma de os evitar é ter o máximo de cuidado quando escrevemos as partes do programa que efectuam a interface com o teclado.

É claro que ao escrevermos programas para uso privado não temos de nos preocupar com o modo como o público lidaria com esses mesmos programas. Podemos até pensar que a prevenção de erros é perfeitamente desnecessária, pois ninguém ignorante da matéria fará correr as nossas listagens, nem se chegará perto do computador. Porém, nem tudo isto é evidente quando se escreveu o programa, e talvez esteja esquecido seis meses mais tarde, quando quisermos servir-nos dele mais uma vez.

O problema de se escreverem programas "amigos do utilizador" é o comprimento que acabam por adquirir. Como se verá mais adiante, são necessárias linhas e linhas de instruções só para ficarmos absolutamente seguros de que eliminámos toda e qualquer possibilidade de erro e de que só um tipo de entrada de dados será aceite, enquanto os outros serão ignorados. Na verdade, e com excepção dos programas de aplicações científicas e de engenharia, é normal que a maior parte de uma listagem seja preenchida com rotinas de entrada de dados e para guiar o utilizador ao longo do trabalho que pretende executar.

Para ilustrarmos os problemas em jogo, consideremos a tarefa de atribuir um valor numérico à variável **X**, em que o valor de **X** tem de estar compreendido entre **1** e **99**.

A solução mais simples é através do comando **INPUT**:

```
10 INPUT X
```

e, para verificarmos que a entrada se efectivou, juntemos as linhas:

```
20 CLS
30 PRINT X
40 STOP
```

Fazendo **RUN**, o computador esperará, na linha **10**, que seja introduzido o valor a dar a **X**. Digamos que se escreve **22**; então o programa avançará para limpar o *écran* (CLS) e imprimirá o valor de **X** — isto é, 22.

É claro que o programa aceitará qualquer número — **220**, **−220** ou **22.22** —, pois não sabe que só deveria reconhecer números inteiros compreendidos entre 1 e 99. Se, contudo, introduzirmos uma letra em lugar de um número, o programa opor-se-á: recusa-se a aceitar a letra e imprime uma mensagem de erro. A variável **X** tem de ser constituída por um número, e nunca por uma letra.

A listagem melhorará muito se soubermos aquilo que temos de escrever, o que se consegue muito simplesmente inserindo uma nova instrução na linha **10**, como por exemplo:

```
10 PRINT "ESCREVA UM NUMERO INTEIRO ENTRE
1 E 99"; : INPUT X
```

Se fizermos agora o **RUN**, esta instrução será impressa pela linha **10**, e o computador ficará à espera, como dantes, pelo valor a introduzir. Este programa já se pode considerar "amigo do utilizador", mas nada se fez para o tornar à prova de erros.

Como atrás referimos, o problema com a linha **10** é que, se se escrever uma letra, esta é imediatamente rejeitada; no entanto, isto é só parte do erro: o processo de rejeição não tem lugar no programa, mas sim no próprio computador — ou, para sermos mais específicos, no programa de linguagem BASIC que faz parte do computador. Tal significa que o programador não se apercebe do sucedido. O controle passa para a máquina, e não há forma de se ultrapassar esta situação. Este facto é nitidamente indesejável, pelo que temos de evitar possibilidades do género.

Método mais correcto será o de fazermos com que o computador aceite qualquer tipo de entrada, quer esta seja correcta ou não, e pôr o programa a analisar a situação, decidindo se pode prosseguir ou se deve parar enviando, por exemplo, um sinal de aviso de erro. O primeiro objectivo consegue-se com a substituição de **X** por **X$** na linha **10**. O computador passa a considerar qualquer entrada como variável literal, pelo que não rejeitará nenhum sinal enviado do teclado. O segundo objectivo, que consiste na construção de restrições no próprio pro-

grama, de modo a que este rejeite entradas inaceitáveis, consegue-se pela adição de mais duas linhas:

```
12 IF VAL (X$) < 1 OR VAL (X$) > 99 THEN GOTO 10

14 IF VAL (X$) < > INT (VAL (X$)) THEN GOTO 10
```

A linha 12 rejeita a entrada se o valor não estiver compreendido entre 1 e 99. Incluem-se as entradas que não sejam constituídas por números, pois o BASIC avalia as variáveis não numéricas como 0. A linha 14 rejeita números não inteiros, comparando o valor da entrada com o seu correspondente inteiro (INT); se não forem iguais, a entrada nunca pode ser um número inteiro. As entradas rejeitadas fazem com que o programa volte à linha 10, para o operador poder corrigir o erro. Por fim, e dado que o objectivo é a entrada do valor de X, X$ é avaliado e tornado igual a X, da forma:

```
16 LET X = VAL (X$)
```

A partir deste momento, o programa não pode falhar por causa de entradas incorrectas. Se suceder uma destas, regressa ao início e aguarda por novos dados. Contudo, mesmo assim o operador pode não se aperceber porque é que as suas entradas continuam a não ter aceitação, pelo que convém acrescentar um aviso de erro, que pode ser feito através da seguinte modificação:

```
IF VAL (X$) < 1 OR VAL (X$) > 99 THEN PRINT
"FORA DOS LIMITES": GOTO 10

14 IF VAL (X$) < > INT (VAL (X$)) THEN PRINT "NU-
MERO NAO INTEIRO": GOTO 10
```

Agora, seremos sempre avisados de que efectuámos uma entrada errada. Através do comando **BEEP** podemos ainda realçar o aviso, que passará assim a ser acompanhado por um sinal sonoro.

O programa passou a conter, pois, todos os requerimentos lógicos pretendidos, mas mesmo assim a apresentação em *écran*

não está em condições. Se introduzirmos dados errados, o televisor fica a parecer-se com uma selva de mensagens de erro, pedidos de introdução de informações e de entradas já colocadas pelo operador. Temos portanto de controlar também as imagens que deverão ser apresentadas.

Para tal, vamos fixar o local do *écran* no qual queremos que isso suceda, servindo-nos da instrução **PRINT AT:**

```
5 CLS

10 PRINT AT 5,0; "ESCREVA UM NUMERO ENTRE 1
E 99"; 11 INPUT X$

12 IF VAL (X$) < 1 OR VAL (X$) > 99 THEN PRINT AT
15,1; "FORA DOS LIMITES": GOTO 10

14 IF VAL (X$) < > INT (VAL (X$)) THEN PRINT AT
15,1; "NUMERO NAO INTEIRO": GOTO 10

16 LET X = VAL (X$)

20 CLS

30 PRINT AT 0,5; "A ENTRADA E:"; X

40 STOP
```

Este programa está muito melhor, mas ainda não se pode considerar perfeito. Se uma introdução for rejeitada e a execução regressar à linha 10, os dados introduzidos ficam no fim da linha, com o cursor sobre o primeiro dígito que escrevemos:

```
ESCREVER UM NUMERO ENTRE 1 E 99 222
```

Ficará mais apresentável se conseguirmos retirar a entrada incorrecta. Para tal, o início do programa tem de ser reorganizado, de modo a que as linhas 12 e 14 regressem a uma nova linha 10. Esta apaga quaisquer entradas prévias, através da impressão sobre as localizações do *écran* em que apareciam os números errados, que substitui por uma série de espaços brancos.

Passa-se problema idêntico com as mensagens de aviso. Se

um erro da linha 12 é seguido por outro da linha 14, a mensagem emitida por esta última não oblitera a da linha 12. Se virmos o problema ao contrário — isto é, erro da linha 14 seguido por um da linha 12 —, tudo estará bem, porque neste caso a última mensagem é mais comprida que a da linha 14. Uma solução possível consiste em aumentar artificialmente o comprimento do aviso da linha 14, adicionando-lhe uma linha de espaços em branco depois da palavra **INTEIRO.**

Vejamos finalmente o programa completo:

```
5 CLS
8 PRINT AT 5,0; "ESCREVA UM NUMERO ENTRE 1 E
99";
10 PRINT AT 5,31; "  "; :PRINT AT 5,31
11 INPUT X$
12 IF VAL (X$) <1 OR VAL (X$)> 99 THEN PRINT AT
15,1; "FORA DOS LIMITES": GOTO 10
14 IF VAL (X$) < > INT (VAL (X$)) THEN PRINT AT
15,1; "NUMERO NAO INTEIRO": GOTO 10
16 LET X = VAL (X$)
20 CLS
30 PRINT AT 0,5; "A ENTRADA E:"; X
40 STOP
```

Estamos assim perante um programa de entrada de dados bastante aceitável; mas para o concebermos passámos de 4 linhas para 10, e continua apenas a aceitar um número! Temos de perdoar aos que queiram desistir desta ideia de "amizade com o utilizador", se para a realizarmos temos de despender todo este esforço.

Felizmente, isto não é sempre necessário, pois todas as rotinas de entrada de dados seguem um padrão semelhante, o que

as torna nas candidatas ideais para a estandardização e compressão em sub-rotinas.

SUBSÍDIOS PARA UMA SUB-ROTINA UNIVERSAL DE ENTRADA DE DADOS

As rotinas de **INPUT** que se seguem destinam-se a satisfazer os requisitos de uma sub-rotina universal para introdução de dados. Todas trabalham de acordo com os seguintes princípios:

— Os dados são pedidos por uma linha de texto apresentada no *écran*. Por exemplo, "Introduza o seu nome:"

— O operador pode sempre responder ao pedido numa das três seguintes formas:

1) Premir a tecla do **RETURN** para fazer entender que necessita de ajuda. Recebe então uma mensagem que lhe diz o que é que se supõe que vai realizar.

2) Para abandonar o programa, basta premir a tecla #.

3) Introduzir os dados que lhe são pedidos. O programa verifica se a entrada obedece às especificações e, caso contrário, apresenta uma mensagem de erro e volta a pedir a introdução de novos dados.

Listamos aqui quatro versões de **INPUT.** As linhas 10 a 70 e 300 a 390 são comuns a todas elas, só variando as linhas 100 a 200, de modo a que possam lidar com os quatro tipos diferentes de dados que a seguir se indicam:

— Entrada de números **(INPUT NUMEROS)**

— Entrada de barra de espaços (tal como em "prima **SPACE** para continuar") — **(INPUT SPACE)**

— Sim ou Não **(INPUT S/N)**

— Entrada de letras **(INPUT LETRAS)**

Todos os programas foram concebidos com a intenção de se tornarem em sub-rotinas, chamadas pelo programa principal

imediatamente a seguir a qualquer pedido de entrada de dados. Eis um possível segmento típico do programa base:

```
3022 PRINT AT 5,0; "INTRODUZA NUMERO DO CLIENTE";

3024 LET D = 6: GOSUB 10: REM Sub-rotina de entrada começa na linha 10
```

A sub-rotina **INPUT** recebe a entrada de dados, verifica a sua validade e reenvia esses valores para o programa base através do **RETURN.**

Dentro da listagem principal, as quatro categorias de saída — isto é, **AJUDA, DADOS VALIDOS, DADOS INVALIDOS E ABANDONO** — distinguem-se umas das outras pela bandeira **RF**, a que se chama "bandeira de retorno" e que pode assumir os valores 0, 1, 2 ou 3.

As linhas seguintes do programa principal extraem, com o valor RF, as quatro diferentes categorias de saída, e para chamar a resposta adequada:

```
3025 IF RF = 1 THEN GOTO 3030

3026 IF RF = 2 THEN GOTO 3032

3027 IF RF = 3 THEN GOTO 3034

3028 REM RF = 0 provoca uma mensagem e depois encaminha o programa para 3022

3030 REM RF = 1 significa Entrada Valida — avance para a fase seguinte

3032 REM RF = 2 significa Entrada não válida — explica a razão e regressa a 3022

3034 REM RF = 3 liga à rotina de abandono (EXIT)
```


Considerem-se agora as próprias sub-rotinas. A finalidade da **INPUT NUMEROS** é a de introduzir uma série de dígitos, que podem incluir o ponto decimal e o sinal menos. O programa permite ainda restringir os limites e os tipos de números que deverão ser admitidos.

```
    5 REM Programa de Carga
        de Dados de Entrada
   10 LET tv=0: LET t$="": LET c
h=0
   20 LET a$=INKEY$: IF a$<>"" TH
EN  GO TO 20
   25 LET a$=INKEY$: IF a$="" THE
N  GO TO 25
   30 IF CODE (a$)=13 AND ch=0 TH
EN  LET rf=0: GO TO 390
   50 IF CODE (a$)=12 THEN  GO TO
 330
   60 IF CODE (a$)=13 THEN  LET r
f=1: GO TO 380
   70 IF CODE (a$)=35 AND ch=0 TH
EN  LET rf=3: GO TO 390
  300 GO TO 20
  310 LET t$=t$+a$: PRINT a$;: LE
T ch=ch+1
  320 GO TO 20
  330 IF ch=0 THEN  GO TO 20
  340 LET ch=ch-1: PRINT CHR$ (8)
;" ";CHR$ (8);
  350 IF ch=0 THEN  LET t$="": GO
 TO 20
  360 LET t$=t$(1 TO LEN (t$)-1)
  370 GO TO 20
  380 LET tv=CODE (t$)
  390 RETURN
```

```
  5 REM Entrada de Numeros
 100 IF ch>d THEN  LET rf=2: GO
TO 390
 110 IF a$="-" AND ch=0 THEN  GO
 TO 310
 120 IF a$="." THEN  GO TO 310
 130 IF a$>="0" AND a$<="9" THEN
  GO TO 310
```

Mal a sub-rotina é chamada, o computador espera na linha 20 que se prima uma tecla. Então a variável **A$** é regulada para o carácter introduzido, e é o valor **ASCII** de **A$** que determina a acção a tomar. As linhas 30, 60 e 70 provocam a saída do programa principal se **A$** for um dos caracteres de controle permitidos através dos quais o operador pode assinalar que pretende ajuda, que terminou a entrada de dados, ou que quer abandonar o processamento. A linha 100 verifica se o número de caracteres introduzido não excede o máximo permitido pela variável **D**, estabelecida pelo programa principal na linha 3024. Esta verificação consegue-se comparando **D** com **CH**, contador que indexa no sentido crescente, uma unidade de cada vez, sempre que se introduz um carácter. Se **D** for excedido, provoca-se a saída **RF = 2** na linha 390. A linha 110 aceita o sinal menos, mas só se este for o primeiro carácter. A linha 120 reconhece o ponto decimal, e por último a linha 130 verifica se a entrada é um número entre 0 e 9.

Se, mesmo assim, a entrada não provocou um desvio do programa principal, a linha 300 executa um círculo de regresso ao início da sub-rotina da linha 20, assegurando deste modo que o computador não registe nenhum carácter para além dos especificados no programa.

No caso da entrada ser constituída por um carácter válido, verifica-se o desvio para a linha 310, em que **T$** acumula cada entrada à medida que estas se sucedem, e **CH** aumenta uma unidade, antes do regresso à linha 20 para se aguardar nova

introdução de um número. Quando o operador acabou de introduzir o dígito, prime **RETURN**. Esta fase é recebida pela linha 60 e provoca um desvio até à linha 380, depois de estabelecer **RF em 1. TV** é regulada para o valor da variável acumulada **T$**, e a linha 390 efectua o regresso ao programa principal com **RF=1**, o que significa que teve lugar uma entrada válida.

Eis agora o que se passa quando o operador pede uma mensagem, premindo a tecla **RETURN** (e só esta): a linha 30 detecta que **A$** é **RETURN** (código ASCII 13), e que se trata do primeiro carácter a introduzir (**CH=0**); troca-se assim o estabelecimento da bandeira **RF** em 0, e realiza-se de imediato o salto para **RETURN** na linha 390. Desta vez, **RF=0** significa que se requereu mensagem de ajuda.

De modo semelhante, se o operador premir **#**, a linha 70 detecta que **A$** é o código ASCII 35, e estabelece **RF** em 3 antes de saltar para o **RETURN**.

No caso de começarmos a introduzir dados e, às tantas, os pretendermos alterar, teremos de premir a tecla da seta esquerda para recuar, o que é o procedimento normal. A linha 50 detecta que se premiu a seta esquerda, enquanto que a linha 330 verifica se já se introduziu algum carácter; se não é esse o caso (**CH=0**), a entrada é ignorada. Se já houver caracteres presentes, a linha 340 efectua o retrocesso de uma letra *(backspace)* da seguinte forma: primeiro, o valor de **CH** diminui de uma unidade; o cursor recua então um espaço, e o que se tenha imprimido nesse local é apagado (**PRINT " ";**). O apagamento da entrada provoca o avanço do cursor, de modo que temos de o fazer recuar de novo. A linha 350 encarrega-se da situação em que todas as entradas foram apagadas, evitando que **CH** fique negativo. A linha 360 elimina a última entrada do acumulador **T$**, antes de voltar a adquirir novo carácter via linha 370.

A introdução de letras é processada de forma muito parecida pela sub-rotina **INPUT LETRAS**. Aqui, a verificação da linha 110 é estabelecida de modo a admitir os caracteres de **A** a **Z**, e nada mais.

```
   5 REM ****INPUT LETRAS****
 100 IF ch>d THEN  LET rf=2: GO
TO 390
 110 IF CODE (a$)>64 AND CODE (a
$)<91 THEN  GO TO 310
```

A sub-rotina **INPUT SPACE** é consideravelmente mais simples — não é necessário incluir **CH**, pois só se introduz um carácter, e a rotina de correcção da seta esquerda torna-se supérflua. Só são mantidas na listagem para salvaguarda da consistência.

```
   5 REM ****INPUT ESPACOS****
 100 IF a$=" " AND ch=0 THEN  LE
T rf=1: GO TO 390
```

A última das sub-rotinas é **INPUT Y** ou **N**, listada a seguir.

```
   5 REM ****INPUT SIM/NAO****
 100 IF A$="s" AND ch=0  THEN  L
ET t$="1": GO TO 310
 110 IF a$="n" AND ch=0 THEN  LE
T t$="2": GO TO 310
```

Como é óbvio, podemos escrever qualquer das versões de **INPUT** de modo a processar todas as restrições lógicas que se pretendam na entrada de dados. Se quisermos introduzir só números inteiros, faremos com que o programa rejeite o ponto decimal, mas podemos estabelecê-lo de forma a aceitar um único ponto decimal, pelo emprego de uma bandeira.

```
120 IF A$="." AND DF=0 THEN LET DF=1: GOTO 310
122 IF A$="." AND DF><0 THEN GOTO 20
```

Quando pretendemos uma resposta sim ou não, a lógica do programa terá de ser tal que só o **S** ou o **N** (CÓDIGOS ASCII 89 e 78) sejam aceites; o raciocínio será sempre o mesmo para toda e qualquer restrição que queiramos determinar.

É mesmo possível fazer com que as diferentes versões deste programa assumam a forma de sub-rotinas dentro da rotina principal (**INPUT**), chamando-as sempre que necessário através de uma bandeira estabelecida no corpo base. Contudo, e a menos que o espaço de memória seja de importância vital, é melhor mantermos as coisas neste grau de simplicidade, pois as rotinas alocatáveis não são tão fáceis de seguir.

EXTRACÇÃO DE DADOS

A maneira mais natural de se escrever um programa é tentar imitar a forma como resolveríamos o problema se não possuíssemos um computador, porventura com a ajuda de qualquer calculadora electrónica. Regra geral, nada há de errado nesta forma de raciocínio — de resto, que outra coisa poderíamos fazer? Ocasiões há, porém, em que mesmo a maneira mais simples de se resolver um problema não é a adequada para o computador. Um bom exemplo desta realidade verifica-se com o problema da extracção de dados.

Imaginemos uma lista de dados que compreenda os números 5, 4, 6, 7 e 3, que pretendemos escrever por ordem decrescente. Para resolver o problema de cabeça, basta olhar para todos os números, retirar o maior, depois o que se lhe segue em valor, e assim por diante, até que todos fiquem na ordem pretendida.

Um computador terá enorme dificuldade em executar a tarefa desta maneira, pois não lhe é possível comparar mais do que duas grandezas de uma só vez; portanto, comparará os dois primeiros números e descobrirá qual deles é o maior, depois comparará os dois números seguintes, e assim sucessivamente; a função do programador consiste em organizar esta limitada capacidade por forma a que a máquina consiga imprimir a lista pretendida.

Designamos por extracção de bolas *(buble sort)* a técnica a empregar para este fim, que se serve da capacidade dos computadores para compararem dois números numa sequência que percorre toda a lista a analisar. A sequência inicia-se pela sepa-

ração do primeiro par de números, que são colocados na ordem correcta — isto é, numa ordem descendente o computador coloca o número maior em primeiro lugar. No caso que estamos a analisar, os primeiros dois números (5 e 4) já se encontram na ordem pretendida. O passo seguinte consiste em retirar o número maior e pegar no par seguinte (4 e 6), que por sua vez é colocado em ordem descendente, ficando o 6 a anteceder o 4. Retirado o algarismo 6, o par seguinte fica a ser constituído por 4 e 7; a sua ordenação coloca-se sob a forma 7 e 4. O último par, resultante de todos os arranjos anteriores, é formado por 4 e 3, que já estão em ordem descendente. Neste momento, a lista assume a forma 5, 6, 7, 4 e 3 — está melhor, mas ainda longe do que se pretende.

A fase seguinte do processo consiste na repetição da extracção de pares de algarismos para toda a lista, com a excepção de que o último (isto é, o mais pequeno, já colocado no local certo) fica de fora. Esta reordenação produz nova lista, em que mais uma vez só o último número é posto no local apropriado. Se todo o processo for repetido um número de vezes suficiente, todos os algarismos serão colocados em ordem descendente, à excepção dos dois do topo — e o computador não terá qualquer dificuldade em organizar este último par. De maneira geral, o número de passagens de extracção é sempre inferior em uma unidade ao total de elementos da lista a ordenar.

O programa listado no fim deste capítulo fornece uma rotina de extracção de bolas tanto para a ordem ascendente como para a descendente. Processa também variáveis literais e mostra como se podem colocar nomes por ordem alfabética através da **matemática de variáveis de texto**. A comparação básica dos pares tem lugar na linha 1420: **C$(K)** é comparado com **C$(K+1)**, para se ver qual é o maior. O que na realidade sucede é a comparação dos códigos da primeira letra de cada uma das variáveis literais. Como esse código atribui números às letras, em ordem ascendente de A a Z (o código para A é 65, para Z é 90), a comparação das duas variáveis literais vai colocá-las em ordem alfabética.

Os dados a extrair são colocados nas linhas 800 a 820, através de declarações **DATA**, sendo depois lidos para o grupo **C$( )**, no qual N representa o número de dados. Seria possível introduzir números em lugar de letras, mas neste caso teriam de ser transformados antecipadamente em *strings* — ou seja, "1", "2", etc. Mesmo assim a extracção continuaria a ser efectuada por meio da "aritmética de variáveis literais" acima descrita, de modo que os números sairiam de acordo com o valor do primeiro carácter, e só uma lista de algarismos todos com valores inferiores a 10 seria colocada na sequência correcta. Para que tal suceda, **C$( )** tem de ser transformado num grupo numérico, **C( )**, e o programa corrigido de modo a poder processar variáveis numéricas.

A escolha entre ordem ascendente ou descendente é feita nas linhas 300 e 350, e a extracção de bolas em si mesma começa na linha 400, em que se estabelece um *loop* (contador K) para repetir o ciclo de extracção, actuando N-1 vezes. A comparação de cada par de entradas é executada por outro círculo, intercalado dentro do primeiro, com início na linha 450. Este *loop* conta "para trás", desde K até 1 em saltos de −1. O factor determinante da ordem ser ascendente ou descendente reside na forma como se especifica a comparação dos pares. A linha 470 (**C$(j)>=etc.**) determina a ordem ascendente, enquanto na linha 490 se estabelece a ordem inversa (**C$(j)<=etc.**). O resultado da comparação de pares pode indicar que os números já estão na ordem correcta, caso em que o programa avança para o par seguinte, ou que tal não sucede, caso em que os números têm de ser "trocados". A troca é feita nas linhas 500 e 520, em que **T$** é um ponto de armazenamento temporário de um dos valores. A extracção termina quando o ciclo de comparações se completa, na linha 550. Por fim, a lista já ordenada é impressa, e a linha 610 avisa que o processamento terminou.

```
100 REM Rotina de Extracoao
180 DIM c$(100,11)
230 LET n=7
240 REM Leitura de dados para o
            array c$
250 FOR j=1 TO n
260 READ m$: LET c$(j)=m$
270 NEXT j
280 REM Busca Ascendente ou
            Descendente
290 PRINT : PRINT : PRINT
300 PRINT "Qual a ordem de pesq
uisa? (a/d)": INPUT a$
350 IF a$<>"a" AND a$<>"d" THEN
 GO TO 290
360 REM Inicio da Pesquisa
400 FOR k=1 TO (n-1)
410 IF a$="a" THEN  GO TO 440
420 IF c$(k)>=c$(k+1) THEN  GO
TO 540
430 IF a$="d" THEN  GO TO 450
440 IF c$(k)<=c$(k+1) THEN  GO
TO 540
450 FOR j=k TO 1 STEP -1
460 IF a$="a" THEN  GO TO 490
470 IF c$(j)>=c$(j+1) THEN  GO
TO 540
480 IF a$="d" THEN  GO TO 500
490 IF c$(j)<=c$(j+1) THEN  GO
TO 540
500 LET t$=c$(j)
510 LET c$(j)=c$(j+1)
520 LET c$(j+1)=t$
530 NEXT j
540 NEXT k
550 REM Fim de Pesquisa
580 FOR i=1 TO n
590 PRINT : PRINT c$(i)
```



```
600 NEXT i
610 PRINT : PRINT "Pesquisa ter
minada"
800 DATA "Joao","Tiago","Maria"
810 DATA "Guilhermina","Rodrigo
","Sara"
820 DATA "Raul"
```

Capítulo 4

Ajustador
Ajustamento da tendência de vendas

INTRODUÇÃO

Os números indicativos de tendências de resultados, quando referidos a determinado período de tempo, são sempre bons para elevar a temperatura da discussão entre dois colegas de negócio. Para uns, qualquer gráfico de vendas em franca subida é o justo prémio do esforço excepcional desenvolvido enquanto que para outros não passa de mera indicação de que os clientes recomeçaram a comprar. Uma queda súbita pode anunciar iminente falência, ou simplesmente revelar que as pessoas ainda não regressaram das férias de Verão.

Este tipo de interpretação não abalizada dos indicadores em nada contribui para a compreensão das realidades. As opiniões expendidas podem estar certas, mas nada têm a ver com os números sujeitos a análise. Se os dados sobre evolução de tendências foram concebidos para prestarem informações, estas têm forçosamente de ser escalpelizadas primeiro e só depois discutidas.

Ponto de vista mais cínico, mas nem por isso menos realista, é o do velho adágio sobre mentiras, grandes falsidades e dados estatísticos. Se tivermos de enfrentar pessoas que se sabe à partida irem agarrar-se a opiniões de momento, o melhor é munirmo-nos de outro conjunto de dados alternativos, capazes de demonstrar que há várias interpretações igualmente válidas sobre um mesmo assunto. E, se o caso se tornar sério, é bom que comecemos a analisar os indicadores apresentados pela oposição!

Qualquer discussão exaustiva da análise de tendências fica fora dos objectivos desta obra; existe bastante literatura especializada e mesmo programas informáticos bem conhecidos sobre tal matéria. Contudo, dispomos ainda de muito espaço para falarmos dela, especialmente se considerarmos os factores bem conhecidos do público, tais como a inflação e os efeitos provocados pelos diferentes dias úteis em cada mês. Se ajustarmos uma tendência de vendas para estes dois aspectos, a realidade apresentada será de certeza bem diferente; talvez os indicadores fiquem ainda piores mas a explicação tornar-se-á bastante mais fácil.

O programa "Ajustador" processa a variação das vendas durante um mês e ajusta-a tomando em consideração a inflação, os dias úteis ou ambos estes factores em simultâneo.

O conceito baseia-se na introdução do número de vendas mensal, até um máximo de 36 valores, juntamente com um índice de preços e o respectivo número de dias de trabalho em cada mês. Para ajustar os resultados à inflação, convertem-se todos os indicadores quantitativos de vendas no que eles representariam se não se tivesse verificado nenhuma inflação durante o período. Por outras palavras, cada valor de vendas é ajustado em função do "preço em vigor no início da variação" dividido pelo "preço em vigor na altura da transacção". Transformando este raciocínio numa notação matemática:

VENDAS AJUSTADAS NO MES 10 = VENDAS NO MES 10 × (INDICE DE PREÇOS NO MES 1/INDICE DE PREÇOS NO MES 10)

Se o índice de preços no mês 1 for 100 e o índice de preços no mês 10 for 200, então os custos de aquisição duplicaram em 10 meses. Portanto, para ajustarmos as vendas do mês 10 ao nível em que estariam no primeiro mês, devemos multiplicá-las por 100/200.

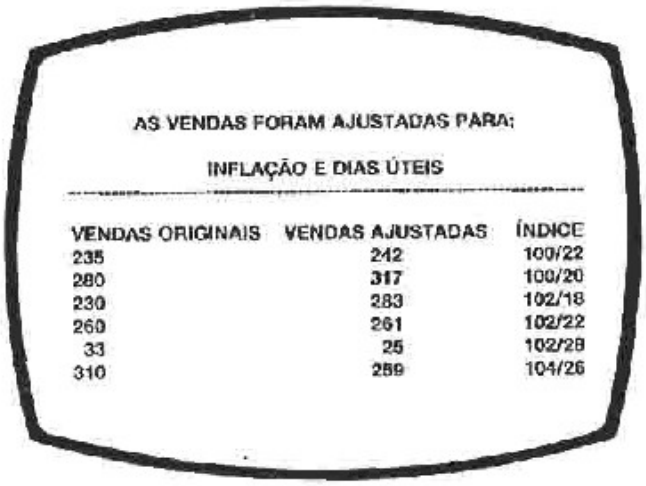
Para efectuarmos o ajustamento em relação aos dias de trabalho, temos de calcular quais as vendas mensais se o mês contivesse um doze avos do número de dias úteis do ano. Ou seja, as vendas são convertidas naquilo que seriam se todos os meses tivessem exactamente o mesmo comprimento. Estamos

aqui perante um cálculo em duas etapas — a primeira determina o número médio de dias de trabalho por mês, enquanto a segunda multiplica as vendas de cada mês pela razão "número real de dias úteis/número médio de dias úteis". Mais uma vez, em notação matemática:

NUMERO MEDIO DE DIAS UTEIS=SOMA DE DIAS UTEIS DE CADA MES/NUMERO DE MESES

VENDAS AJUSTADAS NO MES=VENDAS NO MES×(DIAS UTEIS REAIS/MEDIA DE DIAS UTEIS)

Para efectuarmos o ajustamento tanto para a inflação como para os dias de trabalho, basta aplicarmos os valores obtidos do ajustamento da inflação nos cálculos relativos ao ajustamento para os dias úteis, no lugar das "vendas do mês".



AS VENDAS FORAM AJUSTADAS PARA:

INFLAÇÃO E DIAS ÚTEIS

VENDAS ORIGINAIS	VENDAS AJUSTADAS	ÍNDICE
235	242	100/22
280	317	100/20
230	283	102/18
260	261	102/22
33	25	102/28
310	259	104/26

DESCRIÇÃO DO PROGRAMA

O programa "Ajustador" incorpora o procedimento completo de entrada de dados descrito no capítulo 3, com as sub-rotinas **7000**, **8000** e **8500** encarregadas de processar as diferentes categorias de entradas.

A sub-rotina **9000** é uma rotina estandardizada de "ajuda", chamada pelo programa principal em resposta a um regresso **RF=0** de uma das sub-rotinas de entrada de dados. Do mesmo modo, a sub-rotina **9500** é um procedimento normalizado de "abandono", activado como resposta a um regresso **RF=3**. As duas últimas linhas do *écran* ficam reservadas para as mensagens de auxílio.

O modo como trabalham estas duas rotinas pode ser apreciado logo no início da listagem: a linha **1004** pede o número de meses a processar, e chama a sub-rotina **7000**, que aceita a entrada desse valor (**D=2** limita a entrada a 2 caracteres). As linhas **1005** a **1007** tratam do regresso da sub-rotina e determinam três desvios possíveis, conforme o valor de **RF**. **RF=0** liga directamente à linha **1008**, que chama a sub-rotina **9000**; esta apresenta a mensagem **B$**, que acabou de ser definida. Repare nas linhas **9002** e **9014**, que limpam por duas vezes o fundo do *écran*, *a*) antes da apresentação da mensagem e *b*) depois da mensagem ter provocado a resposta adequada.

RF=1 é o caminho de saída para uma entrada válida — isto é, formada por um número de dois caracteres —, e leva-a à linha **1018**, em que se procede a nova verificação. Se o número enviado pelo teclado não passar nesta, aparece a correspondente mensagem imprimida pela linha **1020**, dando-se início de novo à rotina de entrada. Se tudo estiver correcto, os dados introduzidos são atribuídos à variável **NM** da linha **1018**, antes de se passar à fase seguinte, que começa na linha **1102**.

RF=3 constitui o passo de saída que chama a sub-rotina **9500**, na linha **1012**. Esta sub-rotina verifica em primeiro lugar se o operador pretende mesmo abandonar o programa, não se dê o caso de ter premido a tecla errada por engano; se confirmarmos tal erro de operação, o programa encaminha-se para **RETURN**,

e daí regressa ao início da rotina de entrada de dados, via linha 1014.

Depois de termos estabelecido o número de meses a analisar, dimensionam-se os grupos na linha 1102, preparando-os para o corpo principal de dados a entrar através do circuito FOR/ /NEXT que começa na linha 1104 e acaba na 1162. O *loop* conta de 1 até NM (o número de meses), e os dados relativos a cada mês são introduzidos em cada uma das voltas do círculo. Os três tipos de dados que constituem cada entrada — vendas, índice de preços e dias úteis — são aceites de forma semelhante à anterior, com início na linha 1108. Os dados das vendas são memorizados duas vezes, primeiro no grupo S(M) e depois A(M) (linha 1124); a explicação para esta aparente duplicação tornar-se-á evidente mais adiante. A entrada do índice de preços começa na linha 1126 enquanto que a dos dias úteis se realiza na linha 1144.

Depois de escritos todos os dados, o computador pede que se especifiquem qual das três possibilidades de análise possíveis se quer utilizar. A pergunta é feita e respondida nas linhas 1200 a 1224; consoante a resposta, a bandeira AC é estabelecida em 1, 2 ou 3, e o programa desvia-se para os cálculos apropriados.

Os cálculos para o ajustamento da inflação aparecem na sub-rotina 5000, os relativos aos dias úteis na 6000, e foram descritos no começo deste capítulo.

Agora há dois conjuntos de dados a armazenar — as "vendas em bruto" e as "vendas ajustadas" —, pelo que temos de criar um novo grupo, a que chamaremos A(M) e que, por conveniência de cálculo, vamos tornar igual a S(M) na linha 1124.

Terminado que seja o ajustamento de dados, dá-se início à impressão, na linha 1300; a selecção do título correcto faz-se entre as linhas 1304 e 1312, a linha 1314 sublinha-o e a linha 1316 imprime os cabeçalhos das colunas — que são sempre os mesmos independentemente do tipo de análise efectuado. A impressão efectua-se através de um círculo FOR/NEXT, com início na linha 1318 e término na 1330. Podemos ter um máximo de 36 linhas (NM) a imprimir, que obviamente não caberão num *écran* limitado a 20, como é o caso do *Spectrum*; como tal, a listagem tem de ser parada quando o *écran* fica cheio, passando das operações ao operador, até que este peça a continuação da apresentação. Este pormenor é controlado pelas variáveis L e M, que a linha 1302 estabelece, respectivamente, em 1 e 10. Se houver menos de 10 leituras (10>NM), M é regulada para o total daquelas. Uma vez dentro do círculo da linha 1318, imprimem-se as primeiras 10 linhas de texto, e no momento da chegada ao fim do *loop* a linha 1332 verifica se todo o texto foi apresentado na primeira passagem. Se ainda faltam dados, o operador recebe o pedido de premir a barra de espaços (SPACE) para poder ver a continuação da análise. Para tal, as duas variáveis são indexadas 10 unidades, no sentido crescente, pela linha 1346, e se o número de leituras por imprimir for inferior a dez, M é mais uma vez regulada para o valor de NM. O círculo de impressão inicia-se na linha 1304, e o processo continua até que todas as leituras tenham sido apresentadas. Dentro do círculo de impressão, a bandeira AC — que assume os valores 1, 2 ou 3 de acordo com a escolha feita anteriormente (linha 1322) — chama os adequados valores "ajustados", enquanto a linha 1328 trata da impressão tanto do índice de preços como do número de dias de trabalho, a partir do momento em que AC=3 (inflação e dias úteis) seja o valor seleccionado.

Depois de terminada a impressão, o operador tem duas escolhas: ou faz executar de novo toda a rotina, ou abandona o programa (linha 1348 e seguintes). A linha 1368 encaminha a execução quer para nova impressão (linha 1302) quer para o abandono (linha 9508).

```
AJUSTADOR
1000 REM Entrada dos meses
1002 CLS
1003 PRINT TAB (10);"AJUSTADOR":
 PRINT TAB (10);"=========="
1004 PRINT AT 5,5;"Quantos meses
 pretende?    ";: PRINT AT 5,30
;: LET d=2: LET de=0: GO SUB 700
0
1005 IF rf=1 THEN  GO TO 1018
1006 IF rf=2 THEN  GO TO 1012
1007 IF rf=3 THEN  GO TO 1012
1008 LET b$="Entre um numero int
eiro entre    1 e 36": GO SUB 90
02
1010 GO TO 1004
1012 GO SUB 9500
1014 GO TO 1004
1018 IF tv>0 AND tv<37 THEN  LET
 nm=tv: GO TO 1102
1020 PRINT AT 15,0;"Entrada fora
 dos limites": GO TO 1004
1100 REM Entrada de dias, indice
          de precos e dias uteis
1102 DIM s(nm): DIM p(nm): DIM w
(nm): DIM a(nm)
1104 FOR m=1 TO nm
1106 CLS : PRINT AT 5,1;"Entre d
ados mensais ";m
1108 PRINT AT 7,1;"Vendas      "
;: LET d=5: LET de=1: GO SUB 700
0
1109 IF rf=1 THEN  GO TO 1122
1110 IF rf=2 THEN  GO TO 1120
1111 IF rf=3 THEN  GO TO 1116
1112 LET b$="Entre numero menor
que 100000": GO SUB 9002
1114 GO TO 1108
1116 GO SUB 9500
```



```
1118 GO TO 1106
1120 PRINT AT 15,0;"Entrada fora
 dos limites- Repita": GO TO 110
8
1122 PRINT AT 15,1;"
                    "
1124 LET s(m)=tv: LET a(m)=tv
1126 PRINT AT 8,1;"Indice
 ";AT 8,12;: LET d=3: LET de=0:
GO SUB 7000
1127 IF rf=1 THEN  GO TO 1138
1128 IF rf=2 THEN  GO TO 1142
1129 IF rf=3 THEN  GO TO 1134
1130 LET b$="Entre um numero ent
re 100 e 999": GO SUB 9002
1132 GO TO 1126
1134 GO SUB 9502
1136 GO TO 1106
1138 PRINT AT 15,1;"
                          "
1140 IF tv>99 AND tv<1000 THEN
LET p(m)=tv: GO TO 1144
1142 PRINT AT 15,0;"Entrada fora
 dos limites- Repita": GO TO 112
6
1144 PRINT AT 9,1;"D. Uteis
   ";AT 9,12;: LET d=2: GO SUB 7
000
1145 IF rf=1 THEN  GO TO 1156
1146 IF rf=2 THEN  GO TO 1150
1147 IF rf=3 THEN  GO TO 1152
1148 LET b$="Entre um numero ent
re 10 e 31": GO SUB 9002
1150 GO TO 1144
1152 GO SUB 9502
1154 GO TO 1106
1156 PRINT AT 15,1;"
                    "
1158 IF tv>9 AND tv<32 THEN  LET
```

```
 w(m)=tv: GO TO 1162
1160 PRINT AT 15,0;"Entrada fora
 dos limites - Repita": GO TO 114
4
1162 NEXT m
1200 REM Escolha do tipo de
           ajustamento
1202 CLS : PRINT AT 5,1;"Entre o
 tipo de ajustamento"
1204 PRINT AT 7,3;"1) Inflaccao"
1206 PRINT AT 9,3;"2) Dias Uteis
"
1208 PRINT AT 11,3;"3) Inflaccao
 e D. Uteis"
1210 PRINT AT 13,1;"Entre a sua
opcao=  ";: LET d=1: GO SUB 7000
1211 IF rf=1 THEN  GO TO 1222
1212 IF rf=2 THEN  GO TO 1224
1213 IF rf=3 THEN  GO TO 1218
1214 LET b$="Entre um numero ent
re 1 e 3": GO SUB 9002
1216 GO TO 1202
1218 GO SUB 9502
1220 GO TO 1202
1222 IF tv>0 AND tv<4 THEN  LET
ac=tv: GO TO 1226
1224 PRINT AT 15,0;"Entrada fora
 dos limites- Repita": GO TO 120
2
1226 REM Ajustamentos
1227 IF ac=1 OR ac=3 THEN  GO SU
B 5000
1228 IF ac=2 THEN  GO SUB 6000
1230 IF ac<>3 THEN  GO TO 1302
1232 GO SUB 6000
1300 REM Saida para Impressao
1302 LET l=1: LET m=10: IF m>nm
THEN  LET m=nm
1304 CLS : PRINT "Vendas ajustad
```



```
as para  "
1305 IF ac=1 THEN  GO TO 1308
1306 IF ac=2 THEN  GO TO 1310
1307 IF ac=3 THEN  GO TO 1312
1308 PRINT "Inflaccao": GO TO 13
14
1310 PRINT "Dias Uteis": GO TO 1
314
1312 PRINT "Inflaccao e Dias Ute
is"
1314 PRINT "=====================
=="
1316 PRINT "Vendas  Ven.Ajustada
s  INDICE"
1318 PRINT : FOR n=1 TO m
1320 PRINT s(n);TAB (12);a(n);TA
B (24);
1321 IF ac=1 THEN  GO TO 1324
1322 IF ac=2 THEN  GO TO 1326
1323 IF ac=3 THEN  GO TO 1328
1324 PRINT p(n): GO TO 1330
1326 PRINT w(n): GO TO 1330
1328 PRINT p(n);" / ";w(n)
1330 NEXT n
1332 IF m=nm THEN  GO TO 1348
1334 PRINT : PRINT : PRINT "Prim
a SPACE para continuar": GO SUB
8000
1346 LET l=l+10: LET m=m+10: IF
m>nm THEN  LET m=nm
1347 GO TO 1304
1348 PRINT AT 17,1;"Voce quer:
 1) Rever a Tabela
 2) Terminar
 entre 1 ou 2 --"
1350 LET d=1: GO SUB 7000
1351 IF rf=1 THEN  GO TO 1362
1352 IF rf=2 THEN  GO TO 1366
1353 IF rf=3 THEN  GO TO 1358
```

```
1354 LET b$="Entre 1 para nova l
ietagem": GO SUB 9002
1356 GO TO 1348
1358 GO SUB 9502
1360 GO TO 1348
1362 PRINT AT 15,0;"
                    "
1364 IF tv>0 AND tv<3 THEN  GO T
O 1367
1366 PRINT AT 15,0;"Entrada fora
 dos limites- Repita": GO TO 134
8
1367 IF tv=1 THEN  GO TO 1302
1368 IF tv=2 THEN  GO TO 9508
1370 STOP
5000 REM Ajustamento da Inflacca
o
5002 FOR m=1 TO nm
5004 LET a(m)=a(m)*(p(1)/p(m))
5005 LET a(m)=INT (a(m))
5006 NEXT m
5008 RETURN
6000 REM Ajustamento dos Dias
           Uteis
6002 LET td=0
6004 FOR m=1 TO nm
6006 LET td=td+w(m)
6008 NEXT m
6010 FOR m=1 TO nm
6012 LET ad=td/nm
6014 LET a(m)=a(m)*(ad/w(m))
6015 LET a(m)=INT (a(m))
6016 NEXT m
6018 RETURN
7000 REM Entrada de Numeros
7002 LET tv=0: LET t$="": LET ch
=0
7004 LET a$=INKEY$: IF a$<>"" TH
EN  GO TO 7004
```



```
7005 LET a$=INKEY$: IF a$="" THE
N  GO TO 7005
7006 IF CODE (a$)=13 AND ch=0 TH
EN  LET rf=0: GO TO 7038
7008 IF de=1 AND a$="." THEN  GO
 TO 7022
7010 IF CODE (a$)=12 THEN  GO TO
 7026
7012 IF CODE (a$)=13 THEN  LET r
f=1: GO TO 7036
7014 IF a$="@" THEN  LET rf=3: G
O TO 7038
7016 IF ch=d THEN  LET rf=2: GO
TO 7038
7018 IF a$>="0" AND a$<="9" THEN
  GO TO 7022
7020 GO TO 7004
7022 LET t$=t$+a$: PRINT a$;: LE
T ch=ch+1
7024 GO TO 7004
7026 IF ch=0 THEN  GO TO 7004
7028 LET ch=ch-1: PRINT CHR$ (8)
;" ";CHR$ (8);
7030 IF ch=0 THEN  LET t$="": GO
 TO 7004
7032 LET t$=t$(1 TO LEN (t$)-1)
7034 GO TO 7004
7036 LET tv=VAL (t$)
7038 LET d=0: RETURN
8000 REM Entrada por SPACE
8002 LET tv=0: LET ch=0
8004 LET a$=INKEY$: IF a$<>"" TH
EN  GO TO 8004
8005 LET a$=INKEY$: IF a$="" THE
N  GO TO 8005
8014 IF a$="@" THEN  LET rf=3: G
O TO 8036
8034 LET rf=1
8036 RETURN
```

```
8500 REM Resposta S ou N
8502 LET tv=0: LET t$="": LET ch
=0
8504 LET a$=INKEY$: IF a$<>"" TH
EN  GO TO 8504
8505 LET a$=INKEY$: IF a$="" THE
N  GO TO 8505
8506 IF CODE (a$)=13 AND ch=0 TH
EN  LET rf=0: GO TO 8536
8510 IF CODE (a$)=12 THEN  GO TO
 8526
8512 IF CODE (a$)=13 THEN  LET r
f=1: GO TO 8536
8514 IF a$="@" THEN  LET rf=3: G
O TO 8538
8516 IF a$="s" AND ch=0 THEN  LE
T t$="1": LET rf=1: GO TO 8522
8518 IF a$="n" AND ch=0 THEN  LE
T t$="2": LET rf=1: GO TO 8522
8520 GO TO 8504
8522 PRINT a$;: LET ch=ch+1
8524 GO TO 8504
8526 IF ch=0 THEN  GO TO 8504
8528 LET ch=ch-1: PRINT CHR$ (8)
;" ";CHR$ (8);
8530 IF ch=0 THEN  LET t$="": GO
 TO 8504
8532 LET t$=t$(1 TO LEN (t$)-1)
8534 GO TO 8504
8536 LET tv=VAL (t$)
8538 RETURN
9000 REM Rotina de Auxilio
9002 PRINT AT 14,1;"
                                "
9004 PRINT AT 14,1;b$: PRINT AT
16,4;"Prima SPACE para continuar
": GO SUB 8000
9014 PRINT AT 14,1;"
```



```
                                "
9016 RETURN
9500 REM Rotina de Abandono
9502 CLS : PRINT AT 15,0;"Quer m
esmo terminar?": GO SUB 8500
9503 IF rf=1 THEN  GO TO 9506
9504 IF rf=2 OR rf=3 THEN  GO TO
 9502
9506 IF tv=2 THEN  GO TO 9510
9508 CLS : PRINT AT 10,5;"PROCES
SAMENTO TERMINADO": STOP
9510 CLS : RETURN
```

CONCLUSÃO

Depois de escrito (e gravado!) o programa, observem-se os valores seguintes, relativos a 12 meses. Como correu o ano? Os negócios estão a melhorar ou a piorar? Tratar-se-á de um fenómeno sazonal? Será que o ímpeto do primeiro quadrimestre se perdeu no resto do ano?

Mês	Vendas	Índice de preços	Dias úteis
Jan	83	100	16
Fev	94	100	18
Mar	109	100	21
Abr	106	102	20
Mai	95	102	18
Jun	106	102	20
Jul	114	104	21
Ago	97	104	18
Set	114	104	21
Out	115	106	21
Nov	110	106	20
Dez	94	106	17

Introduzamos agora estes dados no programa **AJUSTADOR.**

O programa melhoraria extraordinariamente com a inclusão da possibilidade de armazenamento em fita ou disco, escrita no final da listagem. Refinamento ainda maior seria a hipótese de se acrescentarem dados de meses seguintes, por forma a manter uma análise de tendências contínua, sem a obrigação de termos de introduzir todos os dados de cada vez que precisamos do programa. Os métodos para se conseguirem estas facilidades tornar-se-ão compreensíveis nos próximos capítulos.

Capítulo 5

Gráficos
Traçado de gráficos e mapas

INTRODUÇÃO

A apresentação cuidada em *écran* tornou-se hoje em dia numa das características mais apreciadas da computação pessoal. Na verdade, a qualidade de apresentação da imagem dos modernos computadores pessoais pode mesmo ser superior à conseguida em alguns programas comerciais de aplicação profissional.

Os gráficos assumem importância especial nos programas escritos para computadores domésticos, devido à natureza interactiva das tarefas que executam e à predominância da apresentação da imagem em *écran*, em detrimento da impressão em papel.

Pelo menos em primeira análise, a programação de gráficos é assunto de fácil abordagem — basta juntar pontos ao *écran* de modo a descobrir traços. A prática, porém, aponta para realidade bem diferente: é necessário sabermos um pouco da forma como o computador traça gráficos, e das particularidades de cada máquina, antes de nos metermos na programação gráfica.

O *écran* pode ser considerado como uma vulgar folha de papel milimétrico, em que cada pequeno quadrado — chamado **pixel** — é formado pela intersecção de linhas verticais e horizontais. O *Spectrum* segue a convenção normal de numeração de coordenadas gráficas, ou seja, atribui ao canto inferior esquerdo da imagem o valor **0,0**. Um ponto designado por **10,40** (a coordenada X seguida pela coordenada Y) está localizado **10** *pixels* à direita da margem esquerda do *écran* e 40 *pixels* acima do fundo do mesmo.

Uma das "facilidades" da programação gráfica é que só temos de lidar com duas instruções: **PLOT**, que posiciona um ponto a partir do qual se inicia uma linha, e **DRAW**, que a traça

para outro ponto definido por coordenadas, onde termina. As coordenadas usadas na instrução **DRAW** são relativas, isto é, são medidas a partir do ponto definido pelo **PLOT** anterior. Deste modo, a instrução **PLOT 10,40**, dá início a uma linha com origem nas coordenadas **10,40**. A instrução seguinte, por exemplo **DRAW 5,10**, traça uma linha para um ponto 5 *pixels* à direita e 10 *pixels* acima do ponto inicial. As coordenadas "verdadeiras" do fim da linha são **15,50**.

DESCRIÇÃO DO PROGRAMA

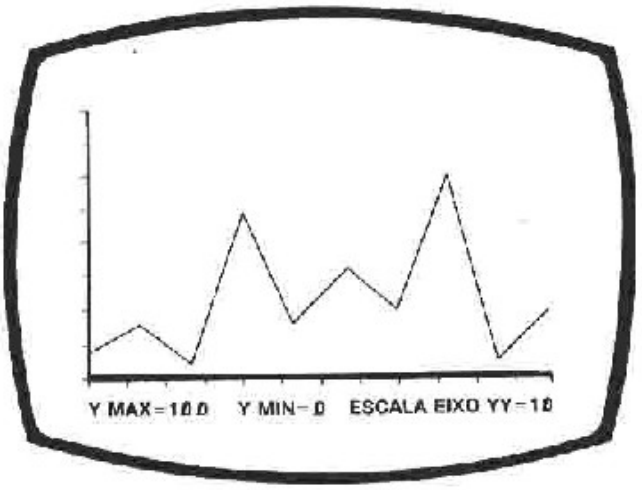
Para compreendermos as possibilidades do programa, imaginemos um gráfico de vendas relativo a 12 meses. Uma linha horizontal (o eixo dos XX) dividida em doze secções igualmente espaçadas, cujo comprimento representa um mês. O eixo vertical (dos YY) está escalado de **0** ao valor máximo das vendas mensais — ou porventura do valor mínimo de vendas mensais ao valor máximo —, e cada divisão representa uma quantidade de artigos vendidos. As vendas de determinado mês são traçadas pela intersecção dos prolongamentos do ponto referente ao mês, no eixo dos XX, e do ponto que assinala o montante de vendas, no eixo vertical. Esse cruzamento de linhas determina um ponto coordenado — isto é (1,20), significa vendas de 20 unidades no mês 1; (12,60) indica a venda de 60 unidades no mês 12.

O programa começa pela entrada de dados, a partir dos quais se irá traçar o gráfico. Como temos muitos números a introduzir, servimo-nos aqui de sofisticada rotina de entrada, cuja principal característica consiste na correcção de erros sem se ter de voltar ao início.

As linhas **1000** a **1026** permitem a entrada do desejado número de pontos coordenados horizontais (representando o número de meses), da forma que já ficou descrita em capítulos anteriores.

A rotina principal de entrada de dados foi concebida de modo a que os valores sejam introduzidos a partir do fundo do *écran*, subindo em *scroll* à medida que o novos valores são digi-

GRÁFICO



tados; deste modo, cada nova entrada fica "por baixo" da anterior. Se se cometer algum erro, o utilizador pode corrigi-lo na ocasião, simplesmente premindo a tecla **E**. A rotina começa na linha **1032**, em que o círculo em **J** é estabelecido para contar de **1** a **NM** (o número de entradas). Na primeira destas (**J=1**), o programa salta para a linha **1042**, sendo o valor atribuído a **D( )** na linha **1066**. Os dados são imprimidos na linha **20** do *écran* (veja-se a linha de programa **1044**). A entrada dos valores é controlada pelas rotinas já familiares descritas no capítulo 3, com a excepção de que, neste programa, **RF** pode assumir um de quatro valores: **RF=4** torna-se na tecla **E**, de correcção de erros. A bandeira **FY** é zero; quando, nesta fase, se torna igual a 1, tal significa que o programa chamou a rotina de verificação. Partindo do princípio de que a entrada é válida, o círculo **J=1** até **NM** avança em indexação na linha **1068**, e o programa recomeça na linha **1032**.

Imagine-se agora que D( ) contém certo número de entradas, e que J já não tem o valor 1, entrando em jogo as linhas 1030 e 1040. Estas linhas executam a listagem dos valores introduzidos que já se encontram armazenados no grupo D( ), ordenada de modo a que a última entrada surja na linha 19 do *écran*, isto é, uma linha acima da reservada à apresentação das entradas. O resultado deste procedimento é uma "coluna" de números em que o primeiro valor aparece no topo, e o último imediatamente por cima da linha de entrada de dados. Cada novo valor enviado do teclado aparece na linha 20, ocupando o lugar que lhe compete no fundo da coluna. À medida que vamos digitando novos valores, a coluna vai «subindo», até alcançar o cimo do *écran*. A linha 1036, em conjunto com a 1038 assegura que, à medida que o processo se desenrola, só a parte da "coluna" que cabe no *écran* é colocada em posição correcta. Portanto, quando J=20, N passa de 1 a 19 e na imagem só aparecem linhas de dados. Quando J>20, N passa a ter o valor zero, mas os valores zero são rejeitados na linha 1038, o que provoca, depois de J exceder 20, nova volta do círculo de 1 a 19, de modo que as entradas apresentadas nunca excedem a capacidade do *écran*.

A rotina de correcção reside na sub-rotina 6000 e é chamada pela tecla E via linha 1072. A finalidade deste bloco consiste em estabelecer o que poderemos designar por "rotina auxiliar de entrada de dados": estabelece-se JJ igual a J, de modo que o programa se "lembra" de quantos dados foram introduzidos até ao momento em que se prime a tecla E. As linhas 6000 a 6008 posicionam o cursor ao lado do topo da "coluna" de números, mas as saídas de RF da sub-rotina 7000 têm aqui uma utilização diferente da habitual: RF=0 (o próprio RETURN) assinala que a entrada em presença é correcta, não precisando portanto de correcção. Se RF for zero, o cursor é movido para uma posição imediatamente abaixo, reduzindo-se o contador JJ em uma unidade, na linha 6012; repare que quando JJ assume o valor 1, a rotina de correcção tem de terminar, pois o cursor desceu até à linha de entrada de dados. Se fizermos a

emenda de um valor entrado, o facto reflecte-se sob a forma de entrada RF=1, que estabelece o valor do elemento apropriado do grupo D( ) para o valor corrigido de TV, na linha 6018. A linha 6012 reposiciona o cursor.

Com a entrada de dados completa, passamos à fase seguinte, a qual oferece a oportunidade de verificar tudo o que se introduziu. Isto consegue-se percorrendo novamente toda a rotina de entrada, mas agora a bandeira FY é estabelecida para o valor 1, na linha 2014, e em resposta à pergunta "Quer corrigir as entradas?" feita pela linha 1080.

A rotina de verificação tem a mesma apresentação em *écran* que a entrada de dados; contudo, em lugar de esperar na linha de entrada por novo valor, o programa imprime logo o número digitado. Esta operação é controlada pela bandeira FY, que assume o valor 1. FY torna-se efectiva, em primeiro lugar, na linha 1046 onde provoca a entrada da sub-rotina 7000, não no início desta, mas na linha 7004. Os valores de CH e de T$ derivam dos dados armazenados em D( ), e são injectados na sub-rotina, com o propósito (conseguido) de a "enganar", levando-a a pensar que a entrada de dados já está a decorrer. Como resultado, a sub-rotina imprime o valor de T$ e coloca o cursor na posição imediata, à espera do valor seguinte da entrada A$ (linha 7004). Se não forem necessárias correcções, o operador prime imediatamente RETURN e sai da sub-rotina 7000, com RF em 1 e TV assumindo o valor dado na linha 146 — isto é, o valor original de D( ). Se for necessária qualquer correcção, a rotina 7000 comporta-se do mesmo modo que durante a normal entrada de dados. O operador recua um espaço, escreve o novo número, prime RETURN e sai do bloco 7000 com RF em 1 e com a entrada revista contida em TV. Esta é então transferida para D( ) na linha 1066. Repare-se na possibilidade (remota) de suceder uma saída RF=0, que é processada como simples entrada RETURN pela linha 1053.

Concluímos deste modo todo o processo de entrada de dados, e vamos passar a analisar o traçado do gráfico. Por estranho que pareça, o procedimento seguido é muito semelhante

ao que seguiríamos se o desenhássemos à mão. Os valores máximo e mínimo de Y são calculados pela primeira fase de um processo de "extracção de bolas", nas linhas 2520 a 2550, e depois imprimidos para que o operador escolha o modo como quer o gráfico apresentado. Não são permitidos valores do eixo dos YY que se situem entre o máximo e o mínimo do valor dos dados, pois de outro modo poderíamos correr o risco de ter o gráfico a "sair do *écran*" (linhas 2584 e 2604).

Passamos assim para a linha 3000, na qual se inicia o cálculo das posições de *écran* das linhas que representam o eixo dos YY. A finalidade é conseguir que a escala deste eixo vá de Y máximo a Y mínimo com um máximo de dez subdivisões ao longo de toda a linha; por outro lado, o intervalo entre essas divisões terá de ser um número inteiro. Torna-se aparente que, para conseguirmos tudo isto, algo terá de ser limitado; esse "algo" é o valor máximo do eixo (isto é, Y max.). Descrevem-se a seguir as várias fases dos cálculos: parecem muito mais complicados do que na realidade são, mas seguem quase os mesmos processos que usaríamos se desenhássemos o gráfico à mão.

1. Calcular (Y max. — Y min.) tal como especificado pelo operador — linha 3002.

2. Se (Y max. — Y min.) for superior a 10 unidades, dividir a escala por 5 (linha 3006). Se o resultado não for um número inteiro, regressar a trás e dividir o comprimento por 2. Se mesmo assim não surgir um número inteiro, somar 1 a Y max. e repetir o processo (linha 3010). Por exemplo, suponha que o operador determina para Y max. o valor 12 e para Y min. 1. (Y max. — Y min.) é 11, valor muito grande. Primeiro, 11 é dividido por 5, o que não dá um número inteiro; a seguir, efectua-se a divisão por 2, mas 11/2 também não dá o resultado pretendido. Como tal, o computador soma 1 a Y max., que passa a 12, e o processo repete-se. Desta vez, 12 é divisível por 2, e como o resultado, 6, é menor que 11 (linha 3004) passa a constituir o valor de ND — o número de divisões da escala.

3. A posição de Y min. (que é a origem do gráfico) é fixada num ponto 40 *pixels* acima do fundo do *écran*. Ficamos portanto com 200 *pixels* disponíveis no plano vertical para o resto do gráfico. A linha 3014 calcula o número de *pixels* por divisão, que tem sempre de ser um valor inteiro; a linha 3018 calcula a escala do eixo dos YY em termos de número de *pixels* por valor de Y, fixando assim a posição do outro extremo deste eixo.

4. O eixo dos XX é menos complicado. A sua origem é posicionada 10 *pixels* à direita da margem esquerda do *écran*, deixando livres 100 *pixels*, no plano horizontal, para o resto do gráfico. O número de *pixels* por cada divisão do eixo dos XX é calculado pela linha 3016.

Depois de estabelecidas as dimensões do gráfico, no que respeita a *pixels*, podemos traçar as linhas no *écran*. A linha 4006 desenha um traço vertical ao longo da margem esquerda da imagem, para representar o eixo dos YY, e cujo comprimento é dado por YI×ND, ou seja, o número de subdivisões multiplicado pelo número de *pixels* por divisão. As linhas 4008 a 4012 desenham pequenos traços horizontais para assinalar as divisões do eixo dos YY. Existem ND+1 destes traços, com um comprimento de 5 *pixels* cada. O eixo dos XX é traçado de forma semelhante, pelas linhas 4014 a 4022.

A parte final do programa trata do posicionamento dos pontos de dados, executado pelo *loop* com início na linha 5002. A linha 5004 adquire os dados aos pares e desenha um traço entre cada par. A coordenada X é definida pela expressão 10+N×XI, em que XI é o número de *pixels* em cada divisão do eixo dos XX. A coordenada Y é calculada pela expressão (D(N+1)×SC+40), em que D(N+1) é o valor dos dados de pontos e SC representa a escala em número de *pixels* por unidade.

A linha 5008 imprime as dimensões da escala do gráfico na "janela de texto" colocada no fundo do *écran*, e a linha 5020 avisa o operador de que pode abandonar o programa ou reiniciar o traçado. Por fim, a linha 5032 executa a opção tomada.

```
1000 REM GRAFICOS
1002 LET fy=0
1004 CLS : PRINT " GRAFICOS": PR
INT "==========="
1006 PRINT AT 2,4;"Entre o numer
o de pontos          (inteiro ent
re 2 e 36)"
1008 PRINT AT 4,24;: LET d=6: GO
 SUB 7000
1009 IF rf=1 OR rf=2 THEN  GO TO
 1020
1010 IF rf=3 THEN  GO TO 1016
1012 LET b$="numero inteiro entr
e 2 e 36": GO SUB 9000
1014 GO TO 1006
1016 GO SUB 9500
1018 GO TO 1006
1020 IF tv<36 AND tv>2 THEN  GO
TO 1024
1022 PRINT AT 15,0;"Numero intei
ro entre 2 e 36?": GO TO 1008
1024 PRINT AT 15,0;"
                    "
1026 LET nm=tv: DIM d(nm)
1028 CLS : PRINT "Entrada de Dad
os"
1030 PRINT "Prima E para corrigi
r"
1032 FOR j=1 TO nm
1034 IF j=1 THEN  GO TO 1042
1036 FOR n=21-j TO 19
1038 IF n>0 THEN  PRINT AT n,16;
```



```
"          ": PRINT AT n,16;d(j-(2
0-n))
1040 NEXT n
1042 PRINT AT 20,1;
1044 PRINT "Entre no. ";j;"...."
: PRINT AT 20,16;"       ";AT 20,
16;
1046 IF fy>0 THEN  LET t$=STR$ (
d(j)): LET ch=LEN (t$): PRINT t$
;: LET d=6: GO SUB 7004: GO TO 1
050
1048 LET d=6: GO SUB 7000
1049 IF rf=1 THEN  GO TO 1064
1050 IF rf=2 THEN  GO TO 1062
1051 IF rf=3 THEN  GO TO 1058
1052 IF rf=4 THEN  GO TO 1072
1053 IF fy=1 THEN  GO TO 1064
1054 LET b$="Entre valor da coor
denada Y": GO SUB 9000
1056 GO TO 1042
1058 GO SUB 9500
1060 GO TO 1042
1062 PRINT AT 15,0;"Numero com m
ais de 6 digitos": GO TO 1042
1064 PRINT AT 15,0;"
                    "
1066 LET d(j)=tv
1068 NEXT j
1070 GO TO 1076
1072 GO SUB 6000
1074 GO TO 1034
1076 IF fy=1 THEN  LET fy=0
1078 CLS
1080 PRINT AT 1,0;"Quer corrigir
 as entradas?  ";
1082 GO SUB 8500
1083 IF rf=1 OR rf=2 THEN  GO TO
 1094
```

```
1084 IF rf=3 THEN  GO TO 1090
1086 LET b$="Entre S ou N": GO S
UB 9000
1088 GO TO 1076
1090 GO SUB 9500
1092 GO TO 1076
1094 IF tv=1 THEN  GO TO 2000
1096 IF tv=2 THEN  GO TO 2500
2000 REM *Rotina de Verificacao*
2002 CLS
2004 PRINT "Verificacao da Entra
da": PRINT "=====================
=="
2006 PRINT "Se OK - ENTER "
2008 PRINT "Se errada - Reentre"
2010 PRINT "Prima E para corrigi
r"
2014 LET fy=1: GO TO 1032
2500 REM *Calculo de ym e de yn*
2510 LET ym=0: LET yn=d(1)
2520 FOR n=1 TO nm
2530 IF d(n)>ym THEN  LET ym=d(n
)
2540 IF d(n)<yn THEN  LET yn=d(n
)
2550 NEXT n
2558 CLS
2560 PRINT AT 1,1;"Entre os valo
res Max. e min.     para o eixo d
os YY"
2562 PRINT AT 3,1;"O Maximo e'
";ym
2564 PRINT AT 5,1;"O minimo e'
";yn
2566 PRINT AT 7,1;"Entre Y_Max.
     ";
2568 LET d=8: GO SUB 7000
2569 IF rf=1 THEN  GO TO 2582
2570 IF rf=2 THEN  GO TO 2580
```



```
2571 IF rf=3 THEN  GO TO 2576
2572 LET b$="Para estabelecer o
Max. e o min. do eixo YY": GO SU
B 9000
2574 GO TO 2566
2576 GO SUB 9500
2578 GO TO 2566
2580 PRINT AT 15,0;"Numero maior
 que 8 digitos";: GO TO 2566
2582 PRINT AT 15,0;"
                   "
2584 LET am=tv: IF am<ym THEN  P
RINT AT 15,0;"Y max < maior entr
ada- Repita": GO TO 2560
2586 PRINT AT 9,1;"Entre Y_min.
     ";
2588 LET d=8: GO SUB 7000
2589 IF rf=1 THEN  GO TO 2602
2590 IF rf=2 THEN  GO TO 2600
2591 IF rf=3 THEN  GO TO 2596
2592 LET b$="Para estabelecer o
Max. e o min. do eixo YY": GO SU
B 9000
2594 GO TO 2586
2596 GO SUB 9500
2598 GO TO 2586
2600 PRINT AT 15,0;"Numero maior
 que 8 digitos";: GO TO 2586
2602 PRINT AT 15,0;"
                   "
2604 LET an=tv: IF an>yn THEN  P
RINT AT 15,0;"Y min > menor entr
ada- Repita": GO TO 2586
3000 REM Calculo do numero de
         divisoes do eixo YY
3002 LET nd=am-an
3004 IF nd<11 THEN  GO TO 3014
3006 LET nd=nd/5: IF (nd-INT (nd
))<0.001 THEN  GO TO 3004
```

```
3008 LET nd=nd*5/2: IF (nd-INT (
nd))<0.001 THEN  GO TO 3004
3010 LET am=am+1
3012 GO TO 3002
3014 LET yi=INT (100/nd)
3016 LET xi=INT (200/(nm-1))
3018 LET sc=(yi*nd)/(am-an)
4000 REM **Desenho do eixo YY**
4002 CLS
4004 PLOT 10,40
4006 DRAW 0,yi*nd
4008 FOR n=0 TO nd
4010 PLOT 5,yi*n+40
4012 NEXT n
4014 REM **** Eixo XX ****
4016 PLOT 10,40
4017 DRAW xi*(nm-1),0
4018 FOR n=0 TO nm-1
4020 PLOT 10+n*xi,40
4021 DRAW 0,-5
4022 NEXT n
5000 REM ***Tracado de Pontos***
5002 FOR n=0 TO nm-2
5004 PLOT 10+n*xi,(d(n+1))*sc+40
: DRAW xi,(d(n+2)-d(n+1))*sc
5006 NEXT n
5008 PRINT AT 20,1;"Y_max= ";am:
 PRINT AT 21,1;"Y_min= ";an;"  E
scala Eixo YY= ";(am-an)/nd
5020 PRINT AT 1,0;"Entre 1) para
 reiniciar                2) para
 terminar  ";: LET d=1: GO SUB 7
000
5021 IF rf=1 THEN  GO TO 5032
5022 IF rf=2 THEN  GO TO 5024
5023 IF rf=3 THEN  GO TO 5028
5026 GO TO 5020
5028 GO SUB 9500
5030 GO TO 5020
```



```
5032 IF tv=1 THEN  GO TO 1000
5033 IF tv=2 THEN  GO TO 5028
6000 REM **Rotina de Correccao**
6002 LET jj=j: IF jj>18 THEN  LE
T jj=18
6006 PRINT AT 20-jj,21;"Correcca
o"
6007 REM     '*' e um cursor
              artificial
6008 PRINT AT 21-jj,21;"*";CHR$
(8);: LET d=6: GO SUB 7000
6009 IF rf=1 THEN  GO TO 6015
6010 IF rf=2 THEN  GO TO 6022
6011 IF rf=3 THEN  GO TO 8300
6012 PRINT AT 21-jj,21;" ";
6013 LET jj=jj-1: IF jj<=1 THEN
 GO TO 6024
6014 GO TO 6008
6015 INVERSE 0
6018 LET d(j-jj+1)=tv: GO TO 601
3
6020 INVERSE 1
6022 PRINT AT 15,0;"Se errado- R
eentre": GO TO 6008
6024 RETURN
7000 REM **Entradas Numericas**
7002 LET tv=0: LET t$=" ": LET c
h=0
7004 LET a$=INKEY$: IF a$<>"" TH
EN  GO TO 7004
7005 LET a$=INKEY$: IF a$="" THE
N  GO TO 7005
7006 IF CODE (a$)=13 AND ch=0 TH
EN  LET rf=0: GO TO 7040
7010 IF CODE (a$)=12 THEN  GO TO
 7025
7012 IF CODE (a$)=13 THEN  LET r
f=1: GO TO 7038
7014 IF a$="@" THEN  LET rf=3: G
```

```
O TO 7040
7016 IF ch=d THEN  LET rf=2: GO
TO 7040
7018 IF a$="e" THEN  LET rf=4: G
O TO 7040
7020 IF a$>="0" AND a$<="9" THEN
  GO TO 7024
7022 GO TO 7004
7024 LET t$=t$+a$: PRINT a$;: LE
T ch=ch+1
7026 GO TO 7004
7028 IF ch=0 THEN  GO TO 7004
7030 LET ch=ch-1: PRINT CHR$ (8)
;" ";CHR$ (8);
7032 IF ch=0 THEN  LET t$="": GO
 TO 7004
7034 LET t$=t$(1 TO LEN (t$)-1)
7036 GO TO 7004
7038 LET tv=VAL (t$)
7040 LET d=0: RETURN
8000 REM ***Entrada por SPACE***
8002 LET tv=0: LET t$=" ": LET c
h=0
8003 LET a$=INKEY$: IF a$<>"" TH
EN  GO TO 8003
8004 LET a$=INKEY$: IF a$="" THE
N  GO TO 8004
8006 IF CODE (a$)=13 AND ch=0 TH
EN  LET rf=0: GO TO 8036
8010 IF CODE (a$)=12 THEN  GO TO
 8024
8012 IF CODE (a$)=13 THEN  LET r
f=1: GO TO 8034
8014 IF a$="@" THEN  LET rf=3: G
O TO 8036
8016 IF a$=" " AND ch=0 THEN  LE
T rf=1: GO TO 8036
8018 GO TO 8004
8020 LET t$=t$+a$: PRINT a$;: LE
```



```
T ch=ch+1
8022 GO TO 8004
8024 IF ch=0 THEN  GO TO 8004
8026 LET ch=ch-1: PRINT CHR$ (8)
;" ";CHR$ (8);
8028 IF ch=0 THEN  LET t$="": GO
 TO 8004
8030 LET t$=t$(1 TO LEN (t$)-1)
8032 GO TO 8004
8034 LET tv=0
8036 RETURN
8500 REM ****Entrada S ou N****
8502 LET tv=0: LET t$=" ": LET c
h=0
8504 LET a$=INKEY$: IF a$<>"" TH
EN  GO TO 8504
8505 LET a$=INKEY$: IF a$="" THE
N  GO TO 8505
8506 IF CODE (a$)=13 AND ch=0 TH
EN  LET rf=0: GO TO 8538
8510 IF CODE (a$)=12 THEN  GO TO
 8526
8512 IF CODE (a$)=13 THEN  LET r
f=1: GO TO 8536
8514 IF a$="@" THEN  LET rf=3: G
O TO 8538
8516 IF a$="s" AND ch=0 THEN  LE
T t$="1": LET rf=1: GO TO 8522
8518 IF a$="n" AND ch=0 THEN  LE
T t$="2": LET rf=1: GO TO 8522
8520 GO TO 8504
8522 PRINT a$;: LET ch=1
8524 GO TO 8504
8526 IF ch=0 THEN  GO TO 8504
8528 LET ch=ch-1: PRINT CHR$ (8)
;" ";CHR$ (8);
8530 IF ch=0 THEN  LET t$="": GO
 TO 8504
8532 LET t$=t$(1 TO LEN (t$)-1)
```

```
8534 GO TO 8504
8536 LET tv=VAL (t$)
8538 RETURN
9000 REM ***Rotina de Auxilio***
9002 PRINT AT 15,0;"
                    "
9004 PRINT AT 15,0;b$: PRINT "Pr
ima SPACE para continuar": GO SU
B 8000
9005 IF rf=1 THEN  GO TO 9014
9006 IF rf=2 THEN  GO TO 9008
9007 IF rf=3 THEN  GO TO 9010
9008 GO TO 9002
9010 GO SUB 9502
9012 GO TO 9002
9014 PRINT AT 15,0;"
                    "
9016 RETURN
9500 REM ***Rotina de Escape***
9502 CLS : PRINT AT 15,1;"Quer m
esmo terminar?": GO SUB 8500
9503 IF rf=1 THEN  GO TO 9506
9504 IF rf=2 OR rf=3 THEN  GO TO
 9502
9506 IF tv=2 THEN  GO TO 9510
9508 CLS : PRINT AT 10,5;"PROCES
SAMENTO TERMINADO": STOP
9510 CLS : RETURN
```

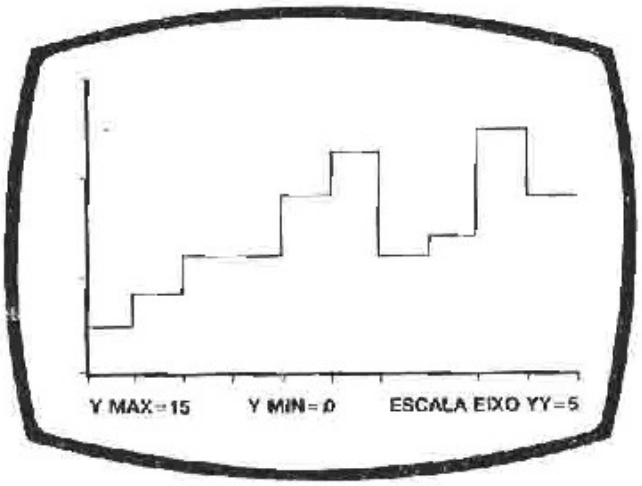
MODIFICAÇÕES

O inconveniente dos gráficos ponto a ponto, especialmente se apresentados numa televisão doméstica, é que as linhas ficam um pouco "irregulares". A razão deste problema reside no facto de que o computador só consegue traçar linhas verticais ou horizontais, enquanto que as diagonais são na verdade formadas por determinado número de pequenos "degraus". Podemos melhorar a apresentação do gráfico usando exclusivamente

linhas verticais e horizontais, mostrando as variações em verdadeiros saltos em lugar de tentarmos desenhar linhas de ponto para ponto. Nas páginas seguintes encontram-se dois exemplos de gráficos mais apresentáveis.

A modificação «Gráficos 1» produz um gráfico em escala. As alterações verificam-se nas linhas 3016, 4017, 4018 e 5004. Precisamos de uma divisão extra no eixo dos XX para o traçado da "escada", o que se consegue alterando NM−1 para NM nas linhas 3016, 4017 e 4018. A linha 5004 desenha em primeiro lugar a parte horizontal do degrau e depois a parte vertical.

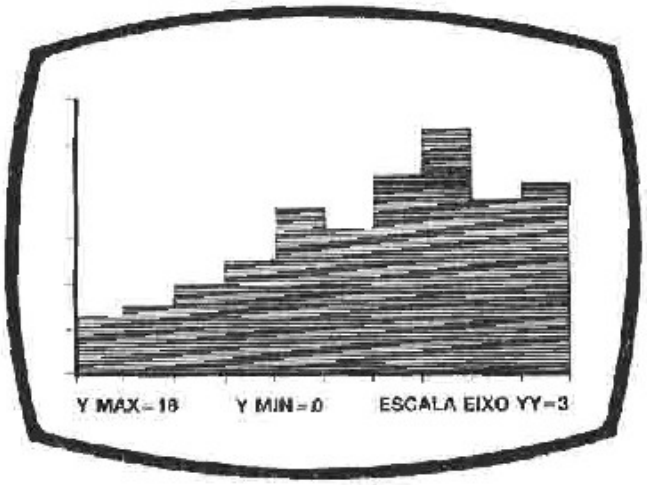
GRÁFICO 1




```
3016 LET xi=INT (200/nm)
4017 DRAW xi*nm,0
4018 FOR n=0 TO nm
5004 PLOT 10+n*xi,d(n+1)*sc+40:
DRAW xi,0: DRAW 0,-sc*(d(n+1)-d(
n+2))
```

Obtido que foi o gráfico em escada, podemos encher a área interior do traçado, para se obter uma apresentação "em barras". Um dos métodos possíveis consiste em desenhar linhas horizontais muito perto umas das outras, paralelas à porção horizontal do gráfico em escada. A modificação "Gráficos 2" ilustra esta técnica. Repare-se na instrução **STEP** 2 da linha 5003, que reduz o grau de sombreado ao desenhar todas as linhas, tornando o *écran* menos brilhante e aumentando a velocidade de traçado do gráfico.

GRÁFICO 2




```
3016 LET xi=INT (200/nm)
4017 DRAW xi*nm, 0
4018 FOR n=0 TO nm
5003 FOR m=10+n*xi TO 9+xi+n*xi
STEP 2
5004 PLOT m,40: DRAW 0,d(n+1)*sc
5005 NEXT m
```


Capítulo 6

Previsor
Previsão de vendas

INTRODUÇÃO

A previsão de vendas é um dos aspectos mais importantes da gestão de vendas. Requer conhecimentos, capacidade de avaliação, perícia e um pouco de sorte, qualidades essenciais para que a realidade venha a coincidir com a previsão. Os computadores só em pequena escala podem contribuir para a tarefa global de preparação de uma previsão ponderada, mas mesmo essa limitada ajuda pode revestir importância considerável, desde que o gestor saiba o que é que o computador está a fazer.

O programa "Previsor" apresenta um modelo exponencial de normalização para estender a variação da tendência até ao momento verificada em direcção ao futuro. O método de previsão requer pelo menos 6 períodos de dados reais (já verificados), a partir dos quais irá calcular a variação de tendência e efectuar uma previsão para seis períodos ainda afastados no tempo.

Incorporámos o programa do capítulo anterior, "Gráficos", para podermos traçar os resultados obtidos. O que aqui se processa é o seguinte: introduzimos os dados conhecidos, efectua-se o cálculo da previsão, e o computador imprime tanto as vendas reais como as projectadas, tudo em forma de gráfico. O grau de confiança da previsão pode ser aumentado se introduzirmos dados de vendas "em bruto" no programa "Ajustador", eliminando deste modo os efeitos da inflação e dos dias de trabalho, antes de iniciarmos o cálculo de previsão.

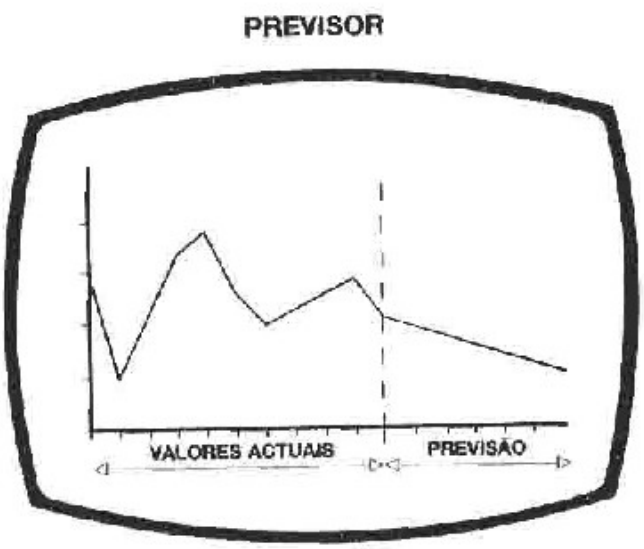
O MODELO DE PREVISÃO

Exigem-se pelo menos seis meses de dados disponíveis neste modelo. Para se efectuar a previsão, o programa avança ao

longo dos valores arquivados, em etapas de período a período. Em cada um destes, o modelo estima quais as vendas futuras e compara-as com as efectivamente praticadas; se a previsão for diferente da realidade, o erro resultante permite o ajustamento da previsão do período seguinte. Este processo de previsão, comparação dos resultados com as vendas efectuadas e ajustamento é repetido pelo menos cinco vezes.

No fim do processo, o modelo trabalhou até ao mais recente período, a partir do qual projecta no futuro um determinado valor de vendas previstas nos seis meses seguintes.

Um exemplo simples ilustra bem as bases deste processo de previsão, ajustamento e nova previsão:



Partamos do princípio de que as vendas previstas para um dado mês, digamos Fevereiro, são de 200 unidades. Suponha-

mos ainda que as vendas reais nesse mês foram de 220 unidades, ou seja, 20 acima do esperado — esta discrepância é o erro a que acima nos referimos. Como se realizaram vendas superiores à expectativa, a previsão anterior tem de ser ajustada para se estimar os valores para Março.

Consegue-se isto somando à primeira estimativa uma fracção do erro, que neste exemplo estabeleceremos em 1/2. Se as vendas fossem inferiores à previsão, o valor do erro seria negativo, o que teria como efeito o abaixamento da nova previsão. Esta será portanto o resultado de $200+1/2\times20=200+10=210$ unidades, e este valor é o que será projectado como vendas para Março.

A tabela seguinte mostra as principais etapas do cálculo!

MÊS	VENDAS REAIS	VENDAS PREVISTAS	ERRO	NOVA PREVISÃO
Fev.	220	200	20	210
Mar.		210		

A segunda tabela mostra a previsão completa baseada em seis meses de registos de vendas, cobrindo o período de Janeiro a Junho. Como será evidente, no início do processo a previsão para o segundo mês (Fevereiro neste caso) tem de ser igual à do primeiro (Janeiro), pois não dispomos de dados anteriores, e não é possível efectuar qualquer estimativa com dados de um só período.

Jan.	200	-	-	-
	220	200	20	210
	206	210	- 2	209
Abr.	191	209	18	218
Mai.	210	218	- 8	214
Jun.	228	214	14	221

Depois de processar todas as vendas reais de Janeiro a Junho, o modelo efectua a previsão para os meses de Julho em

diante, que serão neste caso de 221 unidades por mês. Este método de ajustamento da previsão em proporção do valor do erro é designado por «normalização exponencial simples», e tem a finalidade de normalizar as variações aleatórias e de isolar o padrão base das vendas.

A normalização exponencial dupla leva o processo mais longe, ao incorporar na previsão uma estimativa da variação de tendência. Como o próprio nome sugere, é aplicado um segundo tratamento de normalização exponencial às vendas previstas pela normalização simples. Os valores simples e duplos são depois combinados para se calcular uma previsão final, que irá reflectir a tendência ascendente ou descendente nas vendas.

As previsões obtidas por meio da normalização exponencial dupla incorporam uma estimativa do valor da variação das vendas, de modo que a quantidade prevista para cada mês é diferente das restantes. Pelo contrário, a normalização exponencial simples assume que não há variação de tendência, e portanto o valor estimado é idêntico para todos os períodos futuros.

A decisão importante que temos de tomar quando se nos pede a escolha entre normalização exponencial simples ou dupla é verificar se efectivamente as vendas estiveram sujeitas a qualquer tipo de variação de tendência.

Em ambos os modelos, o número de meses empregados para se construir a previsão depende da quantidade de valores em arquivo. Se dispusermos de dados entre 6 e 11 meses, os modelos servem-se de pelo menos seis meses; se possuirmos registos relativos a doze ou mais períodos, serão utilizados os relativos aos últimos doze meses.

Chama-se «constante de normalização» à fracção de erro que se soma a cada previsão anterior, ao longo das várias fases de cálculo, a qual pode assumir valores compreendidos entre 0 e 1. Se nos servirmos de uma constante de valor elevado, daremos maior ênfase aos dados mais recentes, o que torna o modelo mais sensível às alterações. Se, por qualquer razão, as vendas dispararem num determinado mês, o modelo previsional

comportar-se-á como se se tivesse formado uma verdadeira variação de tendência, e daí para a frente aplicá-la-á nas suas estimativas.

DESCRIÇÃO DO PROGRAMA

O programa abandona a companhia do anterior, "Gráficos", na linha 2018, em que pergunta ao operador se quer alguma previsão. Se a resposta for afirmativa, convida-o então a escolher entre os métodos de normalização exponencial simples ou dupla, e estabelece a bandeira **FO** em **1** ou **2** conforme a opção. A linha 2031 verifica se há suficientes dados em memória — isto é, no mínimo seis meses de registo. Se tal não for o caso, o operador fica a saber que não se poderá efectivar a previsão. O passo seguinte consiste na entrada da constante de normalização **CO**, nas linhas 2048 a 2062. A bandeira **DE** da linha 2048 admite um ponto decimal. **CO** tem de estar compreendida entre **0** e **1**, e se o valor transmitido pelo teclado não estiver dentro destes limites, é imediatamente rejeitado pela linha 2060, regressando o programa à linha 2048 para nova entrada.

Ficamos assim com a introdução de dados completa, pelo que se vai agora efectuar a chamada do modelo previsional, contido na sub-rotina 8750, através da linha 2064. Os princípios de funcionamento do modelo já foram explicados, mas é conveniente debruçarmo-nos sobre a sub-rotina em si.

As linhas 8752 e 8754 estabelecem **K** (o número de períodos usado para a previsão) em **6** ou **12**, consoante a quantidade disponível de dados em arquivo. As linhas 8760 a 8764 estabelecem os valores iniciais de **P( )** (a média exponencial simples), de **R( )** (média exponencial dupla) e de **Q( )** (as previsões exponenciais simples).

As previsões em si são calculadas entre as linhas 8768 e 8792. O alinhamento dos valores previstos com a matriz **D( )** é efectuado da linha 8794 à 8806, em que a linha 8804 impede que a previsão se torne negativa, pois baseámo-nos no princípio de que as vendas negativas são uma raridade! Por fim, a linha 8808 restabelece o número de períodos de **D( )** em mais 6 do

que o original, para criar espaço à previsão — que como vimos se efectua para os seis meses seguintes.

De regresso ao corpo do programa, poderemos ver que a rotina de traçado no gráfico, da linha 2500 à 5009, é semelhante à de "Gráficos", com a diferença de que desta vez o traçado ocupa todo o *écran*, imprimindo o texto separadamente. O operador prime a barra de espaços (SPACE) quando acaba de ver o gráfico (linha 5007), o *écran* fica limpo (linha 5008) e os pormenores da escala são imprimidos pela linha 5009.

O programa termina oferecendo ao operador três opções (linha 5020):

— Nova apresentação do gráfico.
— Regresso ao início do programa para outra entrada de dados.
— Abandono do programa (FIM).

```
1000 REM *****PREVISOR*****
1002 LET fy=0: LET f1=1: LET de=
1
1004 CLS : PRINT "PREVISOR": PRI
NT "========="
1006 PRINT AT 4,1;"Entre numero
de pontos(inteiro  entre 2 e 36)
"
1008 PRINT AT 8,25;"      ";AT 8
,25;: LET d=6: GO SUB 7000
1010 IF rf=1 OR rf=2 THEN  GO TO
 1020
1011 IF rf=3 THEN  GO TO 1016
1012 LET b$="Numero inteiro entr
e 2 e 36": GO SUB 9000
1014 GO TO 1006
1016 GO SUB 9500
1018 GO TO 1006
```



```
1020 IF tv<36 AND tv>2 AND tv=IN
T (tv) THEN  GO TO 1024
1022 PRINT AT 15,1;"Numero intei
ro entre 2 e 36": GO TO 1008
1024 PRINT AT 15,1;"
                    "
1026 LET nm=tv: DIM d(nm+6)
1028 CLS : PRINT "Entrada de Dad
os": PRINT "================="
1030 PRINT "(Prima E para Corrig
ir)"
1032 FOR j=1 TO nm
1034 IF j=1 THEN  GO TO 1042
1036 FOR n=21-j TO 19
1038 IF n>0 THEN  PRINT AT n,0;"
                              ";AT n,
16;d(j-(20-n))
1040 NEXT n
1042 PRINT AT 20,1;
1044 PRINT "Entrada nr. ";j;"...
      ";AT 20,16;
1046 IF fy>0 THEN  LET t$=STR$ (
d(j)): LET ch=LEN (t$): PRINT t$
;: LET d=6: GO SUB 7004: GO TO 1
049
1048 LET d=6: GO SUB 7000
1049 IF rf=1 THEN  GO TO 1064
1050 IF rf=2 THEN  GO TO 1062
1051 IF rf=3 THEN  GO TO 1058
1052 IF rf=4 THEN  GO TO 1072
1053 IF fy=1 THEN  GO TO 1064
1054 LET b$="Entre coordenada Y
do ponto": GO SUB 9000
1056 GO TO 1042
1058 GO SUB 9500
1060 GO TO 1042
1062 PRINT AT 15,1;"Numero maior
 que 6 carac!": GO TO 1042
1064 PRINT AT 15,1;"
```

```
1066 LET d(j)=tv
1068 NEXT j
1070 GO TO 1076
1072 GO SUB 6000
1074 GO TO 1034
1076 IF fy=1 THEN  LET fy=0
1078 CLS
1080 PRINT AT 1,0;"Quer corrigir
 as entradas?";
1082 GO SUB 8500
1083 IF rf=1 OR rf=2 THEN  GO TO
 1094
1085 IF rf=3 THEN  GO TO 1090
1086 LET b$="Se S, corrigir. Se
N, continuar": GO SUB 9000
1088 GO TO 1078
1090 GO SUB 9500
1092 GO TO 1078
1093 IF tv=1 THEN  GO TO 2000
1094 IF tv=2 THEN  GO TO 2016
2000 REM *Rotina de Verificacao*
2002 CLS
2004 PRINT "Verificacao de Entra
das": PRINT "==========
====="
2006 PRINT "Se OK entao ENTER"
2008 PRINT "Se errado Reentre"
2010 PRINT "Prima F para corrigi
r"
2014 LET fy=1: GO TO 1032
2016 REM ****Previsao ou Nao****
2018 CLS : PRINT "Quer uma previ
sao?": IF fl=1 THEN  PRINT "Nao
pode haver previsao com      meno
s de seis pontos"
2019 GO SUB 8500
2020 IF rf=1 THEN  GO TO 2029
2021 IF rf=3 THEN  GO TO 2026
2024 GO TO 2018
```



```
2026 GO SUB 9500
2028 GO TO 2018
2029 IF tv=1 THEN  GO TO 2031
2030 IF tv=2 THEN  GO TO 2500
2031 IF nm<6 THEN  LET fl=1: GO
TO 2018
2032 CLS : PRINT "Opcoes": PRINT
 "1) Previsao Exponencial Simple
s";" 2) Previsao Exponencial Dup
la";AT 16,0;"Entre 1 ou 2   ";:
LET d=1: GO SUB 7000
2034 IF rf=1 THEN  GO TO 2043
2035 IF rf=3 THEN  GO TO 2040
2036 LET b$="Ver texto para expl
icacao da    diferenca": GO SUB
9000
2038 GO TO 2032
2040 GO SUB 9500
2042 GO TO 2032
2044 LET fo=tv
2046 IF tv<1 OR tv>2 THEN  GO TO
 2032
2047 CLS
2048 PRINT AT 1,1;"Entre constan
te de Normalizacao  (Numero entr
e 0 e 1)";AT 3,5;: LET d=4: LET
de=1: GO SUB 7000
2049 IF rf=1 THEN  GO TO 2060
2050 IF rf=2 THEN  GO TO 2062
2051 IF rf=3 THEN  GO TO 2056
2052 LET b$="A CONSTANTE so'pode
 ter 4       caracteres": GO SUB
 9000
2054 GO TO 2048
2056 GO SUB 9500
2058 GO TO 2048
2060 IF tv>0 AND tv<1 THEN  LET
de=0: LET co=tv: GO TO 2064
2062 PRINT AT 15,1;"Constante fo
```

```
ra dos limites": GO TO 2048
2064 CLS : GO SUB 8750
2500 REM **Calculo de YM e YN**
2510 LET ym=0: LET yn=d(1)
2520 FOR n=1 TO nm
2530 IF d(n)>ym THEN  LET ym=d(n
)
2540 IF d(n)<yn THEN  LET yn=d(n
)
2550 NEXT n
2558 CLS
2560 PRINT AT 1,0;"Entre valores
 Max e min para o  eixo dos YY"
2562 PRINT AT 4,0;"A maior entra
da e'  ";ym
2564 PRINT AT 6,0;"A menor entra
da e'  ";yn
2566 PRINT AT 8,0;"Entre Y-max
     ";AT 8,12
2568 LET d=6: GO SUB 7000
2569 IF rf=1 THEN  GO TO 2582
2570 IF rf=2 THEN  GO TO 2580
2571 IF rf=3 THEN  GO TO 2576
2572 LET b$="Topo do Eixo": GO S
UB 9000
2574 GO TO 2566
2576 GO SUB 9500
2578 GO TO 2566
2580 PRINT AT 15,1;"Numero maior
 que 6 digitos": GO TO 2556
2582 PRINT AT 15,1;"
                   "
2584 LET am=tv: IF am<ym THEN  P
RINT AT 15,1;"Y-max < Maior entr
ada - Repita": GO TO 2560
2585 PRINT AT 15,1;"O Grafico va
                    Prima SPACE s
i surgir
e quizer continuar": PRINT : PRI
NT "Entre Y-min ";
```



```
2588 LET d=6: GO SUB 7000
2589 IF rf=1 THEN  GO TO 2602
2590 IF rf=2 THEN  GO TO 2600
2591 IF rf=3 THEN  GO TO 2596
2592 LET b$="Base do Eixo": GO S
UB 9000
2594 GO TO 2586
2596 GO SUB 9500
2598 GO TO 2586
2600 PRINT AT 15,1;"Numero maior
 que 6 digitos": GO TO 2586
2602 PRINT AT 15,1;"
                   "
2604 LET an=tv: IF an>yn THEN  P
RINT AT 15,1;"Y-min > menor entr
ada - Repita": GO TO 2586
3000 REM Calculo das Divisoes do
          eixo dos YY
3002 LET nd=am-an
3004 IF nd<11 THEN  GO TO 3014
3006 LET nd=nd/5: IF (nd-INT (nd
))<0.001 THEN  GO TO 3004
3008 LET nd=nd*5/2: IF (nd-INT (
nd))<0.001 THEN  GO TO 3004
3010 LET am=am+1
3012 GO TO 3002
3014 LET y1=INT (140/nd)
3016 LET x1=INT (200/(nm-1))
3018 LET sc=(y1*nd)/(am-an)
4000 REM Tracado do Eixo dos YY*
4002 CLS
4006 PLOT 10,10: DRAW 0,y1*nd
4008 FOR n=0 TO nd
4010 PLOT 10,10+y1*n: DRAW -4,0
4012 NEXT n
4014 REM *****Eixo dos XX*****
4016 PLOT 10,10: DRAW x1*(nm-1),
0
4018 FOR n=0 TO nm-1
```

```
4020 PLOT 10+n*x1,10: DRAW 0,-5
4022 NEXT n
5000 REM **Tracado dos Pontos**
5002 FOR n=0 TO nm-2
5004 PLOT 10+n*x1,10+d(n+1)*sc:
DRAW x1,(d(n+2)-d(n+1))*sc
5006 NEXT n
5007 IF INKEY$<>" " THEN  GO TO
5007
5008 CLS : PRINT "COORDENADAS DA
S AREAS DO GRAFICO"
5009 PRINT : PRINT "Y-max= ";am;
"     Y-min= ";an: PRINT "Escala
 Eixo YY= ";(am-an)/nd
5020 PRINT AT 15,0;"Entre 1)Ver
grafico de novo              2)Entr
ada de novos valores         3)Aban
dono ";: LET d=1: GO SUB 7000
5021 IF rf=1 THEN  GO TO 5031
5022 IF rf=3 THEN  GO TO 5028
5024 LET b$="Se premir 2 ou 3 pe
rdera'          os dados": GO SUB 9
000
5026 GO TO 5020
5028 GO SUB 9500
5030 GO TO 5020
5031 IF tv=1 THEN  GO TO 4002
5032 IF tv=2 THEN  GO TO 5034
5033 IF tv=3 THEN  GO TO 5028
5034 RUN
6000 REM **Rotina de Correccao**
6002 LET l=j: IF l>19 THEN  LET
l=19
6006 PRINT AT 20-l,22;"Correccao
"
6008 PRINT AT 21-l,25;"*";CHR$ (
8);: LET d=6: GO SUB 7000
6009 IF rf=1 THEN  GO TO 6016
6010 IF rf=2 THEN  GO TO 6022
```



```
6011 IF rf=3 THEN  GO TO 9300
6012 PRINT AT 21-l,25;" ";
6013 LET l=l-1: IF l<2 THEN  GO
TO 6024
6014 GO TO 6008
6016 INVERSE 0: PRINT AT 15,0;
6018 LET d(j-l+1)=tv: GO TO 6013
6020 INVERSE 1
6022 PRINT AT 15,0;"Numero maior
 que 6 digitos": GO TO 6008
6024 PRINT AT 20-l,1;: RETURN
7000 REM **Entrada de Numeros**
7002 LET tv=0: LET t$=" ": LET c
h=0
7004 LET a$=INKEY$: IF a$<>"" TH
EN  GO TO 7004
7005 LET a$=INKEY$: IF a$="" THE
N  GO TO 7005
7006 IF CODE (a$)=13 AND ch=0 TH
EN  LET rf=0: GO TO 7042
7010 IF CODE (a$)=12 THEN  GO TO
 7030
7012 IF CODE (a$)=13 THEN  LET r
f=1: GO TO 7040
7014 IF a$="@" AND ch=0 THEN  LE
T rf=3: GO TO 7042
7016 IF ch=d THEN  LET rf=2: GO
TO 7042
7018 IF a$="e" THEN  LET rf=4: G
O TO 7042
7020 IF a$>="0" AND a$<="9" THEN
  GO TO 7026
7022 IF de=1 AND a$="." THEN  GO
 TO 7026
7024 GO TO 7004
7026 LET t$=t$+a$: PRINT a$;: LE
T ch=ch+1
7028 GO TO 7004
7030 IF ch=0 THEN  GO TO 7004
```

```
7032 LET ch=ch-1: PRINT CHR$ (8)
;" ";CHR$ (8);
7034 IF ch=0 THEN  LET t$="": GO
 TO 7004
7036 LET t$=t$(LEN (t$)-1)
7038 GO TO 7004
7040 LET tv=VAL (t$)
7042 LET d=0: RETURN
8000 REM *****Entrada SPACE*****
8002 LET tv=0: LET t$=" ": LET c
h=0
8004 LET a$=INKEY$: IF a$=" " TH
EN  GO TO 8004
8005 LET a$=INKEY$: IF a$<>" " T
HEN  GO TO 8005
8034 LET rf=1: LET tv=1: RETURN
8500 REM **Entrada Sim ou Nao**
8502 LET rf=0: LET ch=0
8504 LET a$=INKEY$: IF a$<>"" TH
EN  GO TO 8504
8505 LET a$=INKEY$: IF a$="" THE
N  GO TO 8505
8506 IF a$="@" THEN  LET rf=3: G
O TO 8514
8508 IF a$="s" THEN  LET rf=1: L
ET tv=1: GO TO 8514
8510 IF a$="n" THEN  LET rf=1: L
ET tv=2: GO TO 8514
8512 GO TO 8504
8514 PRINT a$: RETURN
8750 REM **Modelo de Previsao**
8752 IF nm<12 THEN  LET k=6
8754 IF nm>=12 THEN  LET k=12
8756 DIM p(12): DIM q(19)
8758 DIM r(12): DIM s(19)
8760 LET p(1)=d(nm-k)
8762 LET r(1)=p(1)
8764 LET q(2)=p(1)
8766 REM *****Normalizacao*****
```



```
8768 FOR h=1 TO k-1
8770 LET a=nm-k+h
8772 LET p(h+1)=p(h)+co*(d(a)-p(
h))
8774 LET q(h+2)=p(h+1)
8776 LET r(h+1)=r(h)+co*(p(h+1)-
r(h))
8778 LET a=2*p(h+1)-r(h+1)
8780 LET b=co*(p(h+1)-r(h+1))/(1
-co)
8782 LET s(h+2)=a+b
8784 NEXT h
8786 FOR h=1 TO 6
8788 LET q(k+h+1)=p(k)
8790 LET s(k+h+1)=a+b*(h+2)
8792 NEXT h
8794 FOR h=1 TO 6
8796 IF fo=1 THEN  GO TO 8798
8797 IF fo=2 THEN  GO TO 8802
8798 LET d(nm+h)=q(k+h+1)
8800 GO TO 8804
8802 LET d(nm+h)=s(k+h+1)
8804 IF d(nm+h)<0 THEN  LET d(nm
+h)=0
8806 NEXT h
8808 LET nm=nm+6
8810 RETURN
9000 REM **Rotina de Auxilio**
9002 PRINT AT 15,0;"

                    "
9004 PRINT AT 15,0;b$: PRINT "Pr
ima SPACE para continuar": GO SU
B 8000
9006 IF rf=1 THEN  GO TO 9014
9008 GO TO 9002
9010 GO SUB 9502
9012 GO TO 9002
9014 PRINT AT 15,0;"
```

```
9016 RETURN
9500 REM ***Rotina de Abandono**
9502 CLS : PRINT AT 15,0;"Quer t
erminar? (S/N)": GO SUB 8500
9503 IF rf=1 THEN  GO TO 9506
9504 IF rf=2 OR rf=3 THEN  GO TO
 9502
9506 IF tv=2 THEN  GO TO 9510
9508 CLS : PRINT AT 10,10;"ATE'
BREVE!";AT 11,9;"============";
STOP
9510 CLS : PRINT AT 15,0;"
                ": RETURN
```


Capítulo 7

Contactos
Registo de clientes

INTRODUÇÃO

Como o nome sugere, um programa de base de dados ou ficheiro transforma um computador num sistema de arquivo. Uma vez armazenada, a informação pode ser chamada para consulta, e existem possibilidades de arquivar novos dados ou de apagar os que já não são precisos.

Parecerá, pelo que ficou dito, que afinal a base de dados não passa de mero substituto electrónico de qualquer pasta de arquivo, de modo que só marginalmente poderá oferecer algum benefício quando comparada com os sistemas manuais bem testados e de comprovada eficiência. No entanto, a realidade é outra: as bases de dados informatizadas constituem grande avanço sobre as suas congéneres manuais, pois nelas o processo de busca de informação pode ser programado. Por outras palavras, o computador pode ser instruído para percorrer todo o ficheiro, localizar e imprimir a informação da forma que se determinar. Esta capacidade de processar e analisar os dados só é limitada pela forma de codificação primária aplicada ao que se introduz nas memórias do computador.

A necessidade de obtenção de informações reveste importância vital em todas as tarefas administrativas e de escrituração. Para qualquer lado que nos viremos deparam-se-nos sempre pessoas a pedirem umas às outras:

— A lista dos empregados que ainda não pagaram o selo do automóvel

— A relação dos devedores com contas superiores a três meses

— O rol dos clientes que nada adquiriram nos últimos seis meses

e assim por diante.

Nos sistemas manuais, as respostas são obtidas por consulta de maços de fichas em cartão ou, pior ainda, através da morosa e por vezes complicada pesquisa em fólios agrupados em pastas. Com um programa de base de dados, o trabalho efectua-se por meios electrónicos e numa pequena fracção de tempo.

Os utilitários informáticos de arquivo dividem-se em duas categorias principais: a dos programas de uso específico e a dos de aplicação geral. Estes últimos são escritos de modo a tornarem-se o mais flexíveis possível, e contêm normalmente o seu próprio sistema — em geral uma linguagem de alto nível — de especificação do modo de manipulação da informação, de organização dos ficheiros e de formatação da apresentação, tudo a determinar pelo operador.

CONTACTOS

```
OPÇÕES:

1) INTRODUZIR NOVO CONTACTO
2) PESQUISAR O FICHEIRO
3) APAGAR UM CONTACTO
INTRODUZIR A OPÇÃO
```


"Contactos" é um programa muito simples de base de dados, especialmente concebido para o arquivo das informações normalmente usadas pelos vendedores. Como tal, armazena entradas únicas que compreendem o nome da firma ou companhia, o nome do contacto e o respectivo número de telefone. As entradas admitem qualquer ordem, e podem acrescentar-se novas fichas à medida que surjam outros contactos. Os dados são extraídos pela aplicação da vulgar técnica da "palavra-chave": trata-se de engenhosa forma de permitir ao operador ditar a finalidade da pesquisa que pretende; a máquina vai à procura das entradas que começam com a palavra-chave especificada, e o propósito da pesquisa depende do comprimento daquela. Se a chave tem só uma letra, o computador apresentará muitos mais dados do que se tiver 10 caracteres de comprimento.

```
CONTACTOS EM FICHEIRO

SILVA LDA. — CONTACTO
JOSÉ MARIA TEL. 91995997
SILVA LDA. — CONTACTO
ABEL SANTOS TEL. 234234234
SILVA LDA. (PORTO) — CONTACTO
RUI CASTRO TEL. 900507879
SILVA LDA. — CONTACTO
LUÍSA FAFE TEL. 564455445

FIM DE PESQUISA
OUTRA PESQUISA?
(S/N)
```

Para encontrarmos o nome ou nomes de determinada firma, temos de introduzir a palavra-chave apropriada. O programa lê então todo o ficheiro, listando as fichas que encontrar. A técnica das palavras-chaves encontra toda a sua justificação se tivermos em memória nomes parcialmente iguais. Se, por exemplo, constar do ficheiro um certo número de dependências de uma mesma firma — digamos, Silva Lda (Lisboa), Silva Lda (Porto) e Silva Lda (Viana do Castelo), todas as sucursais serão listadas se a palavra-chave for «Silva», enquanto que «Silva Lda (Porto)» só apresentará esta última.

DESCRIÇÃO DO PROGRAMA

"Contactos" é constituído por quatro secções principais, a saber:

— O *menu* geral,
— A entrada de dados, com a qual introduzimos toda a informação,
— A pesquisa, em que as fichas iniciadas por determinada palavra-chave são identificadas e apresentadas, e
— O apagamento de dados, com o qual podemos eliminar os contactos desnecessários.

Como se faz muito pouco uso do teclado, as sub-rotinas de entrada de dados foram simplificadas.

O «MENU» GERAL

O programa inicia-se na linha 102, em que **F$( )** — o grupo em que os dados vão ser armazenados — é dimensionado com **B**. Este tem o valor 200, mas podemos alterá-lo caso o pretendamos. As linhas 106 e 108 contêm o título do programa.

Os dados são permanentemente guardados em *microfloppy*, e carregados no programa sempre que este é inicializado. Contudo, na primeira utilização de "contactos", este ficheiro tem de ser criado; qualquer tentativa de carregamento de um ficheiro não existente poderia causar o «rebentamento» do programa. O problema é resolvido pelas linhas 112 a 118, que perguntam se tal arquivo já existe: caso afirmativo, procede-se ao seu carregamento (sub-rotina 5500), e se tal se não verificar a rotina é ultrapassada.

As linhas 126 a 139 apresentam as opções do *menu* e registam a escolha feita pelo operador. Repare que o valor de **T$** só pode ser **1, 2 ou 3**. Se o teclado indicar outro número que não estes, o programa regressa ao princípio via linha 140. A bandeira **FM** é estabelecida em **1** se se escolher a opção 3 ("Apagamento").

ENTRADA DE DADOS

Uma entrada completa compreende três tipos de informação: o nome da firma, o nome do contacto e o respectivo número de telefone. Estes três dados são agrupados e armazenados como um único elemento do grupo **F$( )**.

O programa para a entrada de dados começa na linha 1000. De 1006 a 1018, procede-se ao agrupamento dos pormenores de cada contacto em três *strings*, **N$, C$** e **T$**. Seguidamente, estas 3 variáveis literais são combinadas numa só, **R$**, com cada tipo de informação separado por uma variável divisória, sob a forma:

«CONTACTO ...»

A variável **D** limita o comprimento das variáveis literais de entrada (**N$**, etc.). A primeira pode ter um máximo de **20** caracteres, e as outras duas ficam restringidas a **10**. As variáveis divisórias possuem 25 caracteres, de modo que uma só entrada tem um total de 65 caracteres — isto é, um pouco menos de duas linhas do *écran*, que tem 40 colunas.

NO, na linha 1020, é um contador que vai registando o número de entradas, avançando uma unidade cada vez que se enche uma ficha. Desde que haja espaço no grupo, **R$** transforma-se em **F$(NO)**, isto é, no elemento de ordem **NO** do grupo **F$( )** (linha 1026). Quando **NO** atinge a dimensão de **F$( )**, o ficheiro fica cheio, e a linha 1022 evita que se façam mais entradas. Para tal, o programa assume que a entrada de dados

acabou, e procede para ultrapassar as linhas 1026 a 1030.

Depois de terminar uma entrada, o operador é colocado perante a possibilidade de continuar ou de dar por finalizada a memorização de informações, com o concomitante regresso ao *menu* geral. No primeiro caso, o desvio respectivo (**RF=1**) — linha 1032 — provoca o retorno ao início da rotina de entrada de dados, na linha 1002. O segundo desvio (**RF=2**), segunda opção, grava em primeiro lugar o registo completo **F$( )** na *microfloppy*, através da sub-rotina 5000, e só depois efectua o regresso ao *menu* principal.

PESQUISA

A rotina de busca começa por perguntar qual a palavra-chave (linhas 2004 a 2008), descobrindo qual o seu comprimento, **LE** (linha 2008), e depois pesquisando **F$** para descobrir quais os primeiros caracteres **LE** idênticos a **S$**. A pesquisa é executada por um círculo que corre de **1** a **NO**, entre as linhas 2012 e 2016. Cada elemento de **F$(N)** é extraído por sua vez (2014), ficando com os seus primeiros caracteres **LE** isolados; a variável de texto resultante é comparada com **S$**. Se os primeiros caracteres **LE** de **F$(N)** forem os mesmos que a palavra-chave, o programa desvia-se para a sub-rotina 6000, que imprime na linha 6008 a totalidade de **F$(N)**. Se não forem idênticos, o programa «passa por cima» das linhas 2014 a 2016 e avança o *loop* para aceitar a entrada seguinte destinada a **F$(N)**.

Existem determinadas condições especiais a ser tratadas. Por exemplo, é possível que o programa encontre mais fichas correspondentes a uma palavra chave do que as que cabem no *écran*. Se tal acontecesse, a imagem entraria em rolamento *(scroll)*, provocando o desaparecimento de parte da informação pretendida. Para evitarmos este senão, o rolamento é colocado sob controle do operador, nas linhas 6000 a 6008, que restringem o número de entradas a apresentar a 5 — o número mais conveniente para se encher todo o *écran* se as fichas tiverem todas duas linhas de comprimento. Se **L** ficar maior que **5**, o programa pára até que o operador prima a barra de espa-

ços, ponto em que o contador **L** é reestabelecido em **0** e o *écran* é apagado com a instrução CLS da linha 6007. A bandeira **FL** da linha 6008 encarrega-se da eventualidade de não existirem entradas em todo o **F$(N)** que correspondam à palavra-chave especificada. Neste caso, **FL** é estabelecida em **0** no início da rotina de pesquisa (linha 2002). Mal se encontre uma ficha, **FL** passa a **1** na linha 6008. Se, no fim da pesquisa, **FL** ainda estiver em **0**, a linha 2018 trata de imprimir a mensagem apropriada (fim de pesquisa ... contacto inexistente).

Podemos de momento ignorar a bandeira **FM** (linhas 2015 e 2026) que faz parte da rotina de "Apagamento", a descrever mais adiante. **FM** tem o valor **0** durante a rotina de pesquisa.

Esta acaba pela mensagem «**Quer outra pesquisa?**», impressa pela linha 2020. Se a resposta for **S** (sim), o programa regressa ao princípio da rotina de busca (linha 2000). Se teclarmos **N** (não), provocaremos o salto para o *menu* geral, na linha 122.

APAGAMENTO

O "Apagamento" é uma ramificação da rotina de pesquisa. A opção 3 do *menu* geral leva-nos à linha 142, onde se estabelece a bandeira **FM** em **1**. A partir daqui, o operador especifica o nome a ser apagado, e a busca é feita pelo mesmo programa usado para a opção 2. Quando a primeira entrada é encontrada, **FM** encaminha o programa para a sub-rotina 3000.

As linhas 3002 e 3004 perguntam se a entrada é para ser apagada ou não. Se a resposta for afirmativa, o respectivo valor de **F$( )** é estabelecido em " " (valor nulo) na linha 3006. Caso não queiramos eliminar a entrada, o programa regressa à linha 2016 e inicia mais uma volta de busca na rotina de pesquisa. Se se verificou alguma eliminação, as restantes entradas de **F$( )** têm de ser deslocadas no sentido ascendente, por forma a cancelar o espaço deixado vago; a subida é feita nas linhas 3008 a 3016: o valor de **N** dá a posição da entrada de **F$( )** que acabou de ser eliminada, e a linha 3010 atribui a **F$(N)** o valor de **F$(N+1)**, isto é, o valor da entrada imediatamente a seguir à que foi cancelada. Este processo repete-se para todos os ele-

mentos compreendidos entre N e o término das entradas em NO, usando o *loop* **FOR/NEXT** das linhas 3008 a 3012. Note-se que no fim desta operação, aparece uma entrada no fim do grupo que não deveria aí estar; a linha 3014 encarrega-se de a apagar. Por fim, e como se apagou uma ficha, **NO** (o número de entradas) é diminuído em uma unidade (linha 3016).

O programa regressa depois para a rotina de pesquisa, percorrendo o círculo de busca até descobrir todas as entradas. A rotina "Apagamento" sai do programa de pesquisa na linha 2026, quando o operador faz saber que não são necessárias mais voltas de busca; o programa desvia-se então para a sub-rotina 5000, em que se processa a correcção da *microfloppy*. Finalmente, o operador regressa ao *menu* geral via linha 2028.

```
10 REM PROGRAMA CONTACTOS
11 REM  (Base de Dados)
12 LET no=0
14 REM c$= Variavel do nome
16 REM d=Comprim. da Variavel
20 REM f$=Dados do Ficheiro
22 REM fm=Instrucao de apagar
          bandeiras
24 REM l=contador de paginas
26 REM le=Comprim. da Variavel
       de pesquisa
28 REM m e n sao contadores
    nos loops FOR / NEXT
30 REM n$=Variavel de Nome
32 REM no=Numero de Entradas
       em Ficheiro
34 REM r$=Ficha
36 REM rf=saida de bandeiras
   nas subrotinas de entrada
       de dados
38 REM s$=Variavel de Pesquisa
40 REM q$=Variavel de Numero
       de Telefone
42 REM          t$=Variavel de
```



```
Transferencia das subrotinas de
      entrada de dados
 100 REM Ficheiro de Contactos
 102 LET b=200: DIM f$(b,45)
 106 CLS : PRINT AT 10,10;"CONTA
CTOS"
 108 PRINT AT 11,10;"==========
"
 110 REM Carregamento
 112 PRINT AT 13,4;"Ja' tem um f
icheiro de                contac
tos?"
 114 PRINT AT 15,0;"(S ou N)";
 116 GO SUB 8500
 118 IF rf=1 THEN  GO SUB 5500
 122 REM ***Menu Principal***
 124 CLS : LET fm=0
 125 PRINT AT 2,0;"Voce quer:"
 128 PRINT AT 4,1;"1) Entrar nov
o contacto"
 130 PRINT AT 5,1;"2) Pesquisar
o ficheiro"
 132 PRINT AT 6,1;"3) Apagar um
contacto"
 134 PRINT AT 9,0;"Entre a opcao
";AT 9,30;
 135 LET d=1: GO SUB 7000
 136 IF t$<"1" OR t$>"3" THEN  G
O TO 135
 137 IF t$="1" THEN  GO TO 1002
 138 IF t$="2" THEN  GO TO 2000
 139 IF t$="3" THEN  GO TO 142
 140 GO TO 134
 142 LET fm=1: GO TO 2000
1000 REM Impressao de Contactos
1002 REM Entrada de Dados do
          Contacto
1004 CLS : PRINT "Entre os dados
 do Contacto": PRINT "==========
================"
```

```
1006 PRINT AT 5,1;"Nome.....";
: LET d=20: GO SUB 7000
1008 LET n$=t$
1010 PRINT AT 7,1;"Contacto...";
: LET d=10: GO SUB 7000
1012 LET c$=t$
1014 PRINT AT 9,1;"Telef. ....";
: LET d=10: GO SUB 7000
1016 LET q$=t$
1018 LET r$=n$+"  Cont:  "+c$+"
                  Telef. "+q$
1020 LET no=no+1
1022 IF no<b+1 THEN  GO TO 1026
1024 PRINT AT 8,1;"Ficheiro chei
o. Nao pode entrar mais contacto
s.                    Entre N para
 voltar ao Menu.": GO TO 1031
1026 LET f$(no)=r$
1028 PRINT AT 19,0;"Quer entrar
outro Contacto?"
1030 PRINT AT 20,0;"S ou N"
1031 GO SUB 8500
1032 IF rf=1 THEN  GO TO 1002
1034 GO SUB 5000
1036 GO TO 122
2000 REM **Pesquisa por Nome**
2002 LET f1=0: LET l=0
2004 CLS : PRINT "Pesquisa": PRI
NT "========="
2006 PRINT AT 3,1;"Entre o Nome
";: LET d=20: GO SUB 7000
2008 LET s$=t$: LET le=LEN (s$)
2010 PRINT AT 5,1;"Contactos em
Ficheiro:"
2012 FOR n=1 TO no
2014 LET g$=f$(n): IF le>LEN (g$
) THEN  GO TO 2016
2015 IF g$(1 TO le)=s$ THEN  GO
SUB 6000: IF fm=1 THEN  GO SUB 3
000
```



```
2016 NEXT n
2018 PRINT "Fim de Pesquisa";: I
F f1=0 THEN  PRINT ".... Nao exi
ste   esse contacto"
2020 PRINT : PRINT "Quer outra p
esquisa? (S/N)"
2022 GO SUB 8500
2023 IF rf=1 THEN  GO TO 2000
2024 IF fm=1 THEN  GO SUB 5000
2028 CLS : GO TO 122
3000 REM ***Apagar Contactos***
3002 PRINT "Apagar? (S/N)": GO S
UB 8500
3004 IF rf=2 THEN  GO TO 3018
3006 LET f$(n)=""
3008 FOR m=n TO no
3010 LET f$(m)=f$(m+1)
3012 NEXT m
3014 LET f$(m)=""
3016 LET no=no-1
3018 RETURN
5000 REM **Gravar Ficheiro***
5010 SAVE "CONTACTOS" DATA f$()
5022 RETURN
5500 REM **Carregar Ficheiro**
5510 LOAD "CONTACTOS" DATA f$()
5516 RETURN
6000 REM   Impressao da Lista
           de Contactos
6002 IF l<5 THEN  GO TO 6008
6004 PRINT "Prima SPACE para con
tinuar"
6006 LET a$=INKEY$: IF a$<>" " T
HEN  GO TO 6006
6007 LET l=0: CLS
6008 PRINT f$(n): LET f1=1
6010 LET l=l+1: RETURN
7000 REM Entrada de Dados aceita
         todos os caracters do
              teclado
```

```
7002 LET t$="": LET ch=0
7004 LET a$=INKEY$: IF a$<>"" TH
EN  GO TO 7004
7005 LET a$=INKEY$: IF a$="" THE
N  GO TO 7005
7006 IF ch>d THEN  GO TO 7032
7008 IF CODE (a$)=13 AND ch>0 TH
EN  GO TO 7032
7010 IF CODE (a$)=12 THEN  GO TO
 7018
7012 IF a$>=" " AND a$<="z" THEN
  GO TO 7028
7016 GO TO 7004
7018 IF ch=0 THEN  GO TO 7004
7020 LET ch=ch-1: PRINT CHR$ (8)
;" ";CHR$ (8);
7022 IF ch=0 THEN  LET t$="": GO
 TO 7004
7024 LET t$=t$(1 TO LEN (t$)-1)
7026 GO TO 7004
7028 LET t$=t$+a$: PRINT a$;: LE
T ch=ch+1
7030 GO TO 7004
7032 RETURN
8500 REM ***Resposta S ou N***
8502 LET ch=0: LET rf=0
8504 LET a$=INKEY$: IF a$<>"" TH
EN  GO TO 8504
8505 LET a$=INKEY$: IF a$="" THE
N  GO TO 8505
8506 IF ch>1 THEN  GO TO 8514
8508 IF a$="s" THEN  LET rf=1: G
O TO 8514
8510 IF a$="n" THEN  LET rf=2: G
O TO 8514
8512 GO TO 8504
8514 PRINT a$: RETURN
```


Capítulo 8

Quem vende o quê

INTRODUÇÃO

"Contactos" (capítulo 7) é um programa muito simples de base de dados, em que cada registo é constituído por uma única variável literal; quando se extraem os dados, estes continuam a assumir a forma pela qual foram introduzidos. Isto é, o programa extrai um só campo: só vai à procura do nome do contacto, não tentando localizar quer o nome da empresa quer o número de telefone.

Para aplicações mais pormenorizadas, qualquer programa de ficheiros tem de conseguir reorganizar os dados "em bruto", para os poder apresentar conforme as necessidades de análise do operador. Na realidade, um programa de base de dados tem duas funções: estabelecer e manter fichas de dados (a base) e providenciar meios de apresentação que possam ser especificados pelo utilizador.

Os dados são memorizados à medida que se verifica cada venda. Quando se pretende a análise, o programa "quem" produz tabelas que mostram diferentes arranjos dos dados originais. O processo de rearranjo dos diversos valores é o de grupo encadeado, descrito no capítulo 3.

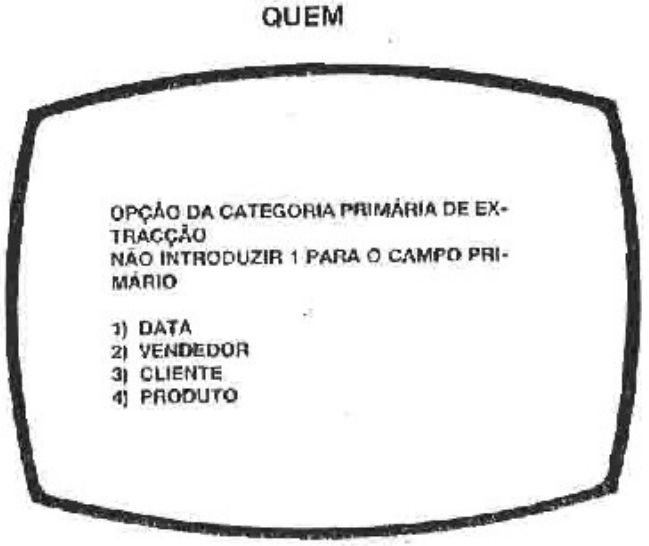
A base de dados compreende certo número de fichas com valores de vendas, cada uma contendo o registo completo das transacções de um mês. Parte da ficha respeita aos dados directamente relacionados com a venda em si (o valor das vendas), e o espaço restante é ocupado por dados pertinentes — data, nome do vendedor, etc. Em lugar de se duplicarem os dados comuns ao longo de todas as fichas, preferimos armazená-los logo no início, através do uso de "directorias", referenciadas por uma entrada no ficheiro dos dados de vendas. Deste modo,

em vez de escrevermos o nome do vendedor cada vez que se regista uma venda, basta introduzirmos um código numérico que se refere a dada entrada da directoria dos vendedores.

Sempre que registamos uma transacção, armazenamos os seguintes dados no ficheiro de valores de venda:

— A data da venda,
— O número do vendedor (em referência cruzada com um ficheiro de nomes),
— O número do cliente (em referência cruzada com um ficheiro de nomes),
— O código do produto (em referência cruzada com a descrição),
— O valor da venda.

Há três ficheiros de directorias, que contêm:



— Os nomes dos vendedores,
— Os nomes dos clientes,
— Os nomes (ou descrições) dos produtos.

Para analisar a base de dados, o operador especifica o período que pretende examinar, entrando o ano e o mês do início e do fim. O programa procede então ao carregamento para a memória do conteúdo dos ficheiros de dados de venda apropriados. De seguida o operador escolhe os campos de análise dos dados; por exemplo, poderá começar pela análise por cliente, passando depois para a verificação por datas, caso em que os dados seriam em primeiro lugar divididos em "vendas por cliente" (todas as vendas para o cliente 1, depois todas as vendas para o cliente 2, e assim por diante), e em segundo lugar alinhados por datas dentro de cada categoria de cliente (todas as transacções com o cliente 1 ordenadas cronologicamente, seguidas pelo mesmo arranjo em relação ao cliente 2, e assim sucessivamente).

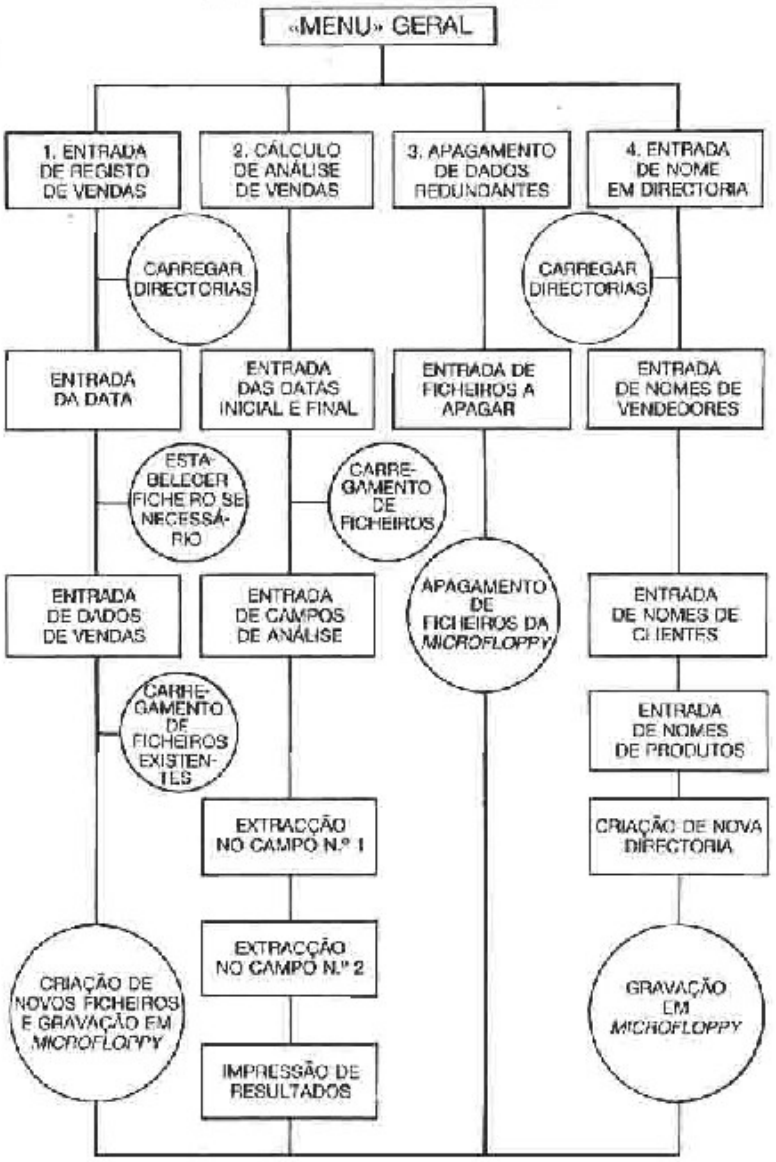
O volume de dados que pode ser analisado de uma só vez é função, claro está, do espaço de memória disponível do computador. Como temos de armazenar um certo número de ficheiros diferentes de cada vez, é essencial manter uma distribuição mais ou menos equitativa de espaço de memória entre eles. Torna-se evidente que será deveras indesejável podermos armazenar dados referentes a centenas de clientes e só sermos capazes de registar ao mesmo tempo umas poucas de vendas. A melhor solução depende das características da actividade comercial a processar pela base de dados. A dimensão dos ficheiros é determinada pelos valores das constantes e pelas dimensões dos grupos e matrizes estabelecidos logo nas primeiras linhas do programa.

Da maior importância neste tipo de programas é o facto de os tornarmos "à prova de erros". Será que devemos evitar que o operador coloque a fita errada na *microdrive*? O que acontece se ele chamar um ficheiro que não existe? É claro que é perfeitamente possível prevenirmos todos os erros, mas o programa ficaria muito comprido e bastante complexo, para além

de se ir gastar espaço de memória precioso. O programa "Quem" assume uma posição de compromisso, prevenindo os erros mais típicos geralmente cometidos pelo operador, mas não evita as tentativas deliberadas de "enganar" a máquina.

A estrutura do programa é assinalada no fluxograma que se segue:

QUEM — FLUXOGRAMA SIMPLIFICADO1



DESCRIÇÃO DO PROGRAMA

"Quem" começa por um *menu*, com as seguintes opções:

1) Entrada de registos de venda.
2) Análise de vendas.
3) Apagamento de dados redundantes.
4) Entrada de nomes em directorias.
5) Estabelecimento de nova base de dados.
6) Abandono do programa.

ESTABELECIMENTO

A opção 5) só é usada quando se pretende criar uma base de dados completamente nova. Tem a finalidade de "abrir" os ficheiros de directorias, nos quais iremos carregar dados em fase posterior. Este procedimento evita que o computador não consiga "encontrar" um ficheiro quando se lhe pede que grave as primeiras entradas de directorias. "Estabelecimento" cria ficheiros "simulados", cada um dos quais contém uma única entrada (o número – 999999), para significar que acabaram de ser criados.

Esta mesma opção chama a sub-rotina 6000, que, depois de avisar o operador de que quaisquer ficheiros já existentes na *microfloppy* poderão ser apagados, cria três ficheiros: **Sadir** (directoria de vendedores), **Cudir** (directoria de clientes) e **Prdir** (directoria de produtos); a cada uma destas aplica-se uma só entrada, – 999999.

CRIAÇÃO DE DIRECTORIAS

A opção 4) carrega os dados para as três directorias. Inicialmente, o ficheiro não contém dados (só o – 999999); porém, depois da base de dados estar em uso, a directoria conterá uma lista de nomes, que podem ter de ser emendados ou adicionados à medida que o tempo passa. O número máximo de entadas que cada directoria pode conter é determinado pelo valor atribuído às variáveis **NS**, **NC** e **NP**, nas linhas 104 a 108. A opção 4) chama a sub-rotina 6500, cuja primeira etapa estabelece um

círculo que abre e depois lê cada um dos ficheiros das três directorias, **Sadir**, **Cudir** e **Prdir**. Um ficheiro vazio é reconhecido se a sua primeira entrada for – 999999, caso em que será imediatamente fechado pela linha 6514. Se o ficheiro já contiver dados, a primeira entrada será constituída pelo número de registos desse ficheiro, **NR**. Estes são lidos para o grupo **B$( )** nas linhas 6515 a 6520.

A explicação que se segue é muito parecida com a da rotina de entrada de dados usados em "Gráficos" (capítulo 5). O *loop* **N**, que corre da linha 6516 à 6520, conta o número de registos a entrar no grupo **B$( )**, de modo que no fim do círculo, **B$( )** contém os registos **NR** que foram carregados do ficheiro em presença. Neste ponto, a bandeira **FY** é estabelecida em **1** (linha 6524). Estes registos "antigos" vão formar a primeira parte da nova directoria, de modo que, antes de introduzirmos novos dados, têm de ser apresentados em *écran*, para eventualmente serem corrigidos. O conteúdo de **B$( )** é apresentado injectando cada elemento de **B$( )** na sub-rotina de *input* 8550, como se se tratasse de uma entrada via teclado: estabelece-se **T$** e **CH** no valor do elemento apropriado de **B$( )**, na linha 6542 e depois passa-se à rotina mencionada através da linha 8554. A sub-rotina 8550 assume então o comando e trata a entrada como se esta fosse um sinal parcialmente completo do teclado, esperando que seja recebido outro valor de **A$**. Para emendar uma entrada, o operador recua os espaços adequados e escreve o novo texto. O sinal **RETURN** provoca a saída **RF = 1** da sub-rotina 8550, transferindo o conteúdo de **T$** para **B$( )** e colocando os dados originais ou corrigidos na nova directoria.

Depois de apresentadas todas as entradas primitivas (em número de **NR**), a condicionante **J < NR + 1** (linha 6542) perde a efectividade, de modo que a próxima entrada passa para a linha 6544, que cancela o apagamento (**FY = 0**) e aumenta o valor de **NR** para **NT**. Isto permite que se aplique a rotina de correcção aos novos dados a entrar. Repare que se estabeleceu **NT** para a capacidade máxima da directoria, na linha 6504.

Se, em qualquer ponto da rotina de entrada de dados, o operador pretender corrigir o que escreveu, basta-lhe premir a tecla X, que provoca uma saída **RF = 4** da sub-rotina 8550 e imediato acesso à sub-rotina 6750. Esta progride através de todas as entradas até ao momento efectuadas (das quais existem L), injectando uma de cada vez na sub-rotina de entrada de dados 8550.

As entradas podem ser: 1) aceites (prima **RETURN** para uma saída **RF = 0**, o que provoca novo circuito do *loop* na linha 6763) ou 2) corrigidas, caso em que o regresso se verifica através de **RF = 1**, ficando o elemento apropriado de **B$( )** estabelecido no novo valor, ou finalmente, 3) podem estar incorrectas e portanto **RF = 2** conduz o programa para a linha 6772 para a apresentação da mensagem de erro, seguindo-se o regresso à linha 6758. Quando o processo de correcção fica completo, o programa sai da sub-rotina 6750, regressa à linha 6572 e depois passa à linha 6532 para recolher a próxima entrada na directoria.

O operador assinala o término da entrada de dados através da tecla @, o que provoca a saída **RF = 5** da sub-rotina 8550 (linha 6551), sendo-lhe novamente perguntado se quer proceder a mais alguma correcção. Se a resposta for afirmativa, o programa repete completamente toda a rotina de entrada de dados, regressando à linha 6530 depois de ter estabelecido **FY** em 1. Este facto tem como efeito a apresentação dos novos valores de **B$( )**, de forma a que o operador possa acrescentar ou emendar a entrada. No caso de não serem necessárias correcções (entrada terminada, não sendo preciso emendar a linha 6578), provoca-se a saída para a linha 6606, em que se cria a primeira directoria, **Sadir**.

Depois de criado o ficheiro, temos de introduzir de novo valores para **D$** e **J$** (linha 6622), repetindo-se todo o processamento para as suas outras directorias, **Cudir** e **Prdir**. Quando todos os ficheiros estiverem gravados, o operador é conduzido para o *menu* geral, na linha 1002.

ENTRADA DE DADOS DE VENDAS

Os dados sobre vendas podem ser introduzidos à medida que forem sendo conhecidos. Mal se introduza qualquer transacção, cada item da informação é encaminhado para a coluna apropriada da matriz quíntupla **R$(,)**. A primeira coluna (#1) contém a data, a segunda o número do vendedor, a terceira o número do cliente, a quarta o número de código do produto e por fim a quinta apresenta o valor da venda. Os registos das vendas são acumulados num ficheiro, cujo nome compreende o ano e o mês em que se introduziram os dados. Por exemplo, FIC 8609 refere-se a Setembro de 1986.

QUEM

DATA	VENDEDOR	CLIENTE	PRODUTO	VENDAS
840406	1	1	5	12.12
840408	1	2	5	12.56
840406	3	3	1	12.45
840408	3	3	1	33.67
840406	4	3	4	34.89
840408	4	4	2	16.34

PREMIR «SPACE» PARA CONTINUAR

A rotina de entrada de dados começa na linha 1500, em que se procede ao carregamento do conteúdo das três directorias (vendedor, cliente e produto) para os grupos **S$( )**, **C$( )** e

P$( ), sob o controle da bandeira FL. O número de entradas em cada directoria está presente em N(1) a N(3).

A fase seguinte consiste em se perguntar ao operador se se trata da primeira entrada do mês. No caso da resposta ser S (sim), a bandeira (FL) é estabelecida em 1 (linha 1552). Esta bandeira provoca a criação do "simulacro" (– 999999) de ficheiro, desde que se tenham introduzido o ano e o mês, através da linha 1582 que se chama a sub-rotina 8900.

Daqui para a frente, a rotina de entrada de dados segue o padrão familiar, já descrito nos programas anteriores. No caso presente, a parte da rotina que processa a entrada do ano e do mês foi colocada na sub-rotina 8700. Quando se torna necessário, os números do dia e do mês são artificialmente constituídos por dois caracteres (linhas 1578 e 8740).

Uma vez introduzida a data, procede-se à introdução dos pormenores da transacção num *loop* M com início na linha 1590. O total de transacções que podem ser introduzidas de uma só vez fica limitado pelo valor de A, que pode ser estabelecido para qualquer valor até à dimensão de R$(,), por sua vez determinada na linha 114.

As linhas 1586 e 1688 seguem o procedimento habitual. A bandeira FY, estabelecida em zero no início, torna-se 1 se mais adiante se seleccionar o modo de correcção. A bandeira DE serve para controlar a admissão do ponto decimal na sub-rotina 7000. Os códigos para o vendedor, o cliente e o produto são processados solicitando ao operador que introduza o número de referência apropriado. Mal seja obtido este dado, o computador apresenta a entrada completa da directoria, mantida em S$( ) (isto é, apresenta o nome completo do vendedor à frente do respectivo número, ou o nome do produto ou do cliente, conforme o caso). O processo é executado pela linha 1620, mas é igualmente verificado se o número de referência está dentro dos limites da directoria, na linha 1618. Existe a oportunidade de corrigir dados no fim da entrada destes (linha 1724), quando se pergunta ao operador se quer modificar as informações que acabou de escrever. Se for o caso, a bandeira FY é estabelecida

em 1 e o limite superior do círculo M (variável A) é estabelecido em NE, o número de entradas; o programa regressa depois ao começo da rotina, na linha 1590. Por fim, as entradas NE de R$(,) são gravadas num ficheiro cujo nome é constituído pelo ano e mês indicados no início da entrada de dados (linha 2000).

Cria-se a seguir novo ficheiro, carregando em primeiro lugar todos os dados que já se encontram registados em R$(,) imediatamente "atrás" das informações acabadas de entrar pela primeira vez. Para tal, o contador N é estabelecido de forma a começar em NE + 1, o primeiro elemento vago, e a acabar em NE + NR, que é agora o número total de registos. A etapa final consiste em refazer o ficheiro gravando todos os elementos de RE$(,), de 1 a NT, na *microfloppy*.

ANÁLISE DE VENDAS

A opção 2) chama à sub-rotina 3000, cuja finalidade é a de analisar os registos de vendas e produzir uma listagem tabular, seleccionada pelo operador, das transacções efectuadas entre duas datas especificadas.

O primeiro passo consiste em introduzir as suas datas, inicial e final, que determinam quais os ficheiros mensais apropriados a colocar na memória. As linhas 3000 e 3054 encarregam-se desta operação. As datas inicial e final (primeiro o ano e depois o mês) são introduzidas através da sub-rotina 8700. O primeiro valor é convertido nas viaráveis numéricas equivalentes T e D, que vão contando os dias até estes se tornarem iguais à data final (linhas 3048 a 3052). Cada ficheiro cuja data resida entre o início e o fim do período em análise é aberto e o respectivo conteúdo carregado para R$(,), usando-se NT e NE para posicionar os dados. Com estes alojados, passa-se à análise propriamente dita. Para compreendermos como se processa esta fase, temos de estudar mais em pormenor o modo de utilização das cadeias.

EXTRACÇÃO DE CADEIAS

Se o operador escolheu a extracção de dados de acordo com as categorias de produtos contidas na coluna 4 de R$(,), cada fila da matriz tem de ser reposicionada de modo a que a coluna 4 fique em ordem numérica descendente. Este facto é ilustrado no diagrama abaixo, em que a matriz

FILA	DATA	VENDEDOR	CLIENTE	PRODUTO	VALOR
1	02	4	2	1	20.05
2	02	2	4	1	40.00
3	04	1	17	2	100.00
4	04	6	14	1	55.20
5	06	3	2	2	45.00
6	08	5	1	2	33.24

tem de ser extraída de acordo com a tabela

FILA	DATA	VENDEDOR	CLIENTE	PRODUTO	VALOR
				1	
				1	
				1	
				2	
				2	

Um "Grupo encadeado" separado torna possível fugirmos ao moroso processo de reposicionamento de toda a matriz. Em lugar deste, uma variável literal de ponteiros indicará a ordem em que as filas têm de ser listadas de modo a se obter a sequência correcta.

O programa abaixo apresentado mostra como isto se faz:

```
10 LET C(4)=2
20 LET C(2)=1
30 LET C(1)=6
40 LET C(6)=5
50 LET C(5)=3
60 LET C(3)=0
70 LET H=4
80 IF H=0 THEN GOTO 120
90 FOR M=1 TO 5: PRINT R$(H,M): NEXT M
100 LET H= C(H)
110 GOTO 80
120 STOP
```

O grupo encadeado é definido nas linhas 10 a 60; é depois usado para extrair as filas apropriadas da matriz R$(,), pelo processo iniciado na linha 70. A primeira entrada de H, que é a cabeça da cadeia, dá início à sequência. Na linha 100, cada valor de C( ) "aponta" para a fila seguinte a ser impressa, C(4) aponta para a fila 2; C(2) aponta para a fila 1, e assim sucessivamente. O encadeamento termina quando o valor zero de C(3) — a cauda da cadeia — é detectado na linha 80.

Este programa forma a base da rotina de impressão usada em Quem a partir da linha 3116.

Como se cria então uma cadeia como esta? Na coluna 4 da matriz RE$(,), o primeiro elemento do grupo encadeado, a cabeça, tem de se referir a uma das filas que contenham o produto 1. Isto é a qualquer das filas 1, 2 ou 4. Estas podem ser identificadas por uma simples pesquisa através da coluna, por meio do programa seguinte:

```
10 FOR R=1 TO 6
20 IF R$(R,4)="1" THEN LET H=R
```

```
30 NEXT R
40 PRINT HS
```

No fim da execução do programa, o valor de **H** será a última fila da matriz a conter um **1** na coluna 4, isto é, a fila **4**. Este valor fixa a posição da cabeça da cadeia. O primeiro elo desta é o valor de **C(4)**, que é estabelecido ("apontado para") no número da fila seguinte da ordenação. Esta fila é outro "1" ou, se tal não suceder, o valor mais elevado que se seguir. Desde que haja outra entrada **1** na coluna, o valor será encontrado entre as filas **1** e 3 do seguinte modo:

```
10 FOR R=1 TO 3
20 IF RE$(R,4)="1" THEN LET C(4)=R
30 NEXT R
40 PRINT C(4)
```

O resultado é **C(4)=2**, se o processo for repetido mais uma vez para **R=1** a 2, então **C(2)=1**. Deste modo, esgotamos as entradas "1" da lista. Quando o programa corre de novo **C(1)** assume o valor zero. Agora o valor mais elevado que se segue na coluna 3 é, neste caso, **2**: se o programa correr com este novo valor, faz com que a cabeça da cadeia seja **6**, e que **C(6)=5**, **C(5)=3** e **C(3)=0**. O resultado completo é o seguinte:

LISTA "1"	LISTA "2"
H=4	H=6
C(4)=2	C(6)=5
C(2)=1	C(5)=3
C(1)=0	C(3)=0

Para criarmos um grupo simples que liste toda a coluna, o elo **C(1)** é tornado igual a **H** da lista 2 — como se uníssemos uma corrente de bicicleta —, com os seguintes resultados:

ou na ordem correcta:

H=4	H=4
C(4)=2	C(1)=6
C(2)=1	C(2)=1
C(1)=6	C(3)=0
C(6)=5	C(4)=2
C(5)=3	C(5)=3
C(3)=0	C(6)=5

Esta é a sequência original com que se criou a ordem correcta de **R$(,)** no início da secção.

Contudo, o arranjo prévio de 2 cadeias, cada uma a terminar por entradas zero, é de maior utilidade, já que proporciona um modo conveniente de assinalarmos o término de uma categoria e o próximo começo de outra. O elo de ligação entre as duas cadeias pode ser estabelecido como:

```
IF C( )=0 THEN LET H=6: REM Novo valor da cabeça
```

Para terminar, é conveniente incorporar a informação existente na cabeça das cadeias nos próprios grupos encadeados, em lugar de os definir separadamente. Para o conseguirmos, temos de reurranjar os subscritos de modo a que a cabeça de cada cadeia seja atribuída a um dos primeiros elementos do grupo. Neste caso **C(1)** e **C(2)** mantêm as cabeças das duas cadeias, e o resto do grupo é atribuído de **C(3)** a **C(8)**. De forma semelhante, **RE$(,)** começa agora com duas filas vazias, ficando os dados contidos entre as filas **3** e **8**.

O programa completo para a geração das cadeias ficará como se segue:

```
10 FOR N=1 TO 2: REM Categorias "1" e "2"
20 LET C(N)=0
```

```
30 NEXT N
40 FOR N=3 TO 8: REM Número de filas
50 FOR L=1 TO 2: REM Número de categorias
60 IF VAL(RE$(N,4))=L THEN GOTO 80: REM Extracção da coluna 4
70 NEXT L
80 LET C(N)=C(L)
90 LET C(L)=N
100 NEXT N
110 FOR N=1 TO 8: PRINT C(N): NEXT N
```

Como resultado, o grupo encadeado C( ) tem agora os valores seguintes:

C(1)=6
C(2)=8
C(3)=0
C(4)=3
C(5)=0
C(6)=4
C(7)=5
C(8)=7

C(1) indica que a cabeça da primeira cadeia é 6; a sequência dos ponteiros é 6→4→3→0 (o fim da primeira cadeia).

C(2) dá à cabeça da segunda cadeia o valor 8. A fila 8 aponta para 7, a 7 para 5 e a 5 para 0 (o fim da segunda cadeia). Se procedermos ao ajustamento da alteração de subscritos, anteriormente descrito, poderemos verificar que a presente sequência ordena a matriz original R$(,) conforme o código

dos produtos. Este programa constitui a base da sub-rotina 8600 de "Quem".

Para alterarmos a posição de um elo da cadeia, teremos de reposicionar três elementos de grupo.

Para ilustrarmos este facto, regressemos à cadeia de categoria "1" anteriormente criada, e em baixo reproduzida sob o título "Original". Para mudarmos a posição da segunda entrada, é necessário trocar os valores de C(4) e C(2) e depois fazer com que C(2) regresse à cadeia apontando para 6. O resultado aparece-nos na coluna sob o título "Novo":

NOVO	ORIGINAL
H=4	H=4
C(4)=1	C(4)=2
C(1)=2	C(2)=1
C(2)=6	C(1)=6
C(6)=5	C(6)=5
C(5)=3	C(5)=3
C(3)=0	C(3)=0

Quando colocamos estes mesmos grupos na ordem correcta dos respectivos subscritos, aparecem como:

NOVO	ORIGINAL
H=4	H=4
C(1)=2	C(1)=6
C(2)=6	C(2)=1
C(3)=0	C(3)=0
C(4)=1	C(4)=2
C(5)=3	C(5)=3
C(6)=5	C(6)=5

Em termos gerais, a sequência para a troca de duas filas (1 e 2) é a seguinte:

NOVO	ORIGINAL
H (ELO DE INÍCIO)	H
C(H)=1	C(H)=2
C(1)=2	C(2)=1
C(2)=F (ELO TERMINAL)	C(1)=F

A troca pode ser traduzida nas seguintes linhas de programa:

```
10 REM H é o elo de início e F o elo terminal
20 LET P=H
30 LET H=C(H)
40 LET F=C(H)
50 LET X=C(P): REM X guarda o valor enquanto se efectua a troca.
60 LET C(P)=C(H)
70 LET C(H)=C(F)
80 LET C(F)=X
90 LET C=F
```

Esta listagem faz parte da sub-rotina 8800 (extracção de bolas) do programa "Quem".

Regressando agora ao bloco principal do programa, na linha 3058, poderemos ver que se pede ao operador a especificação de dois campos nos quais se deverá efectuar a análise dos dados. A bandeira **FL** controla a ordem de entrada desses campos. Pode escolher-se qualquer par de campos, por exemplo, uma primeira triagem de produtos seguida por outra de clientes. Depois de o operador os introduzir (em ordem), os campos assumem os valores de **M1** e **M2** nas linhas 3086 e 3088. A

única restrição a este procedimento é que não podemos escolher a data como primeiro campo de extracção (linha 3066).

Passamos a seguir à extracção de todo o **R$(,)** no primeiro campo, **M1**, através da sub-rotina 8000. Porém, antes de esta ser chamada, temos de "informar" a sub-rotina 8600 de quantas filas terá de extrair, e de quantas categorias serão escolhidas para introdução. O primeiro destes parâmetros é **NT**, o total das transacções individuais lido do ficheiro dos dados de vendas, e o segundo é o número de entradas da directoria, que se vai relacionar com o primeiro campo escolhido para a extracção. Este último valor, **NR**, é retirado do ficheiro da directoria por meio da sub-rotina 5000 (linha 3094).

Já aqui descrevemos pormenorizadamente a sub-rotina 8600. Em toda a listagem do programa, os subscritos do grupo encadeado **C( )** estão feitos de modo a colocarem a matriz de dados **R$(,)** deslocada (em *offset*) **NR** lugares. Este dispositivo permite que os primeiros **NR** elementos em **C( )** armazenem as cabeças de cada cadeia (das quais existem NR), sem termos de deixar em branco um número equivalente de filas de **R$(,)**.

Também o contador **N( )**, na linha 8612, regista o número de itens de cada categoria em preparação para a extracção de bolas que se irá seguir.

Ao sair da primeira triagem, o programa avança imediatamente para a segunda, a efectuar no campo definido por **M2**. Estamos mais uma vez em presença de uma extracção de bolas, praticada dentro de cada uma das **NR** categorias estabelecidas pela primeira extracção. O processo desenrola-se na sub-rotina 8800, que é chamada **NR** vezes, em cada uma das quais se introduz diferente cabeça de cadeia como valor da variável **R**.

Depois de completada a segunda extracção, resta-nos imprimir os resultados, aplicando o grupo encadeado à matriz **RES(,)**. O processo realiza-se da linha 3118 em diante, usando o grupo encadeado para listar **R$(,)**, como ficou explicado no início do presente capítulo. A variável **A** trata do rolamento do *écran*, contando o número de linhas imprimidas, e o círculo **N** vai contando através das **NR** categorias para as quais se estão a

extrair os dados. Depois de os dados encherem um *écran* completo, o controle passa ao operador, na linha 3136. Para continuar a ver os resultados, basta premir a barra de espaços (SPACE).

APAGAMENTO DE DADOS REDUNDANTES

A opção 3) vai chamar a sub-rotina **4000**, que é muito semelhante à primeira parte da rotina de análise. Temos de especificar o período de tempo em que estão compreendidos os ficheiros a apagar, que são depois identificados e anulados definitivamente.

```
  100 REM Quem vende o que
  101 CLOSE #4
  102 LET ns=20
  104 LET nc=50
  106 LET np=20
  108 LET f1=0
  110 DIM n(20)
  114 DIM r$(100,5,9): DIM b$(50,
11): DIM c(150): DIM s$(20,10):
DIM c$(50,10): DIM p$(20,10)
  116 LET de=0
1000 REM ****Menu Principal****
1002 CLS
1004 PRINT TAB (15);"QUEM": PRIN
T TAB (14);"======="
1006 PRINT AT 9,0;"1- Entrar reg
isto de vendas": PRINT AT 10,0;"
2- Analise de Vendas": PRINT AT
11,0;"3- Anular dados redundante
```



```
s": PRINT AT 12,0;"4- Entrar nom
es em directorias": PRINT AT 13,
0;"5- Estabelecer nova base dado
s": PRINT AT 14,0;"6- Abandonar
o programa"
1008 PRINT AT 17,0;"Entre a opca
o   ";AT 18,0;: LET d=1: GO SUB
7000
1009 IF rf=1 THEN  GO TO 1017
1010 IF rf=2 THEN  GO TO 1023
1011 IF rf=3 THEN  GO TO 1016
1012 LET i$="Entre 5 se este for
 o 1/o RUN": GO SUB 9002
1014 GO TO 1008
1015 GO SUB 9502
1016 GO TO 1008
1017 IF tv=1 THEN  GO SUB 1500
1018 IF tv=2 THEN  GO SUB 3000
1019 IF tv=3 THEN  GO SUB 4000
1020 IF tv=4 THEN  GO SUB 6500
1021 IF tv=5 THEN  GO SUB 6000
1022 IF tv=6 THEN  GO SUB 6900:
GO TO 1000
1023 PRINT AT 20,1;"Entrada fora
 dos limites; tente de novo": GO
 TO 1008
1500 LET f1=0
1501 IF f1=1 THEN  GO TO 1506
1502 IF f1=2 THEN  GO TO 1508
1504 LET d$="sadir": GO TO 1514
1506 LET d$="cudir": GO TO 1514
1508 LET d$="prdir": GO TO 1514
1510 REM Entrada de item dispend
ido
1514 OPEN #4; "m";1;d$
1518 INPUT #4;nr: LET n(f1+1)=nr
1520 FOR n=1 TO nr
1522 IF f1=1 THEN  GO TO 1526
1523 IF f1=2 THEN  GO TO 1528
```

```
1524 INPUT #4;s$(n): GO TO 1530
1526 INPUT #4;c$(n): GO TO 1530
1528 INPUT #4;p$(n)
1530 NEXT n
1532 CLOSE #4
1534 IF f1=2 THEN  GO TO 1538
1536 LET f1=f1+1: GO TO 1501
1538 REM Entrada de item dispend
ido
1540 CLS : PRINT "E'a primeira e
ntrada neste mes ?(S/N)": GO SUB
 8500
1541 IF rf=1 THEN  GO TO 1552
1543 IF rf=3 THEN  GO TO 1548
1544 LET i$="Se SIM tem de criar
 novo        ficheiro- Entre S":
 GO SUB 9000
1546 GO TO 1540
1548 GO SUB 9500
1550 GO TO 1540
1552 IF tv=1 THEN  LET f1=1
1554 REM Entrada de item dispend
ido
1556 REM Entrada da data
1558 CLS : PRINT "Entre a data d
a transaccao": PRINT "==========
==================="
1560 GO SUB 8700
1562 PRINT AT 15,0;"Dia(1 a 31)"
;AT 15,15;: LET d=2: GO SUB 7000
1563 IF rf=1 THEN  GO TO 1574
1564 IF rf=2 THEN  GO TO 1576
1565 IF rf=3 THEN  GO TO 1570
1566 LET i$="Esta data sera grav
ada no        registo de vendas":
 GO SUB 9002
1568 GO TO 1562
1570 GO SUB 9502
1572 GO TO 1562
```



```
1574 PRINT AT 20,0;"==========
==============================
====="
1576 IF tv<1 OR tv>31 THEN  PRIN
T AT 20,0;"Valor fora dos limite
s - Tente de novo": GO TO 1562
1578 IF LEN (t$)=1 THEN  LET t$=
"0"+t$
1580 LET y$=t$: REM Dia
1582 IF f1=1 THEN  GO SUB 8900:
LET f1=0
1586 REM Entrada de Registos de
Vendas
1588 LET fy=0: LET a=20
1590 FOR m=1 TO a
1592 LET r$(m,1)=y$
1594 CLS : PRINT "Registo de Ven
das": PRINT "================"
1596 PRINT AT 10,0;"Vendedor num
ero     ";AT 10,17;
1598 IF fy=1 THEN  LET t$=STR$ (
VAL (r$(m,2))): LET ch=LEN (t$):
 PRINT t$;: GO SUB 7004: GO TO 1
602
1600 LET d=2: GO SUB 7000
1601 IF rf=1 THEN  GO TO 1616
1602 IF rf=2 THEN  GO TO 1614
1603 IF rf=3 THEN  GO TO 1610
1604 IF fy=1 THEN  GO TO 1616
1606 LET i$="Um numero entre 1 e
  "+STR$ (ns): GO SUB 9000
1608 GO TO 1596
1610 GO SUB 9502
1612 GO TO 1596
1614 PRINT AT 18,0;"Entrada fora
 dos limites- Tente de novo": GO
 TO 1596
1616 PRINT AT 18,0;"
```

```
                                    "
1618 IF tv>n(1) THEN  GO TO 1614
1620 PRINT AT 10,17;s$(tv): LET
r$(m,2)=t$
1622 REM r$(m,2) e'o numero do v
endedor
1624 PRINT AT 11,0;"Vendas totai
s    ";AT 11,17;
1626 IF fy=1 THEN  LET t$=STR$ (
VAL (r$(m,5))): LET ch=LEN (t$):
 PRINT t$;: LET de=1: LET d=7: G
O SUB 7004: GO TO 1630
1628 LET d=7: LET de=1: GO SUB 7
000
1629 IF rf=1 THEN  GO TO 1644
1630 IF rf=2 THEN  GO TO 1642
1631 IF rf=3 THEN  GO TO 1638
1632 IF fy=1 THEN  GO TO 1644
1634 LET i$="Montante de Vendas
          (max.7 digitos)": G
O SUB 9000
1636 GO TO 1624
1638 GO SUB 9502
1640 GO TO 1624
1642 PRINT AT 18,0;"Entrada fora
 dos limites- Repita": GO TO 162
4
1644 LET de=0: PRINT AT 18,0;"
                             "
1646 LET r$(m,5)=t$: REM Total b
ruto
1648 PRINT AT 12,0;"Cliente num.
      ";AT 12,17;
1650 IF fy=1 THEN  LET t$=STR$ (
VAL (r$(m,3))): LET ch=LEN (t$):
 PRINT t$;: LET d=2: GO SUB 7004
: GO TO 1654
1652 LET d=2: GO SUB 7000
1653 IF rf=1 THEN  GO TO 1668
```



```
1654 IF rf=2 THEN  GO TO 1666
1655 IF rf=3 THEN  GO TO 1662
1656 IF fy=1 THEN  GO TO 1668
1658 LET i$="Numero entre 1 e "+
STR$ (nc): GO SUB 9000
1660 GO TO 1648
1662 GO SUB 9502
1664 GO TO 1648
1666 PRINT AT 18,0;"Entrada fora
 dos limites- Repita": GO TO 164
8
1668 PRINT AT 18,0;"
                    "
1670 IF tv>n(2) THEN  GO TO 1666
1672 PRINT AT 12,17;c$(tv): LET
r$(m,3)=t$
1674 REM r$(m,3) e' numero de cl
iente
1676 PRINT AT 13,0;"Produto nume
ro    ";AT 13,17;
1678 IF fy=1 THEN  LET t$=STR$ (
VAL (r$(m,4))): LET ch=LEN (t$):
 LET d=2: PRINT t$;: GO SUB 7004
: GO TO 1681
1680 GO SUB 7000
1681 IF rf=1 THEN  GO TO 1696
1682 IF rf=2 THEN  GO TO 1692
1683 IF rf=3 THEN  GO TO 1690
1684 IF fy=1 THEN  GO TO 1696
1686 LET i$="Numero entre 1 e "+
STR$ (np): GO SUB 9000
1688 GO TO 1676
1690 GO SUB 9502
1692 GO TO 1676
1694 PRINT AT 18,0;"Entrada fora
 dos limites- Repita": GO TO 167
6
1696 PRINT AT 18,0;"
                    "
```

```
1698 IF tv>n(3) THEN  GO TO 1694
1700 PRINT AT 13,17;p$(tv): LET
r$(m,4)=t$
1702 REM r$(m,4) e'o numero do p
roduto
1704 IF fy=1 THEN  GO TO 1720
1706 PRINT AT 18,0;"Quer entrar
outra venda? (S/N)": GO SUB 8500
1707 IF rf=1 THEN  GO TO 1717
1708 IF rf=2 THEN  GO TO 1708
1709 IF rf=3 THEN  GO TO 1714
1710 LET i$="Se N, dados gravado
s e regresso ao Menu": GO SUB 90
00
1712 GO TO 1706
1714 GO SUB 9502
1716 GO TO 1706
1717 IF tv=1 THEN  GO TO 1720
1718 IF tv=2 THEN  LET ne=m: LET
 m=a
1720 NEXT m
1724 PRINT AT 20,0;"Quer corrigi
r as entradas? (S/N)";
1726 GO SUB 8500
1727 IF rf=1 THEN  GO TO 1738
1729 IF rf=3 THEN  GO TO 1734
1730 LET i$="Se S, nova listagem
, prima RETURN para a 1a entrada
"; GO SUB 9000
1732 GO TO 1724
1734 GO SUB 9500
1736 GO TO 1724
1738 IF tv=1 THEN  LET fy=1: LET
 a=ne: CLS : PRINT "Se OK entao
ENTER": PRINT "Se errado, volte
a entrar": GO TO 1590
2000 REM Entradas em Ficheirc
2010 INPUT #4;nr
2012 IF nr=-999999 THEN  LET nr=
```



```
0: LET nt=ne: GO TO 2028
2014 LET nt=nr+ne
2016 LET n=ne+1
2019 FOR m=1 TO 5
2020 INPUT #4;r$(n,m)
2022 NEXT m
2024 IF n=nt THEN  GO TO 2028
2026 LET n=n+1: GO TO 2018
2028 CLOSE #4
2030 ERASE "m";1;w$+z$
2032 OPEN #4; "m";1;w$+z$
2034 PRINT #4;nt
2036 LET n=1
2039 FOR m=1 TO 5
2040 PRINT #4;r$(n,m)
2042 NEXT m
2044 IF n=nt THEN  GO TO 2048
2046 LET n=n+1: GO TO 2036
2048 CLOSE #4
2050 GO TO 1002
2052 CLS : INVERSE 1: PRINT AT 2
0,0;"1/a Entrada neste mes": INV
ERSE 0: PRINT AT 0,0;
2054 GO TO 1004
3000 REM Analise de Vendas
3002 REM Entrada da Data
3004 LET f1=0
3006 CLS : IF f1=1 THEN  PRINT "
Entre a data final": GO TO 3009
3008 PRINT "Entre a data inicial
"
3009 PRINT "====================
="
3010 GO SUB 8700
3012 IF f1=1 THEN  GO TO 3016
3014 LET v$=w$: LET g$=z$: LET f
1=1: GO TO 3006
3016 LET x$=w$: LET h$=z$: LET f
1=0
```

```
3018 REM Carregamento dos Ficheiros escolhidos
3020 LET t=VAL (v$): LET d=VAL (g$)
3022 LET ne=1: LET nt=0: LET w$=v$: LET z$=g$
3024 OPEN #4; "m";1;w$+z$
3028 CLS : PRINT AT 10,5;"Dados em carregamento"
3030 INPUT #4;nr
3032 LET nt=nt+nr
3034 FOR n=1 TO nr
3036 FOR m=1 TO 5
3038 INPUT #4;r$(n,m)
3040 NEXT m
3042 LET r$(n,1)=w$+z$+r$(n,1)
3044 NEXT n
3046 CLOSE #4
3048 LET ne=nt+1
3050 LET d=d+1: IF d>12 THEN LET t=t+1: LET d=1
3052 LET w$=STR$ (t): LET z$=STR$ (d): IF LEN (z$)=1 THEN LET z$="0"+z$
3054 IF (w$=x$) AND (z$=h$) THEN GO TO 3058
3056 GO TO 3024
3058 REM Seleccao da Categoria
3060 REM Entre Categoria e numero de entradas em CAT
3062 CLS : PRINT "Seleccione a escolha primaria decategoria": IF f1=0 THEN GO TO 3066
3064 CLS : PRINT "Seleccione a escolha secundaria de categoria": GO TO 3068
3066 PRINT AT 20,0;"Nao entre 1 como campo primario"
3068 PRINT AT 3,0;"1) Data": PRI
```



```
NT "2) Vendedor": PRINT "3) Cliente": PRINT "4) Produto"
3070 PRINT AT 10,0;"Escolha de Campo?    ";AT 20,23;
3072 LET d=1: GO SUB 7000
3073 IF rf=1 THEN GO TO 3084
3074 IF rf=2 THEN GO TO 3090
3075 IF rf=3 THEN GO TO 3080
3076 LET l$="Determinacao da ordenacao dos Ficheiros": GO SUB 9000
3078 GO TO 3062
3080 GO SUB 9500
3082 GO TO 3062
3084 PRINT AT 18,0;"

   "
3086 IF f1=0 THEN LET m1=tv: LET f1=1: GO TO 3064
3088 LET m2=tv: LET f1=0: GO TO 3092
3090 PRINT "Numero ilegal - Repita": LET f1=0: GO TO 3062
3092 LET m=m1: REM 1/a Seleccao
3093 IF m=1 THEN GO TO 3090
3094 IF m=3 THEN GO TO 3098
3095 IF m=4 THEN GO TO 3100
3096 LET d$="sadir": GO SUB 5000: GO TO 3102
3098 LET d$="cudir": GO SUB 5000: GO TO 3102
3100 LET d$="prdir": GO SUB 5000
3102 CLS : PRINT AT 10,5;"Dados em seleccao para                A presentacao"
3104 GO SUB 8600
3106 REM Seleccao da Categoria secundaria
3108 LET m=m2
```

```
3110 FOR n=1 TO nr
3112 GO SUB 8800
3114 NEXT n
3116 CLS
3118 PRINT "Data    Vend.    Cli
Prod  Vendas"
3120 LET a=0
3122 FOR n=1 TO nr
3126 LET r=c(n)
3128 IF r=0 THEN  GO TO 3156
3130 LET t$=r$(r-nr,1): PRINT t$
(1 TO 6);"  ";: FOR i=2 TO 4: LE
T t$=r$(r-nr,i): PRINT t$(1 TO 5
);: NEXT i: LET t$=r$(r-nr,5): P
RINT t$(1 TO 8)
3132 PRINT : LET a=a+1
3134 IF a<10 THEN  GO TO 3150
3136 PRINT AT 20,0;"Prima SPACE
para Continuar": GO SUB 8000
3138 IF rf=1 THEN  GO TO 3148
3139 IF rf=3 THEN  GO TO 3144
3140 LET i$="O ecran mostrara' a
 continuacao da listagem": GO SU
B 9000
3142 GO TO 3136
3144 GO SUB 9500
3146 GO TO 3136
3148 LET a=0: CLS
3150 LET r=c(r)
3152 IF r=0 THEN  GO TO 3156
3154 GO TO 3128
3156 LET a=a+1: NEXT n
3158 PRINT AT 18,0;"Quer outra a
nalise? (S/N)": GO SUB 8500
3160 IF rf=1 THEN  GO TO 3170
3161 IF rf=3 THEN  GO TO 3166
3162 LET i$="Se S, pode seleccio
nar mais     campos, se N, volta
 ao Menu": GO SUB 9000
```



```
3164 GO TO 3158
3166 GO SUB 9500
3168 GO TO 3158
3170 IF tv=2 THEN  GO TO 1002
3171 IF tv=1 THEN  GO TO 3174
3172 CLS : INVERSE 1: PRINT AT 1
8,0;"Os Dados nao estao bem.  Re
veja          as entradas
     ": INVERSE 0: PRINT AT 0,0;
: GO TO 1004
3174 GO TO 3000
4000 REM Apagar dados redundante
s
4002 REM Entrada da Data
4004 LET f1=0
4006 CLS : IF f1=0 THEN  PRINT "
Escolha a data para apagar": GO
TO 4008
4007 PRINT "Escolha data a apaga
r"
4008 PRINT "=====================
=="
4010 GO SUB 8700
4012 IF f1=1 THEN  GO TO 4016
4014 LET v$=w$: LET g$=z$: LET f
1=1: GO TO 4006
4016 LET x$=w$: LET h$=z$: LET f
1=0
4018 REM Apagar os ficheiros esc
olhidos
4020 LET t=VAL (v$): LET d=VAL (
g$)
4022 LET ne=1: LET nt=0: LET w$=
v$: LET z$=g$
4024 ERASE "m";1;w$+z$
4028 LET ne=nt+1
4030 LET d=d+1: IF d>12 THEN  LE
T t=t+1: LET d=1
4032 LET w$=STR$ (t): LET z$=STR
```

```
$ (d): IF LEN (z$)=1 THEN  LET z
$="0"+z$
4034 IF (w$=x$) AND (z$=h$) THEN
  GO TO 4038
4036 GO TO 4024
4038 GO TO 1002
5000 REM Pesquisa do nome
5002 PRINT "  ";
5004 OPEN #4; "m";1;d$
5008 INPUT #4;nr
5016 CLOSE #4
5018 RETURN
6000 REM Preparar nova floppy
6002 CLS : INVERSE 1: PRINT "Qua
lquer ficheiro na fita sera'
       destruido            ":
INVERSE 0
6004 PRINT : PRINT "Presumo que
vai comecar          com novos da
dos"
6005 PRINT "QUER CONTINUAR?"
6006 GO SUB 8500
6007 IF rf=1 THEN  GO TO 6018
6008 IF rf=3 THEN  GO TO 6014
6010 LET i$="Presumo que vai com
ecar          com novos dados": G
O SUB 9000
6012 GO TO 6002
6014 GO SUB 9500
6016 GO TO 6002
6018 IF tv=1 THEN  GO TO 6022
6019 IF tv=2 THEN  GO TO 1002
6020 GO TO 6002
6022 LET fl=0
6023 IF fl=1 THEN  GO TO 6028
6024 IF fl=2 THEN  GO TO 6030
6025 IF fl=3 THEN  GO TO 6048
6026 LET d$="sadir": GO TO 6034
6028 LET d$="cudir": GO TO 6034
6030 LET d$="prdir"
6034 ERASE "m";1;d$
6036 OPEN #4; "m";1;d$
6040 PRINT #4;-999999
6042 CLOSE #4
6044 LET fl=fl+1
6046 GO TO 6023
6048 GO TO 1002
6500 REM Entrada de fichas na Di
rectoria
6502 CLS : LET fy=0
6504 LET d$="sadir": LET j$="Ven
dedor": LET nt=ns
6506 PRINT
6508 OPEN #4; "m";1;d$
6512 INPUT #4;nr
6514 IF nr=-999999 THEN  LET nr=
0: GO TO 6522
6516 FOR n=1 TO nr
6518 INPUT #4;b$(n)
6520 NEXT n
6522 CLOSE #4
6524 IF nr>0 THEN  LET fy=1
6526 CLS : PRINT "Entrada em Fic
ha": PRINT "================"
6528 PRINT "(Prima SPACE para a
entrada      seguinte)": PRINT "
(Apos terminar prima * para
 corrigir ou . para continuar)"
6530 FOR j=1 TO nt
6532 IF j=1 THEN  GO TO 6540
6534 FOR n=21-j TO 19
6536 IF n>0 THEN  PRINT AT n,0;"
                          ": PRIN
T AT n,13;b$(j-(20-n))
6538 NEXT n
6540 PRINT AT 20,0;j$;j: PRINT A
T 20,13;"          ";: PRINT AT
20,13;
```

```
6542 IF (fy=1) AND (j<nr+1) THEN
  LET t$=b$(j): GO SUB 9800: PRI
NT t$(1 TO ch);: LET t$=t$(1 TO
ch): LET d=9: GO SUB 8554: GO TO
 6547
6544 LET fy=0: LET nr=nt
6546 LET d=9: GO SUB 8550
6547 IF rf=1 THEN  GO TO 6562
6548 IF rf=2 THEN  GO TO 6560
6549 IF rf=3 THEN  GO TO 6556
6550 IF rf=4 THEN  GO TO 6570
6551 IF rf=5 THEN  GO TO 6574
6552 IF (fy=1) THEN  GO TO 6562
6553 LET 1$="Entre o nome(ate' 9
 letras)": GO SUB 9000
6554 GO TO 6540
6556 GO SUB 9500
6558 GO TO 6540
6560 PRINT AT 17,0;"Mais de 9 le
tras!": GO TO 6532
6562 PRINT AT 17,0;"
                    "
6564 LET b$(j)=t$
6566 NEXT j
6568 GO TO 6574
6570 GO SUB 6750
6572 GO TO 6532
6574 IF (fy=1) THEN  LET fy=0
6576 CLS
6578 PRINT AT 1,0;"Quer corrigir
 entradas? (S/N)";
6580 GO SUB 8500
6582 IF rf=1 THEN  GO TO 6592
6583 IF rf=3 THEN  GO TO 6588
6584 LET 1$="Se S, obtera de nov
o a listagem. Prima RETURN para
obter o 1/o   registo": GO SUB 9
000
```



```
6586 GO TO 6574
6588 GO SUB 8500
6590 GO TO 6574
6592 IF tv=2 THEN  GO TO 6606
6594 REM Rotina de Verificacao
6596 CLS
6598 PRINT "Entre o nome ate 9 l
etras": PRINT "==========
========"
6600 PRINT "Se OK entao -> RETUR
N"
6602 PRINT : PRINT "Se errado, r
eentre dados"
6604 LET fy=1: GO TO 6530
6606 ERASE "m";1;d$
6607 OPEN #4; "m";1;d$
6610 PRINT #4;j-1
6612 FOR n=1 TO j
6614 PRINT #4;b$(n)
6616 NEXT n
6618 CLOSE #4
6620 IF f1=1 THEN  GO TO 6624
6621 IF f1=2 THEN  GO TO 6625
6622 LET f1=1: LET d$="cudir": L
ET j$="Cliente n.": LET nt=nc: G
O TO 6506
6624 LET f1=2: LET d$="prdir": L
ET j$="Prod.n.": LET nt=np: GO T
O 6506
6625 GO TO 1002
6750 REM Rotina de Correcao
6752 LET l=j: IF l>19 THEN  LET
l=19
6756 PRINT AT 20-l,23;"Correc/n"
6758 PRINT AT 21-l,23;"x";CHR$ (
8);: LET d=9: GO SUB 8550
6760 IF rf=1 THEN  GO TO 6766
6761 IF rf=2 THEN  GO TO 6772
6762 IF rf=3 THEN  GO SUB 9500:
```

```
GO TO 6774
6763 PRINT AT 21-1,23;" ";
6764 LET l=l-1: IF l<=1 THEN  GO
 TO 6774
6765 GO TO 6758
6766 INVERSE 0: PRINT AT 20,1;
6768 LET b$(j-l+1)=t$: GO TO 676
4
6770 INVERSE 1
6772 PRINT AT 22,1;"Escreva os n
omes(ate 9 letras)": GO TO 6758
6774 PRINT AT 20-1,0;: RETURN
6900 REM Rotina de Escape
6902 GO SUB 9500
7000 REM Entrada de Numeros
7002 LET tv=0: LET t$="": LET ch
=0
7004 LET a$=INKEY$: IF a$<>"" TH
EN  GO TO 7004
7005 LET a$=INKEY$: IF a$="" THE
N  GO TO 7005
7006 IF CODE (a$)=13 AND ch=0 TH
EN  LET rf=0: GO TO 7044
7010 IF CODE (a$)=12 THEN  GO TO
 7032
7012 IF CODE (a$)=13 THEN  LET r
f=1: GO TO 7042
7014 IF a$="@" AND ch=0 THEN  LE
T rf=3: GO TO 7044
7016 IF ch)=d THEN  LET rf=2: GO
 TO 7044
7018 IF a$="-" AND ch=0 THEN  LE
T rf=1: GO TO 7028
7020 IF a$="." AND de=1 THEN  GO
 TO 7028
7022 IF (a$>="0") AND (a$<="9")
THEN  GO TO 7028
7026 GO TO 7004
7028 LET t$=t$+a$: PRINT a$;: LE
```



```
T ch=ch+1
7030 GO TO 7004
7032 IF ch=0 THEN  GO TO 7004
7034 LET ch=ch-1: PRINT CHR$ (8)
;" ";CHR$ (8);
7036 IF ch=0 THEN  LET t$="": GO
 TO 7004
7038 LET t$=t$(1 TO LEN (t$)-1)
7040 GO TO 7004
7042 LET tv=VAL (t$)
7044 RETURN
8000 REM Entradas por SPACE
8002 LET tv=0: LET t$="": LET ch
=0
8004 LET a$=INKEY$: IF a$=" " TH
EN  GO TO 8004
8006 LET a$=INKEY$: IF a$<>" " T
HEN  GO TO 8006
8008 LET rf=1: RETURN
8500 REM Resposta S/N
8502 LET tv=0: LET t$="": LET ch
=0
8504 LET a$=INKEY$: IF a$<>"" TH
EN  GO TO 8504
8505 LET a$=INKEY$: IF a$="" THE
N  GO TO 8505
8506 IF CODE (a$)=13 AND ch=0 TH
EN  LET rf=0: GO TO 8538
8508 IF a$="@" AND ch=0 THEN  LE
T rf=3: GO TO 8538
8510 IF CODE (a$)=12 THEN  GO TO
 8526
8512 IF CODE (a$)=13 THEN  LET r
f=1: GO TO 8536
8516 IF a$="s" AND ch=0 THEN  LE
T t$="1": LET rf=1: GO TO 8522
8518 IF a$="n" AND ch=0 THEN  LE
T t$="2": LET rf=1: GO TO 8522
8520 GO TO 8504
```

```
8522 PRINT a$;: LET ch=ch+1
8524 GO TO 8504
8526 IF ch=0 THEN  GO TO 8504
8528 LET ch=ch-1: PRINT CHR$ (8)
;" ";CHR$ (8);
8530 IF ch=0 THEN  LET t$="": GO
 TO 8504
8532 LET t$=t$(1 TO LEN (t$)-1)
8534 GO TO 8504
8536 LET tv=VAL (t$)
8538 RETURN
8550 REM Entrada de textos
8552 LET tv=0: LET t$="": LET ch
=0
8554 LET a$=INKEY$: IF a$<>"" TH
EN  GO TO 8554
8555 LET a$=INKEY$: IF a$="" THE
N  GO TO 8555
8556 IF (a$="*") AND (ch=0) THEN
  LET rf=4: GO TO 8592
8558 IF CODE (a$)=13 AND ch=0 TH
EN  LET rf=0: GO TO 8592
8560 IF (a$=",") AND (ch=0) THEN
  LET rf=5: GO TO 8592
8562 IF a$="@" AND ch=0 THEN  LE
T rf=3: GO TO 8592
8564 IF CODE (a$)=12 THEN  GO TO
 8580
8566 IF CODE (a$)=13 THEN  LET r
f=1: GO TO 8590
8570 IF ch>d THEN  LET rf=2: GO
TO 8592
8572 IF (CODE (a$)>47) AND (CODE
 (a$)<123) THEN  GO TO 8576
8574 GO TO 8554
8576 LET t$=t$+a$: PRINT a$;: LE
T ch=ch+1
8578 GO TO 8554
8580 IF ch=0 THEN  GO TO 8554
```



```
8582 LET ch=ch-1: PRINT CHR$ (8)
;" ";CHR$ (8);
8584 IF ch=0 THEN  LET t$="": GO
 TO 8554
8586 LET t$=t$(1 TO LEN (t$)-1)
8588 GO TO 8554
8592 RETURN
8600 REM Seleccao de Categoria
8602 FOR n=1 TO nr
8604 LET c(n)=0: LET n(n)=0
8606 NEXT n
8608 FOR n=nr+1 TO nt+nr
8610 LET l=1
8611 IF r$(n-nr,m,1)=" " THEN  G
O TO 8614
8612 IF (VAL (r$(n-nr,m))=1) THE
N  LET n(l)=n(l)+1: GO TO 8616
8614 LET l=l+1: IF l<=nr THEN  G
O TO 8611
8616 LET c(n)=c(l)
8618 LET c(l)=n
8620 NEXT n
8622 RETURN
8700 REM Entrada da Data
8702 PRINT AT 7,1;"Ano (aa)";AT
7,22;: LET d=2: GO SUB 7000
8703 IF rf=1 THEN  GO TO 8716
8704 IF rf=2 THEN  GO TO 8714
8705 IF rf=3 THEN  GO TO 8710
8706 LET i$="Numero entre 82 e 9
0": GO SUB 9002
8708 GO TO 8702
8710 GO SUB 9502
8712 GO TO 8702
8714 PRINT AT 20,0;"Entrada fora
 dos limites- Repita": GO TO 870
2
8716 PRINT AT 20,0;"
                    "
```

```
8718 IF (tv<82) OR (tv>90) THEN
 PRINT AT 20,0;"Numero fora dos
limites- Repita": GO TO 8702
8720 LET w$=t$: REM Ano
8722 PRINT AT 9,0;"Mes (mm)";AT
9,22;: LET d=2: GO SUB 7000
8723 IF rf=1 THEN  GO TO 8734
8724 IF rf=2 THEN  GO TO 8736
8725 IF rf=3 THEN  GO TO 8730
8726 LET i$="Numero entre 1 e 12
": GO SUB 9002
8728 GO TO 8722
8730 GO SUB 9502
8732 GO TO 8722
8734 PRINT AT 20,0;"
                    "
8736 IF (tv<1) OR (tv>12) THEN
PRINT AT 20,0;"Numero fora dos l
imites- Repita": GO TO 8722
8738 LET z$=t$: REM Mes
8740 IF LEN (z$)=1 THEN  LET z$=
"0"+z$
8742 RETURN
8800 REM Seleccao aleatoria da C
ategoria
8802 FOR s=1 TO n(n)-1
8804 LET q=0
8806 LET r=n
8808 FOR l=1 TO n(n)-s
8810 LET p=r
8812 LET r=c(r)
8814 LET j=c(r)
8816 IF j=0 THEN  GO TO 8838
8818 IF (VAL (r$(r-nr,m))<=VAL (
r$(j-nr,m))) THEN  GO TO 8832
8820 LET q=1
8822 LET a=c(p)
8824 LET c(p)=c(r)
8826 LET c(r)=c(j)
```



```
8828 LET c(j)=a
8830 LET r=j
8832 NEXT l
8834 IF q=0 THEN  GO TO 8838
8836 NEXT s
8838 LET a=0: RETURN
8900 REM Preparacao do Ficheiro
                Principal
8902 CLS
8904 ERASE "m";1;w$+z$
8908 OPEN #4; "m";1;w$+z$
8912 PRINT #4;-999999
8914 CLOSE #4
8916 RETURN
9000 REM Rotina de Auxilio
9002 PRINT AT 18,0;"

    "
9004 PRINT AT 18,0;i$: PRINT "Pr
ima SPACE para continuar": GO SU
B 8000
9006 IF rf=1 THEN  GO TO 9014
9007 IF rf=3 THEN  GO TO 9010
9008 GO TO 9002
9010 GO SUB 9502
9012 GO TO 9002
9014 PRINT AT 18,0;"



                                  "
9016 RETURN
9500 REM Rotina de Escape
9502 CONTINUE : PRINT AT 20,0;"Q
uer parar? (S/N)": GO SUB 8500
9504 IF rf=2 OR rf=3 THEN  GO TO
 9502
9506 IF tv=2 THEN  GO TO 9510
9508 CLS : PRINT AT 10,15;"FIM";
AT 11,13;"=======": STOP
```

```
9510 CLS : RETURN
9800 REM Comprimento de t$ em ch
9802 LET ch=LEN (t$)
9804 IF t$(1)=" " THEN  LET ch=0
: RETURN
9806 IF t$(ch)=" " THEN  LET ch=
ch-1: GO TO 9806
9808 RETURN
```


Capítulo 9

Tendência de vendas O utilitário do director de vendas

INTRODUÇÃO

Os programas descritos nos capítulos anteriores referiam-se todos a aspectos específicos de direcção ou gestão de vendas. Contudo, na utilização do dia-a-dia, seria muito melhor dispormos de um único programa que reunisse todas as características daqueles, e que no fundo cobrisse todas as necessidades do director do departamento de vendas.

A organização em pormenor de tal programa depende em larga medida do papel desempenhado pelo gestor, dos objectivos do seu trabalho e de quais os aspectos mais importantes da tarefa a cumprir. Este último capítulo contém o esboço de como se poderá construir esse programa, a partir dos elementos discutidos ao longo da obra.

O resultado é uma base de dados inteligível, que proporciona a base de um sistema de fiscalização do cumprimento dos objectivos relacionados com o mundo das vendas. Servimo-nos aqui das superiores capacidades de processamento de dados inerentes aos computadores, para criar e manter um registo permanente não só das *performances* globais como das reveladas pelas várias facetas de negócio. O programa permite ainda a realização de análise de dados, factor de grande importância para o director de vendas, pois é determinante na exploração das causas de quaisquer variações de resultados da empresa.

Sem a sólida base de apoio proporcionada por um bom programa de base de dados, é hoje quase impossível dominar o enorme volume de explicações racionais e argumentos necessários ao suporte das tarefas de gestão, e ao estabelecimento de elementos demonstrativos que apoiem e permitam apresentar

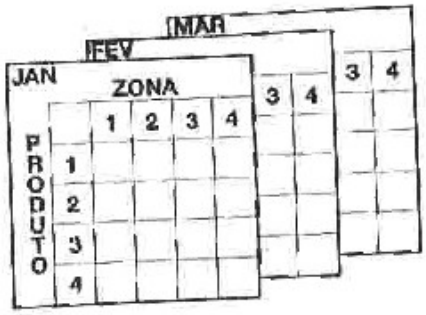
relatórios "em cima do acontecimento". Para além do mais, qualquer análise de dados actualizada deve estar sempre à mão quando se trata de verificar a eficiência dos subordinados, pois sem esse suporte não haverá limites para o número e ingenuidade das justificações que poderão ser apresentadas em face de maus (ou bons) resultados.

Um sistema com estas características e possibilidades está fora do alcance de qualquer método de processamento manual. Da mesma maneira, não é normalmente efectivo o acesso imediato a informações actualizadas num sistema baseado em computadores de grandes dimensões. Em conclusão, as bases de dados são uma das aplicações ideais para os computadores pessoais.

DESCRIÇÃO GERAL

A base de dados é mantida em ficheiro sob a forma de uma série de matrizes, uma para cada período, que no exemplo abaixo contêm os dados da aceitação de cada produto nas várias zonas de actuação. As matrizes poderiam muito bem ser estabelecidas para outros valores, como os das encomendas feitas, das consultas recebidas ou quaisquer outros dados estatísticos entretanto coligidos.

O esquema básico é o seguinte:



Cada coluna contém as vendas por produto numa única zona.

Cada fila, por sua vez, reflecte as vendas por zona de um único produto:

Para examinar as vendas de um determinado produto/zona ao longo de dado período de tempo, o programa percorre os registos mensais de forma a criar um novo ficheiro, que ficará constituído por uma sequência de estatísticas de vendas retiradas da mesma célula de cada registo mensal.

Depois de coligir todos os dados básicos, o programa tem de os apresentar de forma tal que se possa examinar a variação de tendência de cada uma das células individuais produto/zona. Esta disposição fornece-nos poderosos meios para analisarmos as causas dos diferentes resultados alcançados, e a correlação

entre as aparentes variações dos diversos sectores da actividade. A metodologia seguida pode compreender-se melhor através dos exemplos que se seguem:

Suponhamos que a aceitação de um dado produto constitui motivo de preocupação. O primeiro passo seria verificarmos como o total das vendas se reparte pelas várias zonas; a conclusão de que a aceitação é fraca só em duas zonas poderá significar que as causas deste resultado excepcional se relacionam mais com essas duas zonas do que com as características do produto em si. Para confirmarmos este pressuposto, será útil irmos verificar como é que essas duas zonas fora do vulgar se comportaram perante outros produtos, e porventura compararmos o total das vendas aí verificado com os totais de outras zonas.

Problema mais difícil de resolver é o fenómeno da sazonalidade, que permite respostas pouco claras, do tipo "nunca vendemos muito em Janeiro, mas recuperaremos a partir de Abril". Quer esta conclusão seja razoável ou não, poderemos confirmá-la se examinarmos as tendências anteriores. Se existir uma variação sazonal, terá esta exercido influência sobre outros produtos nesta ou noutras zonas? Se só algumas zonas se revelarem sazonais, poderemos associar o facto com os produtos, com as zonas ou com ambos? E muito mais poderá surgir...

Uma base de dados concebida propositadamente para a actividade de *marketing* constitui instrumento poderoso na previsão de vendas, pois a sua estrutura é o reflexo preciso dos mecanismos que geram a efectivação das transacções. Os métodos de previsão consistem, efectivamente, na divisão da actividade nos seus vários componentes lógicos, projectando-os depois em direcção ao futuro. A previsão final será a soma das previsões dos elementos constituintes da actividade.

OS ELEMENTOS DO PROGRAMA

Torna-se evidente que os elementos principais de tal programa já foram descritos nos capítulos anteriores. O "Ajustasdor" fornece os meios para compensarmos os efeitos da inflação e do

número desigual de dias úteis em cada mês. O programa "Gráficos" resolve a apresentação das vendas, ao traçar os respectivos valores. O "Previsor" contém o modelo de normalização exponencial necessário para a projecção no futuro das variações de tendência, e por fim o programa "Quem" proporciona um método de criação de ficheiros e de triagem de períodos seleccionados, para análise subsequente. Falta descobrir o modo de introduzir todos estes programas dentro dos limites do computador.

Antes de prosseguirmos, convém, porém, mencionar uma particularidade importante da programação não referida nos capítulos precedentes, e que se refere à apresentação dos dados sob a forma de tabelas. Uma matriz de 15×15 é demasiado grande para caber no *écran*, de modo que só podemos apresentar parte dela de cada vez. A solução ideal seria a apresentação em "folha de cálculo" *(spreadsheet)*, que consegue o rolamento da imagem tanto na vertical como na horizontal; porém, a programação deste processo em BASIC torna-o tão lento que deixa de ter aplicação prática. Menos ambiciosa é a aceitação das limitadas capacidades de rolamento vertical de imagem do *Spectrum*, e procurarmos obter um pouco do efeito de "folha de cálculo" dividindo o plano horizontal em secções. De acordo com este esquema, a matriz 15×15 é dividida em matrizes mais pequenas (por exemplo, 15×3), que deste modo podem ser totalmente apresentadas em *écran*. Para além da matriz 15×3, temos ainda de acomodar na imagem os totais das colunas e filas, dados essenciais para a compreensão da tabela em apreciação.

Na primeira apresentação da imagem, surge-nos a matriz número **1** e os respectivos totais. Para passar à número **2**, o operador prime a tecla da seta direita, provocando a impressão da matriz em questão. Nova pressão na mesma tecla faz com que apareça a matriz **3**. Para voltar a qualquer das matrizes anteriores, basta premir a tecla da seta esquerda, tantas vezes quantas as necessárias. Os processos de apresentação das tabelas são controlados por dois contadores, um para a tecla da seta direita

e outro para a da seta esquerda, e que têm a função de contabilizar o número de pressões exercidas sobre essas teclas. Os contadores só podem assumir os valores **0**, **1** ou **2**. Se premirmos a tecla da seta esquerda durante a apresentação da matriz **1** (isto é, com o respectivo contador em zero), o contador permanece em **0**, voltando a aparecer na imagem a matriz **1**. O mesmo se passa no outro extremo da apresentação: com o contador da seta direita em **2**, se premirmos de novo essa tecla provocamos novo aparecimento da matriz número **2**, ficando o contador com o mesmo valor.

O programa contém outros comandos de tecla única, para passarmos a dados de outros meses e para sairmos da apresentação dos dados em direcção à parte seguinte do processamento.

ESTRUTURA DO PROGRAMA

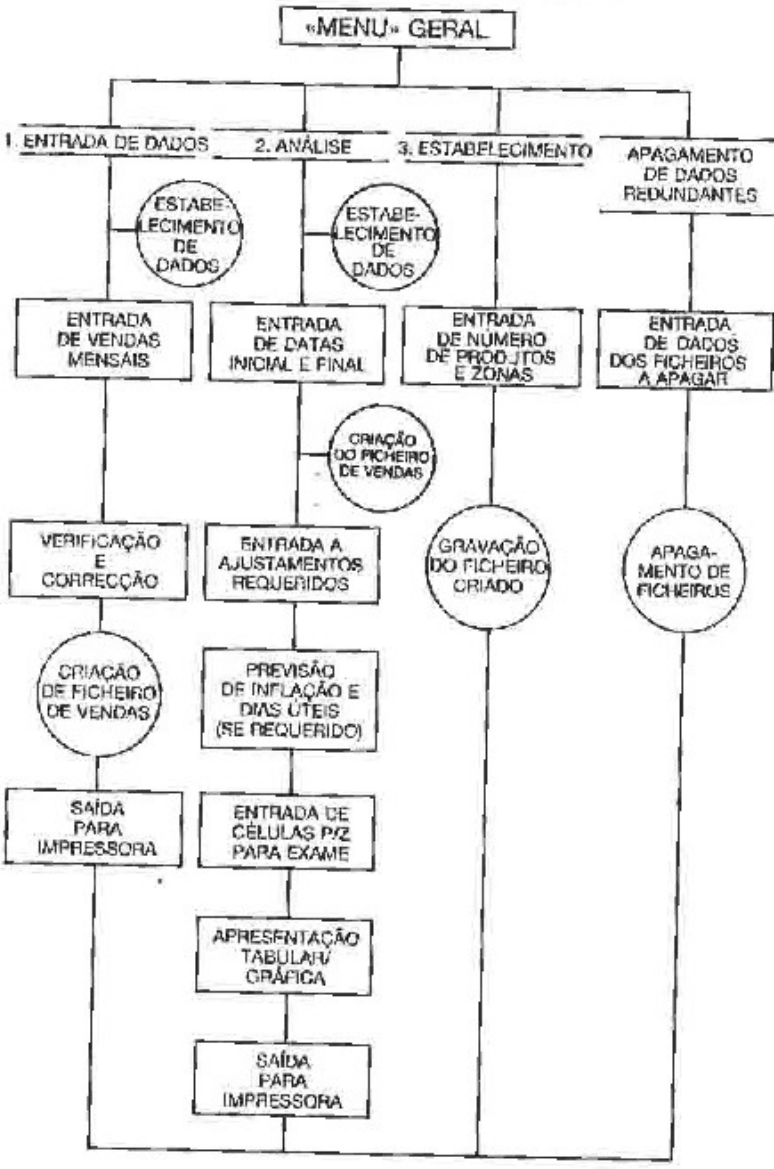
Há muitas maneiras de arrumar a estrutura pormenorizada de uma base de dados de *marketing*, de modo a correspondermos às características específicas do tipo de negócio e às necessidades do director de departamento ou do chefe de secção. Existem porém, além da configuração pretendida, determinados problemas de programação e de concepção da apresentação comum a qualquer solução adoptada.

O programa divide-se naturalmente em duas secções principais: entrada de dados e análise de resultados. A primeira é relativamente simples: lê os valores das vendas de cada mês, posiciona-os na matriz, calcula os totais e cria um ficheiro de dados. A segunda é mais complexa; tem de extrair os diversos valores dos ficheiros, analisá-los, proceder ao ajustamento para a inflação, efectuar a previsão e apresentar os resultados, sob a forma de gráfico ou em tabela, conforme o pretendido.

Para a primeira secção a matriz 15×15 produto/zona deverá revelar-se suficiente. Na secção de análise, aceitaremos o compromisso entre o volume de dados a processar e os vários tipos de análise possíveis.

O fluxograma seguinte explica a estrutura do programa:

TENDÊNCIAS — FLUXOGRAMA SIMPLIFICADO



O *menu* geral oferece as seguintes opções:

— Entrada de dados (registo dos dados de vendas mensais).
— Análise de dados (que produz a apresentação da parte seleccionada da base de dados).
— Estabelecimento (para quando se cria pela primeira vez o ficheiro destinado ao registo dos parâmetros principais da base de dados).
— Apagamento (que serve para eliminar fichas de dados desnecessários).

ESTABELECIMENTO

O estabelecimento do programa mantém um registo das dimensões da matriz de dados, ou seja, do número de produtos e de zonas. É essencial que cada conjunto de dados mensais contenha o mesmo número de informações.

ENTRADA DE DADOS

O programa de entrada de dados tem de ser concebido de forma a que estes possam ser introduzidos com relativa facilidade, com apresentação clara e dispondo de possibilidade de correcção de erros, quer quando são cometidos quer durante posteriores verificações de validade. Uma das soluções consiste em entrar uma só coluna de dados de cada vez — por exemplo, todas as vendas de produtos na zona 1. Pode-se usar aqui uma rotina semelhante à aplicada no programa "Gráficos". Depois de completada a entrada de dados, o programa tem de calcular os totais e apresentar a tabela inteira, para posterior gravação se tudo estiver correcto. É útil dispormos da possibilidade de imprimir em papel os valores em jogo, para que o operador possa manter um registo físico permanente (arquivo).

Qualquer programa necessário à inicialização da impressora será sempre longo, tendo portanto de ser inserido como sub-rotina ou mesmo como programa separado dependente de opção do *menu*; temos também de o equipar com qualquer tipo de "rolamento", pois o mais natural é as 15 colunas de dados não

caberem na impressora. Se a rotina de impressão tiver de suportar diferentes formatos, e até estar preparada para várias marcas de impressora, será necessário introduzir-lhe constantes de impressão — número de linhas, folha simples ou papel contínuo, avanço de linha, etc., logo no início do programa.

ANÁLISE DE DADOS

Para iniciar a análise de dados, o operador especifica qual a parte do ficheiro que quer trabalhar, através dos seguintes parâmetros:

— Datas inicial e final.
— Identificação dos produtos e zonas a examinar.
— Ajustamento ou não para a inflação e dias de trabalho.
— Necessidade ou não de qualquer tipo de previsão.

As entradas têm de ser cuidadosamente verificadas nesta fase, para que não conduzam à chamada de ficheiros não existentes ou ao extravasamento da capacidade de memória disponível. Como a utilização da memória é determinada por um certo número de parâmetros definidos pelo utilizador, a formulação destas constantes tem de ser o mais criteriosa possível.

O programa lê seguidamente os ficheiros escolhidos, e extrai os dados relacionados com os produtos e zonas seleccionados de cada ficheiro, colocando-os em memória, sob a forma de matriz produto/zona para o período de tempo em causa. Se necessário, os valores são ajustados para a inflação e/ou dias de trabalho, e também projectados no futuro por meio de modelo de normalização exponencial semelhante ao descrito em "Ajustador".

O arranjo preciso da apresentação de resultados dependerá naturalmente da finalidade específica do programa, e será sempre um compromisso entre vários objectivos de igual importância. Se pretendermos que a primeira apresentação seja um gráfico, não nos devemos esquecer da utilidade de dispormos de qualquer modo de referência rápida aos dados numéricos originais, mediante a sua apresentação em imagem secundária

— na verdade, ninguém pode acreditar num gráfico sem poder ver os valores utilizados para o construir.

O próprio traçado do gráfico deve merecer certa consideração. Se todas as variáveis produto/zona forem "atiradas" ao mesmo tempo para dentro do *écran*, será impossível distingui-las umas das outras; se forem colocadas em coluna infindável, passar-se-á problema semelhante, pois o mais certo é excederem os limites da imagem. Uma solução possível consiste em desenhar cada gráfico e depois apagá-lo "redesenhando-o" com a cor alterada para preto.

Por fim, para conseguirmos comparar variações de tendência entre diferentes produtos e zonas de venda, teremos de poder seleccionar os vários gráficos, e especificar a ordem por que devem aparecer no *écran*.

Todos estes requisitos múltiplos e conflituosos provocam verdadeiros problemas de programação. As diversas imagens apresentadas têm de ser organizadas de modo a que o operador possa passar de umas para outras através de comandos simples — de preferência entradas de uma só tecla —, sem cair em situações tais como saltos inopinados de imagem ou desconexão entre os dados acabados de processar e a respectiva apresentação.

O esquema básico mais aconselhável consiste em colocar o operador perante as várias opções dispostas em *menu*, e depois colocar todo o programa de apresentação sob a forma de um círculo *(loop)*. O operador começa por escolher uma das várias alternativas colocadas à sua disposição, e seguidamente trabalha a apresentação especificada, após o que regressa ao *menu* inicial. Não nos podemos esquecer de programar a saída da rotina de apresentação e concomitante retorno ao programa principal, para que novos elementos possam ser extraídos da base de dados.

Tal como a fase de entrada de dados, é deveras aconselhável, senão indispensável, podermos obter cópias em papel das análises efectuadas pela segunda fase do programa, que irão constituir a base do *dossier* das decisões do gestor de vendas ou

de *marketing*. A saída de texto pode obter-se imprimindo directamente a matriz de dados, por meio de rotina semelhante à indicada para o bloco de títulos da entrada de dados. Para a impressão de gráficos, dispomos de três opções: fotografar o *écran* (!), utilizar uma impressora de matriz de pontos *(dot matrix)*, ou uma impressora-traçadora (do tipo X-Y). Em essência, a impressora de matriz executa verdadeira cópia da imagem do monitor ou televisor, reproduzindo o resultado de uma "varredura" do *écran* feita *pixel* a *pixel*; a impressora produz um ponto ou deixa um espaço em branco conforme o que estiver presente na imagem. As traçadoras gráficas são programadas de forma semelhante à da execução dos próprios gráficos, definindo as coordenadas dos pontos que uma caneta ou estilete acabará por unir.

EM CONCLUSÃO

No capítulo de abertura — «Computadores, amigos ou inimigos» —, chamámos a atenção para a posição desvantajosa em que se poderão encontrar os directores de vendas caso não disponham de acesso imediato a informações actualizadas.

Nos capítulos subsequentes, tentámos apresentar programas que permitam coligir dados importantes e expô-los correctamente dentro do contexto da gestão comercial.

Este último capítulo pretendeu ser mera introdução a um tipo mais avançado de programação, servindo-se dos elementos antecedentes da obra.

A tarefa de construir "Tendência de vendas" a partir dos programas descritos representa a forma como um director de vendas pode competir com os colegas dos departamentos de contabilidade e de produção, quando se trata de explicar como se comporta o complexo "mundo dos negócios".

Se, como é possível, o leitor tiver ideias para a criação de outros programas que não os referidos neste livro, será bom que aprenda um pouco sobre a concepção e programação da máquina em si, neste caso o computador *Spectrum*. Esta evolução implica a abordagem da tecnologia de computadores, podendo começar pelo estudo do manual do que se está a usar, passando depois para obras especializadas em código máquina e em *hardware*.

Como é sempre mais fácil estudarmos assuntos complexos possuindo uma sólida base de partida, esperamos que os programas descritos nesta obra sirvam ao leitor para iniciar a sua própria actividade de programação, e como aliciante para a sua profissão.

Melhor dizendo, esperamos que dentro de pouco tempo o leitor possa comparecer na reunião mensal com o administrador e comece por dizer "O meu computador afirma que...".

3.º volume

O Domínio do Código Máquina do Spectrum
de *Toni Baker*

Este livro satisfaz a ambição – e a necessidade – de todos os programadores que utilizam o *Spectrum*: dominar directamente a linguagem que o coração da máquina conhece, o que representa um salto qualitativo quanto a versatilidade de programação, velocidade de execução e domínio do grafismo.

4.º volume

As Melhores Rotinas para o ZX Spectrum
de *John Hardman e Andrew Hewson*

40 rotinas em código máquina para o *Spectrum*. Obra destinada a quem quer tirar o melhor partido do seu minicomputador: utilizando a linguagem máquina do *Spectrum*, multiplicam-se espantosamente as suas capacidades.
Além disso, trata-se de um instrumento de trabalho de inexcedível utilidade.

5.º volume

Os 20 Melhores Pogramas para o Spectrum
de *Andrew Hewson*

Estes programas, além da sua utilidade prática, representam uma perfeita fonte de referência das técnicas de programação mais difundidas e ensinam a arte de programar através de exemplos perfeitos.

6.º volume

Guia Avançado para o Spectrum
de *Mike James*

Introdução prática às características mais avançadas do *Spectrum*, tanto no *hardware* como no *software*.
Oferece ao leitor a exploração das possibilidades mais sofisticadas deste microcomputador.

7.º volume

57 Rotinas em Basic para o Spectrum

Autêntico instrumento de trabalho para quem se dedica a criar os seus próprios programas e ambiciona libertar-se da resolução dos múltiplos problemas práticos rotineiros que tornam fastidiosa aquela fascinante actividade.

8.º volume

Astronomia no ZX Spectrum

Obra dinâmica e cheia de interesse, que introduz o leitor com simplicidade nos vários campos da astronomia mas se dirige especialmente a quem possui o *Spectrum* e deseja expandir os seus conhecimentos informáticos a outros campos.

9.º volume

Introdução ao Pascal

O Pascal, útil linguagem de computadores concebida para promover um modo lógico e simples de programar, está

actualmente disponível na maioria dos microcomputadores pessoais — por exemplo, o *ZX Spectrum*. Este estudo mostra as vantagens do emprego do Pascal e abre grandes perspectivas aos amadores de informática.

10.º volume

O Spectrum por dentro

O interesse pelo *Spectrum* em si, como máquina – isto é, o *hardware* –, impôs-se naturalmente a inúmeros entusiastas da microinformática, desejosos, por curiosidade e por necessidade, de conhecerem efectivamente o funcionamento e o *design* do mais popular dos computadores.
Esta obra permite aos iniciados curiosos manusear projectos até agora «secretos», com perfeita compreensão do que acontece dentro do computador.