# MICROCOMPUTADOR

## CURSO BÁSICO



EDIÇÃO REVISTA E ATUALIZADA

EDITORA GLOBO

# MICRO COMPUTADOR

## CURSO BASICO

Todos os direitos reservados. Nenhuma parte desta edição pode ser utilizada ou reproduzida — em qualquer meio ou forma, seja mecânico ou eletrônico, fotocópia, gravação, etc. — nem apropriada ou estocada em sistema de banco de dados, sem a expressa autorização da editora.

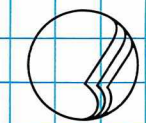
**Tradutoras:**
Maria Clara Cescato
Regina Maria M. Gonçalves
**Consultor técnico:**
Jakow Grajew
**Adaptação dos programas BASIC:**
SAD — Sistema de Apoio e Decisão
**Fotos de equipamentos nacionais:**
Salvador de Rosa
**Composição:**
Lídio Ferreira Jr. Artes Gráficas e Editora Ltda.
**Impressão:**

Cayfosa - Sta. Perpètua de la Mogoda. Barcelona
Depósito legal: B.525-1987 (I)

© Orbis Publishing Co., 1984
© Editora Globo, 1987, para a língua portuguesa.

# MICRO COMPUTADOR

## CURSO BASICO



# Construa seus jogos

## O Programa de Criação de Fliperama — um avanço notável em desenho de software — permite que você projete e brinque com seus jogos na tela de um computador tipo Apple.

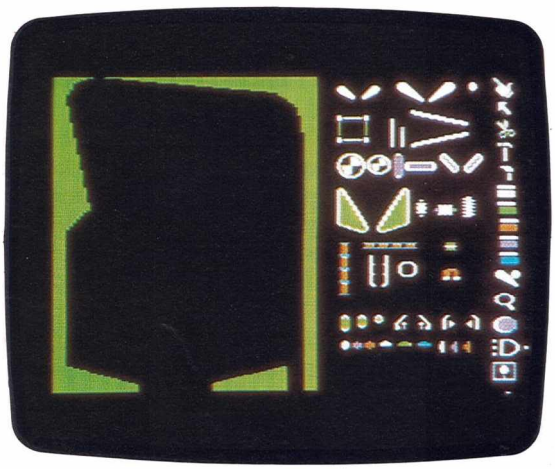
Até mesmo na indústria dos microcomputadores, que se desenvolve com rapidez e onde se pode esperar inovações notáveis, é difícil encontrar um produto radicalmente diferente na originalidade da concepção e na qualidade. Tal produto existe — é o software do Programa de Criação de Fliperama (Pinball Construction Set, PCS), da Budgeco. Operado num computador tipo Apple II com 48 Kbytes, com uma unidade de disco e um joystick, esse equipamento realiza uma função aparentemente simples. Apresenta ao usuário a figura de uma mesa de fliperama vazia, e um "menu" de 38 tipos de "mobília" que poderão ser usados para completar o desenho do jogador. Tem, ainda, um "menu" de funções, no qual o jogador escolhe os instrumentos que deseja utilizar.

Depois de preencher a mesa de acordo com seu plano — você pode posicionar 128 peças na mesa e usar qualquer uma quantas vezes quiser —, tudo o que você tem a fazer é começar o jogo. Mas, para isso, selecione outra função com o joystick. Até quatro jogadores podem tomar parte no jogo, e a cada um cabe apenas uma bola, em vez das três geralmente presentes em jogos eletrônicos; além disso, não existe a facilidade da "bola livre". No final do jogo, basta pressionar a tecla ESCAPE, e você tem outra vez o menu de escolha. A experiência adquirida em cada jogo estimula-o a planejar coisas diferentes.

Tanto em concepção como em execução, o PCS configura-se como um software realmente "amigo do usuário" (user-friendly). Assim que se carrega o programa (e para isso basta colocar o disco e pressionar a tecla RETURN), toda a ação é controlada pelo joystick. O primeiro instrumento a ser utilizado consiste numa pequena mão. Ela é movida para apontar um objeto no menu da "mobília" (por exemplo, um obstáculo ou um "flipper"), e quando você pressiona o botão do joystick a mão "apanha" o objeto indicado e o leva para a posição desejada na mesa. No momento em que você solta o botão do joystick, o objeto é colocado no local escolhido.

O interessante aqui é que, quando você começa a brincar, não move apenas a coleção de dados que define a forma do objeto, mas também o conjunto de regras que comandará o andamento do jogo. Um determinado flipper, por exemplo, costuma se mover dentro de um ângulo de 45 graus, primeiro para cima, depois para baixo. Um obstáculo sempre repele a bola ao mesmo tempo em que a impulsiona, conforme a força aplicada. A bola obedece às leis do movimento de Newton e corre na mesa de acordo com a lei da gravidade.

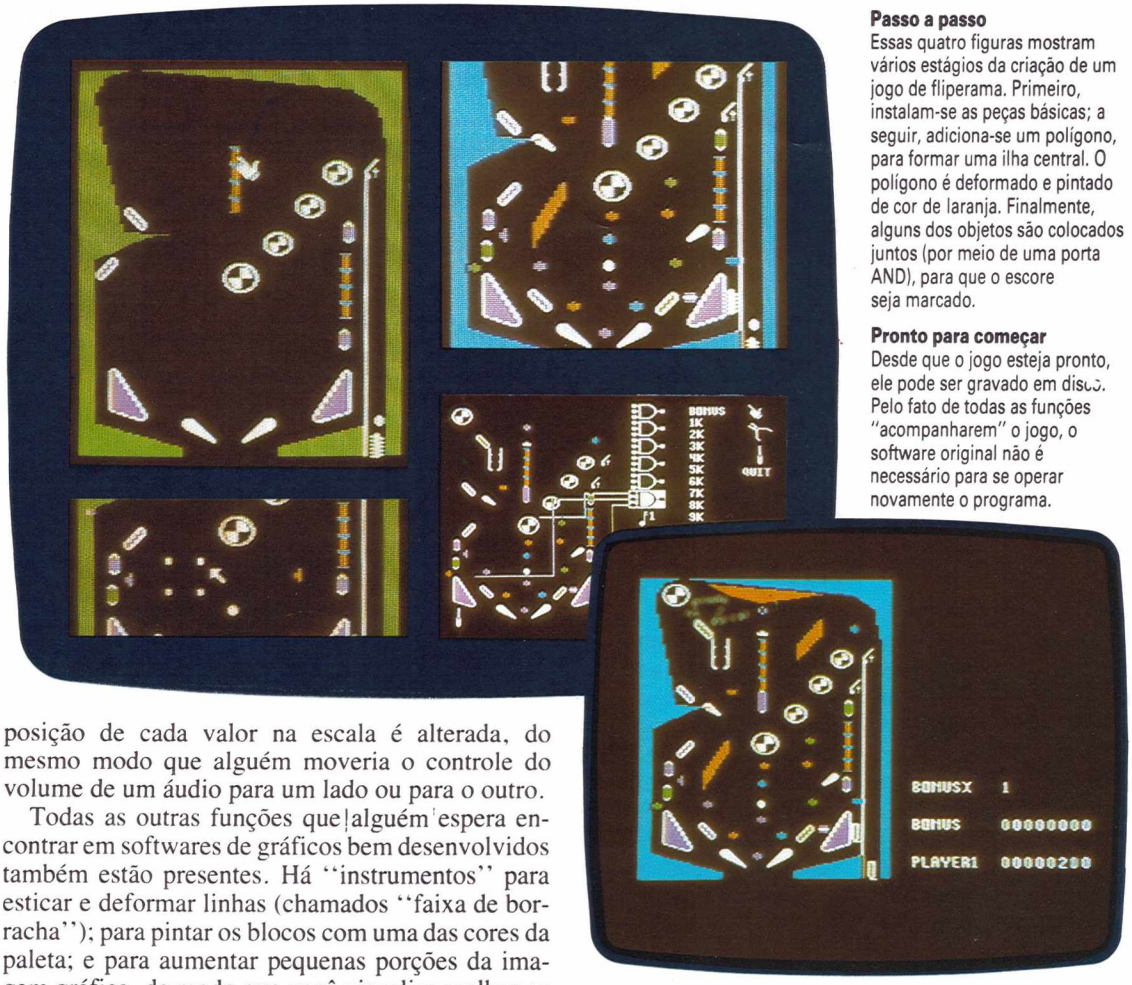
Entretanto, há um instrumento (adequadamente simbolizado por um planeta com luz solar parcial) que permite modificar os parâmetros do mundo real — força da gravidade, por exemplo, e até mesmo o tempo. O joystick controla também esta função. A



**Jogos de faça-você-mesmo**
O Programa de Criação de Fliperama mostra uma mesa vazia, uma variedade de tipos de "mobília" — obstáculos, alvos, flippers etc.; e, na coluna da direita, os instrumentos para a colocação dos objetos na mesa. Essa coluna contém, ainda, funções para ajuste de tamanho, forma, cor e grau de interação das peças, assim como funções para gravar em disco jogos já realizados.




**Coisa de criança**
O PCS lhe oferece até mesmo sons autênticos e o equivalente a flashes de luz. Mas é ainda mais divertido montar jogos do que propriamente brincar com eles.




**Passo a passo**
Essas quatro figuras mostram vários estágios da criação de um jogo de fliperama. Primeiro, instalam-se as peças básicas; a seguir, adiciona-se um polígono, para formar uma ilha central. O polígono é deformado e pintado de cor de laranja. Finalmente, alguns dos objetos são colocados juntos (por meio de uma porta AND), para que o escore seja marcado.

**Pronto para começar**
Desde que o jogo esteja pronto, ele pode ser gravado em disco. Pelo fato de todas as funções "acompanharem" o jogo, o software original não é necessário para se operar novamente o programa.

posição de cada valor na escala é alterada, do mesmo modo que alguém moveria o controle do volume de um áudio para um lado ou para o outro.

Todas as outras funções que alguém espera encontrar em softwares de gráficos bem desenvolvidos também estão presentes. Há "instrumentos" para esticar e deformar linhas (chamados "faixa de borracha"); para pintar os blocos com uma das cores da paleta; e para aumentar pequenas porções da imagem gráfica, de modo que você visualize melhor os detalhes.

Não só as funções e os recursos do Programa de Criação de Fliperama têm importância, mas sobretudo sua filosofia geral de operação. A programação de objetos orientados — onde cada elemento de operação do pacote de software traz consigo detalhes de como ele funcionará e como vai interagir com qualquer outro objeto ou elemento — possibilita a produção de programas que requerem pouca experiência em computação por parte do usuário. Este será o método de programação quase que exclusivamente utilizado na quinta geração de computadores, já em desenvolvimento. A programação de objetos orientados é considerada o maior acontecimento no campo da ciência do software, desde que surgiram as linguagens de alto nível no final dos anos 50.

A maioria dos microcomputadores tem capacidade de memória e poder de processamento suficientes para as necessidades do usuário. Qualquer aumento na capacidade e no poder do micro possibilitará a utilização de softwares mais complexos. O que existe de realmente surpreendente no PCS é sua capacidade de atingir um alto nível de eficiência utilizando apenas 48 Kbytes.

Embora a programação de objetos orientados se aplique diretamente a jogos e outros programas de gráficos, é preciso haver uma certa engenhosidade na programação, para que esse tipo de software se torne viável comercialmente. Apesar de não utilizarem gráficos como principal meio de comunicação, os pacotes de folhas eletrônicas (como o Visicalc e o Supercalc) possuem objetos até certo ponto orientados, pois cada campo ou célula contém um dado e a relação que o define.

Outro exemplo é o sistema Lisa, da Apple. Este emprega um mouse para manobrar um indicador na tela que seleciona o programa desejado, representado por um símbolo gráfico. O processador de palavras, por exemplo, tem como representação uma folha de papel datilografada e o programa traçador de gráficos, uma folha de papel quadriculado.

Talvez a mais fascinante de todas essas funções seja o método Lisa de transferir dados de um programa para outro. Um de seus "Icons" (nome dado às representações visuais da tela) é uma espécie de "prancheta". Se quisermos tomar uma pequena porção da folha eletrônica e reproduzi-la graficamente, será necessário apenas definir a "janela" (espaço) na folha, transferindo-a para a "prancheta" (que funciona como área de armazenagem temporária). A reprodução será feita em seguida pelo programa traçador de gráficos.

Quando falamos sobre jogos eletrônicos (ver p. 221), notamos que vários tipos diferentes podem ser identificados. O PCS poderia ser visto como uma nova categoria. É possível prever que a próxima etapa da indústria de jogos para computadores domésticos seja a produção de Programas de Criação de Labirintos e Caçadores, Programas de Criação de Invasores Espaciais, e assim por diante; e neste momento alguns escritores de programas de jogos poderiam se tornar dispensáveis.



**Visão objetiva**
Além de um jogo intrigante e educativo, o Programa de Criação de Fliperama é um bom exemplo de programação de objeto orientado. Em programação normal, a estrutura dos dados se define, e as rotinas do programa são escritas para manipulá-los. Em programação de objeto orientado, os cálculos e os procedimentos são inseparáveis dos dados. Nesse programa de fliperama, o fato de mover os flippers no quadro não só estabelece os dados (neste caso, a forma do flipper), mas também os organiza.

A programação de objetos orientados possibilita muitas aplicações visuais. As folhas eletrônicas constituem outro exemplo: o campo que mostra um resultado também terá a fórmula para se chegar a ele.

A tendência moderna de os computadores comerciais se tornarem "estações de trabalho", simulando os objetos que se encontram sobre uma mesa de escritório, também deriva da mesma idéia. Apontando-se para a imagem de uma folha de papel datilografada na tela, ativa-se o processador de palavras, e ao se apontar para o desenho em miniatura de um arquivo os resultados são automaticamente arquivados.

# Controle seu percurso

**Os mísseis Cruise são um tema controverso, mas possuem uma interessante tecnologia de computação — a memória de bolha — que poderá fazer parte de microcomputadores.**

O primeiro passo de Neil Armstrong sobre a superfície da Lua foi possível, em grande parte, em decorrência dos sistemas de orientação computadorizados. Evidentemente, a engenharia de foguetes interplanetários apóia-se em uma tecnologia muito precisa, mas, sem o hardware e o software de computadores, jamais seria possível executar cálculos de posição com rapidez e exatidão suficientes para permitir o acoplamento de dois objetos a uma grande distância — mesmo que um desses objetos tenha o tamanho da Lua.

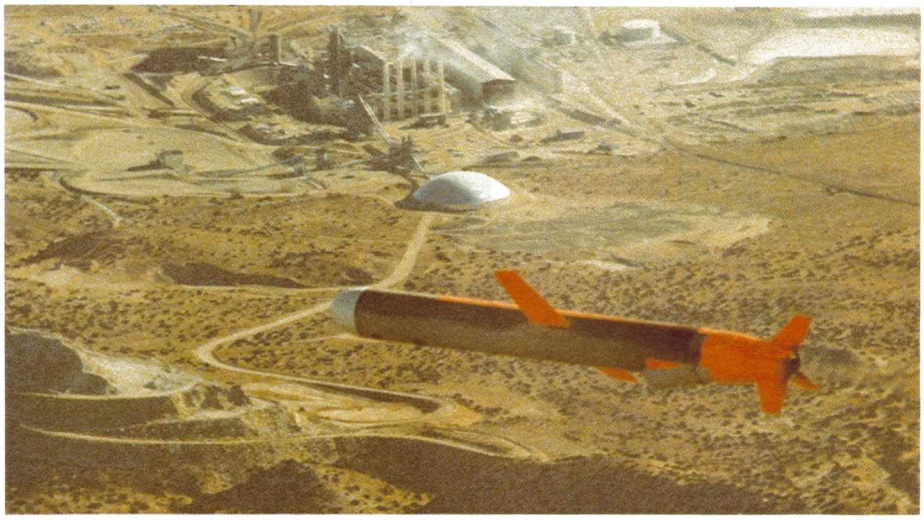
Quando se levam em conta as técnicas militares modernas, que exigem a colocação de ogivas com limite de erro de 20 ou 30 m após o vôo transcontinental, é enorme a capacidade de processamento de dados necessária para os cálculos.

As primeiras experiências militares mostraram que o problema fundamental da tecnologia de mísseis estava no fato de que eram impossíveis correções no seu trajeto após ter sido feito o lançamento. A primeira grande conquista deu-se com o desenvolvimento de sistemas de orientação capazes de calcular a posição do foguete em relação a um ponto na superfície (o local de lançamento) pela dedução da distância percorrida e de sua direção. Mas até mesmo os equipamentos modernos de alta qualidade estão sujeitos a erros graves.

Outro método mais preciso utiliza satélite em órbita geoestacionária como ponto de referência. A principal desvantagem desses sistemas é que a linha de vôo do míssil — e provavelmente seu alvo — pode ser calculada pelo inimigo imediatamente após o lançamento, dada a capacidade dos modernos radares de longo alcance. Para eliminar essa vulnerabilidade, projetou-se um míssil capaz de voar a baixa altura, provido de radar de varredura horizontal, que avalia dados para o cálculo do percurso até o alvo. Assim nasceu o míssil Cruise.

Esse míssil corrige continuamente sua posição, por meio da análise dos contornos do solo que sobrevoa. É feita uma seqüência de leituras de altitude, por um radar altímetro de alta precisão, acompanhado de um traçado ou mapa de contornos do terreno armazenado em uma memória de bolha.

Esse sistema, desenvolvido pela McDonnell Douglas é conhecido como TERCOM (TERrain COntour Matching, Comparação de Contornos do Terreno), ou DPW-23. Cada míssil pode armazenar em sua memória de bolha cerca de 25 "perfis de rota" alternativos, com os quais compara o terreno que está sendo sobrevoado. Todavia, há dificuldades nesse procedimento. Por exemplo, o sistema não pode ser utilizado em trajetos sobre a água, cuja superfície varia continuamente de conformação.

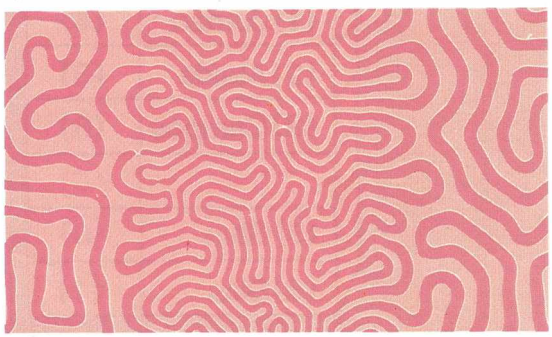


**Míssil autodirigido**
O míssil Cruise General Dynamics "Tomahawk" tem 6,40 m de comprimento e pesa menos de 1.200 kg. Lançado de um tubo montado sobre uma carreta, inicia seu trajeto como um foguete comum, mas logo desdobra suas pequenas asas e desce para um vôo baixo, impulsionado por uma turbina a jato extremamente pequena e compacta.

Sua confiabilidade é também precária sobre as areias do deserto, que estão em mudança constante. Suspeita-se ainda de que não possua precisão durante o rigor do inverno no norte da Europa, quando o perfil do solo se altera bastante pelas intensas nevascas.

O Cruise não utiliza esse sistema de orientação logo no instante do lançamento: ele permanece em movimento inercial, ou seja, numa só direção, durante o vôo no espaço não-inimigo. Sendo vulnerável ao ataque aéreo ou terrestre, ele mergulha para um limite de 15 m acima do solo tão logo começa a voar sobre território inimigo. Mesmo estando à distância de 1 km fora de rota, está previsto que se encontrará suficientemente próximo de uma das suas 25 rotas mapeadas, de modo que lhe é possível ajustar automaticamente a posição.

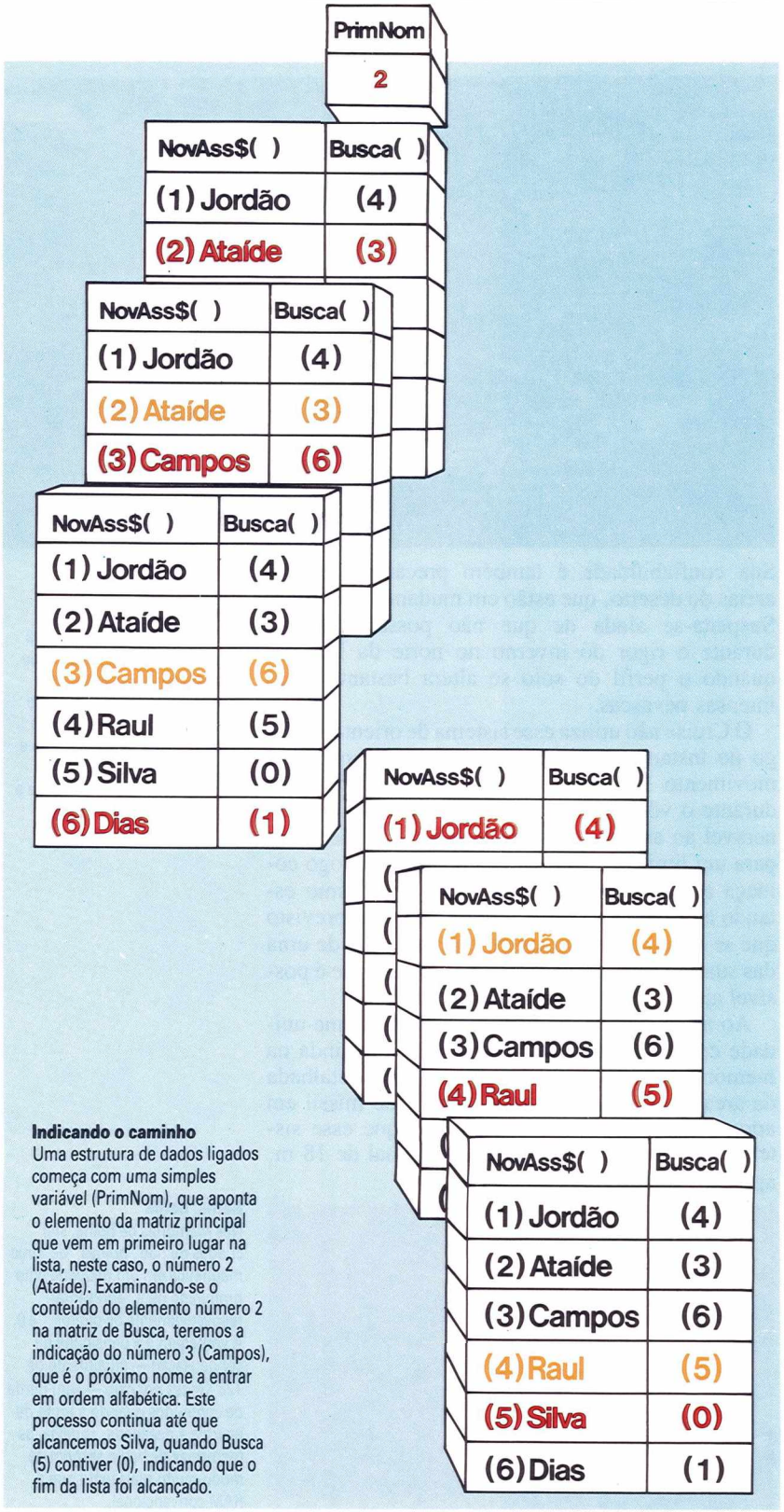
Ao aproximar-se do alvo, o míssil liga uma unidade correlatora terminal que contém — ainda na memória de bolha — uma imagem digital detalhada da área do alvo, como seria "vista" do míssil em aproximação. Os testes comprovam que esse sistema tem possibilidade de precisão final de 18 m, após um vôo de cerca de 2.800 km.



**Bolha, bolha**
Nas memórias de bolha, são criadas ou não "bolhas" de força magnética em um pequeno chip granulado para representar respectivamente os dígitos 1 e 0. A vantagem é a densidade de condensação — atualmente de 128 Kbytes por chip — sem perda de conteúdos quando a fonte de energia é desligada. Todavia, as memórias de bolha reagem de modo muito mais lento que a RAM convencional.

# Nomes encadeados

**A indexação é uma forma de estruturar muitos dados, tais como nomes e endereços. A lista de ligação ou encadeamento é outra alternativa que apresenta vantagens distintas.**




**Indicando o caminho**
Uma estrutura de dados ligados começa com uma simples variável (PrimNom), que aponta o elemento da matriz principal que vem em primeiro lugar na lista, neste caso, o número 2 (Ataíde). Examinando-se o conteúdo do elemento número 2 na matriz de Busca, teremos a indicação do número 3 (Campos), que é o próximo nome a entrar em ordem alfabética. Este processo continua até que alcancemos Silva, quando Busca (5) contiver (0), indicando que o fim da lista foi alcançado.

Na memória de um computador só existem dados, byte após byte, armazenados em milhares de padrões de voltagem. O significado é dado aos bytes através da estrutura de dados que o processador central impõe. Estas várias estruturas de dados decidem se algum byte específico é interpretado como parte de uma instrução, ou como dígitos pertencentes a um número maior, ou como um código de caractere.

Do ponto de vista do usuário, alguns tipos de estrutura de dados são ligados nos computadores por fios. Linguagens de programação geralmente exigem que os dados sejam estruturados segundo um limitado número de maneiras. O BASIC classifica dados como numéricos e alfanuméricos, e fornece variáveis e matrizes que possibilitam sua manipulação. Outras linguagens em geral utilizam esta e outras estruturas de dados, cuja extensão e variedade determinam o poder da linguagem.

As estruturas de dados em BASIC — variáveis e matrizes — constituem tudo que precisamos para simular outras maneiras de se examinarem dados.

A matriz indexada é uma estrutura de dados útil, facilmente utilizada em BASIC. Tem suas limitações, entretanto, particularmente quando os dados podem ser alterados de modo imprevisível.

Suponhamos que uma companhia telefônica tenha um arquivo dos novos assinantes, que serão incluídos na próxima lista telefônica. Até este momento, os nomes e endereços estão arquivados em ordem alfabética, para serem encontrados com maior facilidade, mas o arquivo cresce constantemente, e novos dados chegam a todo instante. Certo dia, o arquivo NovAss$( ) pode estar desta forma, quando colocado na matriz:

| NovAss$( ) | Índice( ) |
|---|---|
| (1) Jordão | (2) |
| (2) Ataíde | (3) |
| (3) Campos | (6) |
| (4) Raul | (1) |
| (5) Silva | (4) |
| (6) Dias | (5) |

A matriz Índice( ) mostra a ordem em que NovAss$( ) deve ser lido para que os novos itens sejam colocados em ordem alfabética. Desta forma, o primeiro item a ser colocado alfabeticamente é NovAss$(2), Ataíde. O segundo item é NovAss$(3), Campos. Neste exemplo, apenas os nomes estão sendo mostrados, mas, na verdade, uma lista abrange nomes, iniciais e endereços — cerca de sessenta caracteres normalmente. A movimentação de sessenta caracteres na memória é bastante lenta (já

que a seleção requer a movimentação de muitos dados) e desperdiça memória; sendo assim, é mais eficaz manter NovAss$( ) em seu lugar. Agora, um novo nome, Barros, deve ser acrescido à lista:

| NovAss$( ) | Índice( ) |
|---|---|
| (1) Jordão | (2) |
| (2) Ataíde | (7) |
| (3) Campos | (3) |
| (4) Raul | (6) |
| (5) Silva | (1) |
| (6) Dias | (4) |
| (7) Barros | (5) |

Note que o conteúdo do Índice( ) acima do novo acréscimo continua o mesmo, e o conteúdo abaixo dele permanece na mesma ordem anterior, mas todos pularam uma casa abaixo na matriz. Assim, o acréscimo no índice requer: encontrar a posição do novo elemento, mover cada elemento entre ele e o do fim do índice e escrever a nova entrada. É preferível fazer isso, em vez de fazer a mesma coisa com os dados reais, NovAss$( ), mas esse processo ainda é relativamente lento, se o índice for muito extenso.

Suponha, agora, que estruturamos os dados de outra forma. Deixemos NovAss$( ) não classificada, já que sua manipulação é lenta e cara. Vamos, então, estabelecer uma matriz paralela chamada Busca( ), cujos conteúdos são numerados de acordo com as posições equivalentes na NovAss$( ):

PrimNom(2)

| NovAss$( ) | Busca( ) | Índice( ) |
|---|---|---|
| (1) Jordão | (4) | (2) |
| (2) Ataíde | (3) | (3) |
| (3) Campos | (6) | (6) |
| (4) Raul | (5) | (1) |
| (5) Silva | (0) | (4) |
| (6) Dias | (1) | (5) |

A primeira diferença é que uma simples variável chamada PrimNom é necessária: ela indica NovAss$(2) que, alfabeticamente, é o primeiro elemento de NovAss$( ). A outra diferença é que o número (0) foi utilizado em Busca(5), o que indica que o NovAss$(5) é, alfabeticamente, o último da matriz.

A outra diferença diz respeito ao conteúdo do Índice( ) e de Busca( ). O Índice( ) deve ser lido: "o primeiro elemento está em NovAss$(2), o segundo está em NovAss$(3), o terceiro em NovAss$(6)" etc. Ao passo que a leitura do PrimNom é feita da seguinte maneira: "o primeiro elemento está em NovAss$(2); e Busca(2) diz que o próximo elemento está em NovAss$(3); Busca(3) diz que o elemento seguinte está em NovAss$(6), e assim por diante. Busca(5) diz que NovAss$(5) é o último elemento. O Índice( ) dá uma posição absoluta para os elementos do arquivo, enquanto Busca( ) fornece apenas posições relativas; qualquer item em Busca( ) pode lhe informar apenas onde encontrar o elemento seguinte e não dá nenhuma informação sobre a posição absoluta. O número de Índice(4) indica o quarto item que está alfabeticamente organizado no arquivo, enquanto o número em Busca(4) indica apenas o item que segue o NovAss$(4) no arquivo classificado. O Busca( ) auxilia a estrutura de dados chamada Lista de Ligação. A leitura de uma Lista de Ligação é como a procura de um tesouro: no início você recebe informações sobre o caminho a seguir; quando chega a certo local, encontra uma indicação do próximo etc. A leitura de uma matriz organizada em Índice é como participar de uma corrida de automóveis. No início, você recebe informações a respeito da rota e a ordem do percurso.

A grande vantagem da estrutura de Lista é a flexibilidade que apresenta. Considere a Lista após o acréscimo do elemento Barros:

PrimNom (2)

| NovAss$( ) | Busca( ) |
|---|---|
| (1) Jordão | (4) |
| (2) Ataíde | (7) |
| (3) Campos | (6) |
| (4) Raul | (5) |
| (5) Silva | (0) |
| (6) Dias | (1) |
| (7) Barros | (3) |

A matriz Busca( ) foi modificada em dois lugares:

i) Busca(2), que anteriormente indicava o NovAss$(3) que continha o próximo elemento em ordem alfabética depois do NovAss$(2), agora indica o NovAss$(7), já que ele passa a representar o elemento seguinte ao NovAss$(2).

ii) Busca(7), que antes não era usado, agora indica o NovAss$(3) como o próximo item depois do NovAss$(7), que aparece em ordem alfabética.

Isso ilustra o processo geral de inserção ou acréscimo em uma Lista de Ligação: encontrar o elemento da lista que deveria vir antes do novo elemento, e fazer com que ele aponte o novo elemento; a seguir, fazer com que o novo elemento indique aquele que perdeu a colocação. Estas operações constituem tudo que é necessário para um acréscimo à Lista de Ligação, e apenas as operações básicas serão afetadas pelo tamanho da Lista. A inserção de um elemento na Lista é como pôr um elo numa corrente — decida onde colocar o elo, rompa a corrente, junte o novo elo ao anterior e ao seu consecutivo. As Listas de Ligação são às vezes chamadas de Listas de Encadeamento. Os números em Busca( ) — os elos — são chamados ponteiros.

Um fator importante das Listas é sua característica de seqüência: é impossível encontrar um elemento na Lista a não ser começando pelo início da Lista e pesquisando cada elemento até encontrar aquele desejado. A Lista é aqui complementada pelo uso de matrizes, que são projetadas com o objetivo de serem estruturas de Acesso Direto, mas a Lista transformou-as efetivamente em Arquivos Seqüenciais. Em outras linguagens, como LISP ou PASCAL, o recurso da Lista já vem incorporado.

As Listas são estruturas úteis quando se quer lidar com dados dinâmicos (dados que mudam com freqüência) e quando se trata de uma linguagem natural (como no reconhecimento da fala) ou de uma artificial (compilação de programas), onde os próprios dados formam uma lista de elementos.

# Apresentando o som...

**"Som e luz" é a nova seção que vai ajudá-lo a melhor utilizar os recursos sonoros e gráficos existentes em seu computador.**

À medida que os microcomputadores foram se desenvolvendo, tornou-se maior a quantidade de recursos disponíveis. Os dispositivos para jogos concorreram para a aceitação comercial de cada novo modelo e também investiram-se muito tempo e esforço no aperfeiçoamento de capacidades gráficas. Embora sua importância não tenha sido reconhecida de imediato, os recursos sonoros e a produção de música igualmente se desenvolveram com rapidez.

Se você perguntar aos programadores de jogos de maior aceitação sobre a importância das rotinas de som em seus programas, eles as colocarão, provavelmente, em posição muito próxima ao plano conceitual do jogo e aos gráficos. O uso engenhoso de efeitos sonoros e de música amplia consideravelmente a capacidade de entretenimento e de atração de todos os jogos do tipo fliperama.

Além das aplicações em jogos, é possível estender conhecimentos de música com a utilização dos recursos sonoros de seu microcomputador. Em muitos casos, comandos especiais para música são proporcionados em BASIC, habilitando o usuário a desenvolver programas curtos para tocar melodias bastante complexas, que até mesmo incluem acordes. Alguns computadores também dispõem de meios de mudar a característica do som, para torná-lo mais agradável ao ouvido, ou para imitar os sons de instrumentos musicais conhecidos. Em todos esses casos, o teclado do computador pode ser "adaptado", por meio de um programa adequado, para atuar de modo semelhante ao teclado de um piano, o que capacita o usuário a tocar a música em "tempo real"

Mesmo que você tenha pouco conhecimento de programação, é possível escrever programas curtos e simples para a produção de sons musicais razoavelmente bem elaborados. Se quiser aproveitar melhor os recursos sonoros, a maioria das lojas de software tem programas de música completos, com os quais você poderá escrever e tocar melodias imediatamente. Qualquer que seja seu propósito, é interessante compreender como seu computador gera, modela e dirige a emissão de som.

# ...e a luz

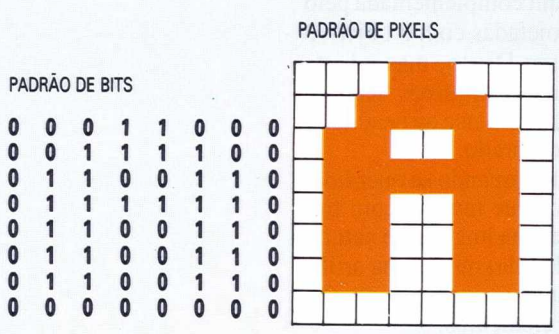
## Alta e baixa resolução

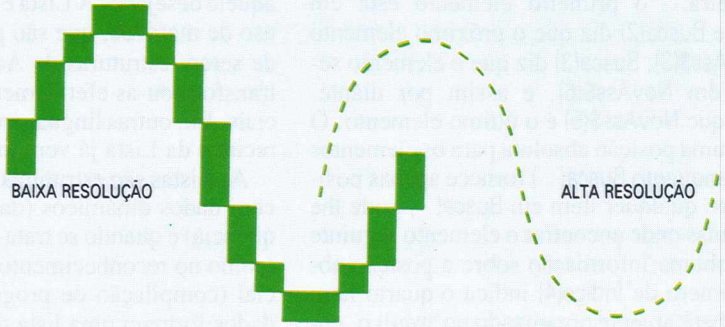
Os gráficos dos microcomputadores podem ser divididos em duas categorias: de baixa resolução e de alta resolução. A diferença entre as duas será melhor compreendida se examinarmos como é feito um caractere (letra, número ou forma).

Se você olhar detalhadamente um caractere standard impresso na tela da televisão, notará que sua forma é constituída por um conjunto de pequenos quadrados, chamados elementos da figura, ou pixels, e cada caractere ou forma que aparece na tela é um arranjo desses elementos de acordo com um padrão. Na maioria dos microcomputadores, os caracteres são formados por um quadriculado de 8 × 8 linhas, reunindo 64 pixels. A letra A pode ser constituída por um padrão de pixels como o seguinte:

Cada pixel iluminado no quadriculado pode ser representado na memória do computador pelo dígito 1 e cada pixel não iluminado pelo dígito 0. Oito bits constituem 1 byte e, assim, cada linha do quadriculado de caracteres pode ser armazenada em uma posição individual na memória do computador. Portanto, são necessárias oito posições de memória para comportar um único caractere.

As imagens gráficas às vezes são constituídas por blocos que têm o tamanho de um quarto, meio ou um quadriculado inteiro de caractere. Os gráficos projetados com esses blocos grandes e simples são denominados "de baixa resolução". Em muitos microcomputadores é agora possível desenhar imagens gráficas constituídas de pixels individuais; essas imagens são chamadas "de alta resolução". A diferença entre os dois tipos pode ser notada no gráfico de uma curva senoidal, como na ilustração abaixo.



## Osciladores

São circuitos eletrônicos que produzem sinais (impulsos) repetitivos. Quando ampliados e transmitidos a um reprodutor de som, esses sinais geram emissões de uma determinada altura. A quantidade de osciladores nos microcomputadores varia de um a quatro — quanto maior o número de osciladores, maior o número de notas que você poderá tocar ao mesmo tempo.

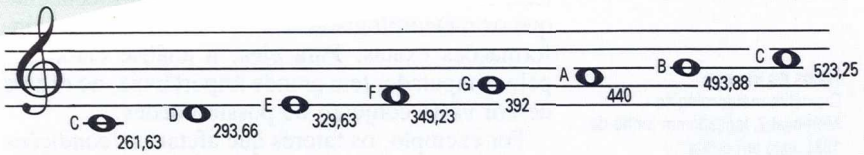
São três as características que descrevem o som criado: freqüência, envelope (que inclui o volume) e forma de onda. A freqüência é apresentada aqui; os geradores de envelope e a forma da onda, na próxima parte dessa seção.

## Freqüência

É a mais importante das características necessárias ao controle, pois determina a altura do som. A freqüência é o número de vezes que um sinal (ou impulso) se repete a cada segundo, e é medida em hertz (Hz, ciclos por segundo). Os sons que podem ser captados pela audição humana estão em freqüências entre 20 Hz e 20.000 Hz. Embora não possam ser ouvidas, as freqüências abaixo de 20 Hz podem ser utilizadas para modificar as características dos sons audíveis. Esta técnica é chamada modulação.

Todavia, não é necessário aprofundar muito a descrição das freqüências. O que você realmente precisará saber é como tocar as notas musicais. A facilidade para fazê-lo varia enormemente de uma máquina para outra. Algumas têm comandos em BASIC que produzem as freqüências por você, de modo que seu trabalho consistirá apenas em especificar um número para a altura ou então o símbolo de uma nota musical — A, A#, B, e assim por diante. Em outras, esse procedimento é consideravelmente mais difícil, porque é preciso consultar uma tabela no manual do usuário, para ver a freqüência correspondente à nota, e armazenar, pelo comando POKE, o valor da freqüência em uma posição da memória. A ilustração abaixo apresenta as conversões exatas para a escala de dó central (C); é útil também para a programação de música em código de máquina, porque o BASIC, nesse caso, não poderá ajudá-lo a calcular as freqüências.

**Das notas às freqüências**
Você pode calcular a freqüência de cada nota da escala multiplicando um semitom abaixo dela por 1,0594631. Isto pode parecer um tanto estranho, mas, se a multiplicação for feita 12 vezes, a freqüência original será duplicada. Há 12 semitons em uma oitava (a diferença entre duas notas de mesmo nome); desse modo, a duplicação da freqüência eleva o tom para uma oitava acima. A ilustração abaixo fornece as conversões exatas dos símbolos das notas musicais (para a escala de dó central) nas freqüências correspondentes.



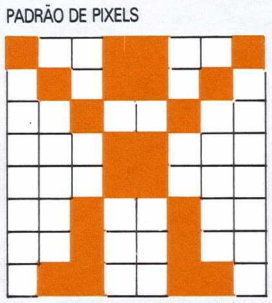

## Definição pelo usuário

Para criar imagens atraentes e originais na tela será interessante recorrer a caracteres que não sejam do conjunto normal de caracteres alfanuméricos. O Vic-20 e o Commodore 64 possuem um conjunto especial de caracteres gráficos que pode ser utilizado diretamente a partir do teclado, porém, mesmo estes não abrangem todas as possibilidades. Na maioria dos microcomputadores é possível criar novos caracteres. Isso é conseguido pela redefinição dos padrões binários de oito posições de memória, nas quais o caractere é armazenado. No processo, o conjunto original de padrões binários geralmente se perde ou é "sobreposto", e o caractere "definido pelo usuário" assume algumas das propriedades daquele que está sendo substituído na memória. Desse modo, o novo caractere pode ser utilizado em instruções PRINT, pelo simples pressionar da tecla do caractere substituído. Eis aqui um exemplo de caractere definido pelo usuário, acompanhado por seus códigos binários correspondentes.

PADRÃO DE BITS

| 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 |

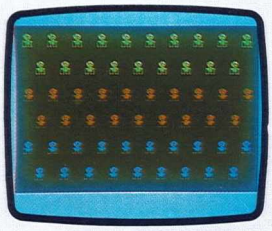
PADRÃO DE PIXELS



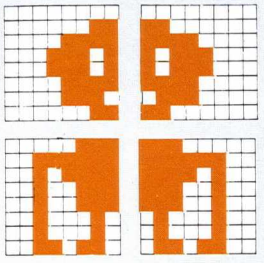
A facilidade com que os caracteres definidos pelo usuário podem ser constituídos varia enormemente, de acordo com o computador utilizado. Por exemplo, com o comando USR, dos computadores compatíveis com o Sinclair (TK's, CP 200, Ringo), é necessário apenas o fornecimento dos padrões binários adequados; no Commodore 64 o usuário tem de, primeiro, transpor o conjunto completo de caracteres, de ROM para RAM, antes de armazenar na memória, com o comando POKE, os oito equivalentes decimais dos padrões de bits que constituem a forma. Entretanto, vários programas para desenho de caracteres encontrados em fornecedores independentes facilitam o trabalho do usuário do Commodore 64.

Para criar figuras maiores é possível agrupar dois ou mais caracteres definidos pelo usuário. As figuras dos alienígenas (mostradas à direita) foram construídas a partir de quatro caracteres definidos pelo usuário. O programa, que pode ser processado no Commodore 64, imprime os grupos de caracteres na tela em três cores. Os caracteres foram criados usando-se uma pequena rotina para transportar o conjunto normal de caracteres de ROM para RAM e para substituir os caracteres gráficos ▞, ▙, ▚ e ' ' pela leitura de números decimais em instruções DATA, utilizando comandos POKE para colocá-los nas posições apropriadas. Você pode ter maiores detalhes de como fazer isso em outras partes do curso.

Mesmo quando o computador dispõe de sprites (ver p. 152), há geralmente um limite para a quantidade que pode ser apresentada ao mesmo tempo na tela; assim, os gráficos definidos pelo usuário mostram-se úteis quando muitas formas semelhantes têm de ser apresentadas na tela ao mesmo tempo.



**Extraterrestre**
Essas criaturas alienígenas foram criadas de quatro caracteres, definidos pelo programador. O método pode ser empregado em muitos micros que não têm sprites.



# Tempo de observação

## Computadores de alta velocidade tornaram mais exatas as previsões meteorológicas, com o processamento de imagens de satélites e a análise das configurações de dados recebidos.

Os resultados de complexos processamentos de informações estão presentes em nossa vida diária, sem que, geralmente, tenhamos conhecimento disso. Uma das mais avançadas aplicações do computador, e que requer uma capacidade de processamento de dados superior a quase todas as outras em um país, nos dá informações e prognósticos diários sobre as condições meteorológicas. Como a previsão do tempo não é uma tarefa simples, é surpreendente que os meteorologistas tenham, de modo geral, informações exatas. Para eles, a análise elaborada pelo computador tem grande importância, no exame de um vasto conjunto de possibilidades.

Por exemplo, os fatores que afetam as condições meteorológicas nas linhas britânicas e, em menor escala, no litoral atlântico da Europa são extremamente complexos, devido à proximidade do pólo norte e à contigüidade com o Atlântico. Situadas na parte leste do oceano, as ilhas britânicas estão submetidas às conseqüências climáticas criadas, em seus 4.000 km de extensão, pelo efeito Coriolis — fenômeno que, com a rotação oeste-leste da Terra, impulsiona, no hemisfério norte, os ventos para a direita. (No hemisfério sul, para a esquerda.) Lembremos que, no equador, por exemplo, um objeto na superfície está sendo transportado a mais de 1.600 km/h; o poderoso movimento de rotação do planeta, combinado com as correntes de ar que descem do pólo para o equador, cria os ventos dominantes originados na parte oeste do hemisfério. Essa poderosa corrente constante de ar úmido — que se eleva ou abaixa, condicionando as variações locais de temperatura — influencia as condições meteorológicas da Grã-Bretanha.

Os meteorologistas britânicos baseiam-se essencialmente nas observações feitas pelas estações de coleta de dados atmosféricos, situadas em locais estratégicos do Atlântico — navios apropriados, balões, bóias e aeroplanos de patrulha —, que fornecem os dados sobre as condições próximas. É possível prever, então, o que vai acontecer, à medida que



**Fotos do espaço**
O satélite meteorológico Meteosat 2, lançado em junho de 1981, está em órbita geoestacionária (ou seja, ele não se move em relação à Terra), a cerca de 35.880 km acima do equador, no meridiano zero. Ele colhe e retransmite informações de um grande número de estações terrestres.

os fenômenos climáticos se aproximarem da Terra, tomando como base o comportamento conhecido de tais fenômenos em situações anteriores.

Antes de março de 1979, quando foi lançado o satélite meteorológico Meteosat 1, o único método de previsão utilizado pelos técnicos eram os relatórios gráficos das estações meteorológicas, ou seja, os diagramas isobáricos. Isóbares são linhas imaginárias que unem pontos de igual pressão barométrica (como as linhas de contorno em um mapa que unem os pontos de mesma altitude). A partir disso, pode-se avaliar a velocidade e a direção das frentes frias ou quentes, com os seus respectivos ciclones e anticiclones; assim é feita a previsão meteorológica ou, melhor dito, a suposição ou estimativa baseada na experiência adquirida.

Embora os diagramas isobáricos sejam o método mais comum, eles não representam os únicos mapas de tempo produzidos pelo serviço meteorológico. Da imensa quantidade de dados registrados em sistemas de computador elaboram-se cartas sinópticas que mostram temperatura média, chuvas, horas de sol por dia, e assim por diante.



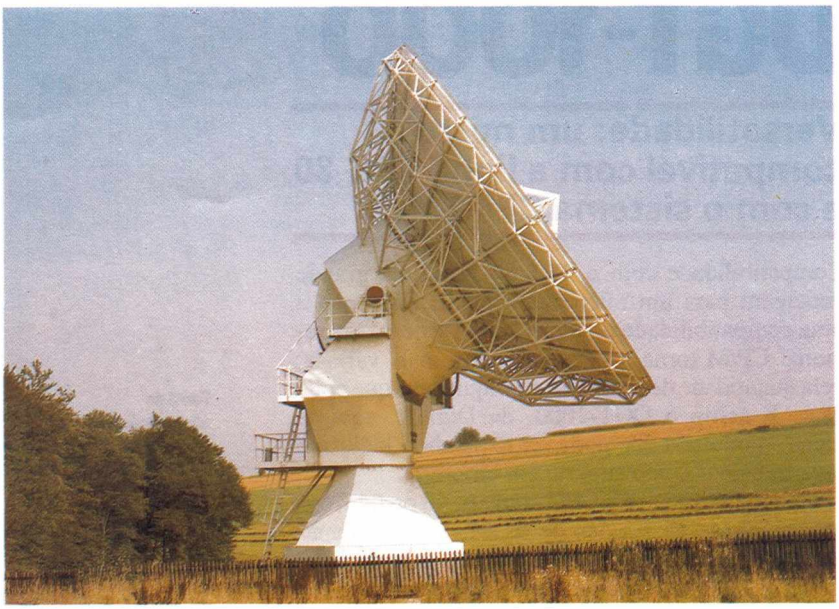
## Salada de números

Um dos principais usos dos grandes computadores na pesquisa científica é o processamento puramente numérico de informação. Aplicações de ciência pura, como a física nuclear, e de ciência aplicada, como a meteorologia, têm esses requisitos. Se alguém realizasse cálculos nesse nível de complexidade num microcomputador, a quantidade de tempo perdida seria um aspecto proibitivo, não só devido ao número de termos da equação, mas também à magnitude absoluta dos números envolvidos, que podem ir a 30 ou mais pontos decimais. Para realizar essas funções em tempo razoável, são necessários computadores muito rápidos, com grande capacidade de memória.

O serviço meteorológico britânico ainda segue este procedimento em seus detalhados mapas diários das condições do tempo, mas, também, recebe agora as imagens — sinais analógicos — enviadas pelo Meteosat. Os sinais são digitalizados, processados e exibidos pelo computador na forma de mapas coloridos artificialmente. As imagens apresentam um quadro nítido do estado atmosférico. São reproduzidas de 4 em 4 minutos, para que o meteorologista possa observar a evolução das condições climáticas em tempo real.

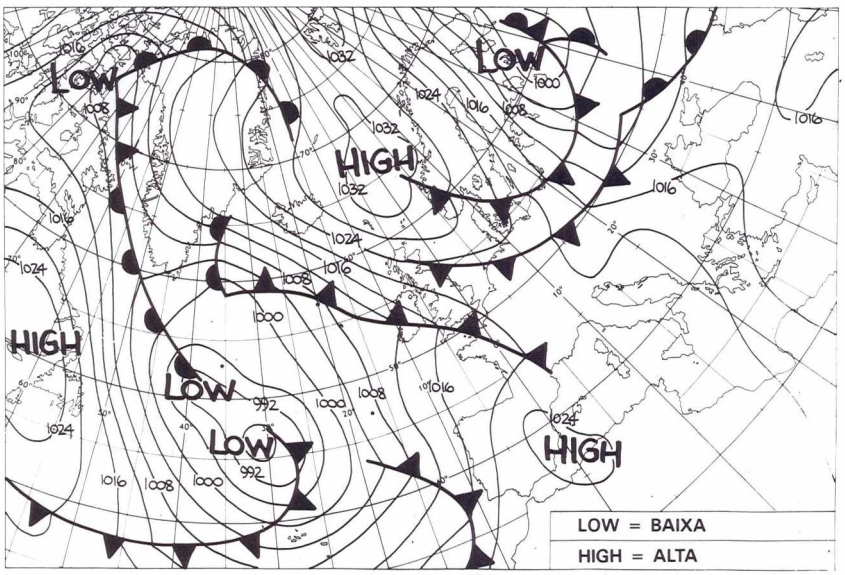
O Meteosat 2, que substituiu um outro satélite em junho de 1981, situa-se numa órbita geoestacionária a cerca de 35.880 km acima do equador. Esse satélite reúne os dados de um grande número de estações meteorológicas espalhadas pela superfície do globo, e retransmite as informações a qualquer país ou pessoa que deseje participar do sistema.

Seria teoricamente possível analisar e interpretar essas informações (não em tempo real) num microcomputador, em casa, gravando-se os dados recebidos em disco, à medida que eles chegassem do satélite. Entretanto, o sinal é analógico, e a conversão pode ser difícil. Além disso, o usuário precisaria instalar sua própria antena, alinhada com precisão



**Estação terrestre**
As antenas de intercomunicação com satélite geoestacionário variam em tamanho e complexidade. Na foto, a antena de prato (assim chamada devido ao seu formato), além de receber dados meteorológicos, possui um sofisticado sistema computadorizado que faz com precisão o rastreamento de um satélite em órbita.

para o satélite. O processamento dessas imagens de satélite é apenas uma pequena função do sistema de computador do serviço meteorológico. Juntamente com organizações similares em outras partes do mundo, o serviço inglês mantém um modelo de sistema meteorológico global, e extrai desse modelo um vasto acervo de dados estatísticos. Assim é formado um banco de dados com um histórico de informações, por meio do qual as tendências climáticas locais e globais são traçadas. Não se incluem,




**Diagramas isobáricos**
Os "mapas meteorológicos" são diagramas de pressão barométrica. As linhas concêntricas unem pontos de igual pressão de ar. Os ventos fluem em sentido anti-horário próximos de um local assinalado como "baixa" e em sentido horário próximos das regiões marcadas como "alta" (ocorre o inverso no hemisfério sul); a velocidade do vento está diretamente relacionada à distância entre os isóbares.

aí, apenas dados barométricos, mas detalhes sobre a velocidade e a direção dos ventos, as chuvas e a temperatura, ao nível de terra e mar, como também a determinadas altitudes.

O conjunto desses dados é importante para a análise histórica. É vital para a agricultura, para muitas indústrias e também para a economia e a ecologia dos continentes. Só pela análise desse conjunto, as mudanças climáticas do planeta podem ser reconhecidas. Exemplos disso são os resultados da destruição progressiva da floresta amazônica pelas chuvas e o aumento das calotas polares, que talvez indiquem a aproximação de uma outra era glacial.

# DGT-1000

## Versatilidade: um micro compatível com a linha TRS-80 e com o sistema CP/M.

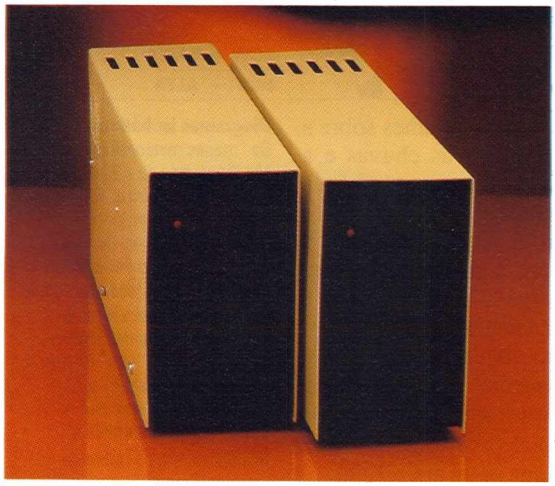
Compatibilidade com o TRS-80 já é uma grande vantagem para um microcomputador; aliar isso a uma compatibilidade opcional com o sistema operacional CP/M torna a máquina ainda mais valiosa, pela quantidade de programas e linguagens que pode usar. É assim o DGT-1000, da Digitus, equipamento modulado que pode ser expandido inclusive para trabalhar como suporte em empresas.

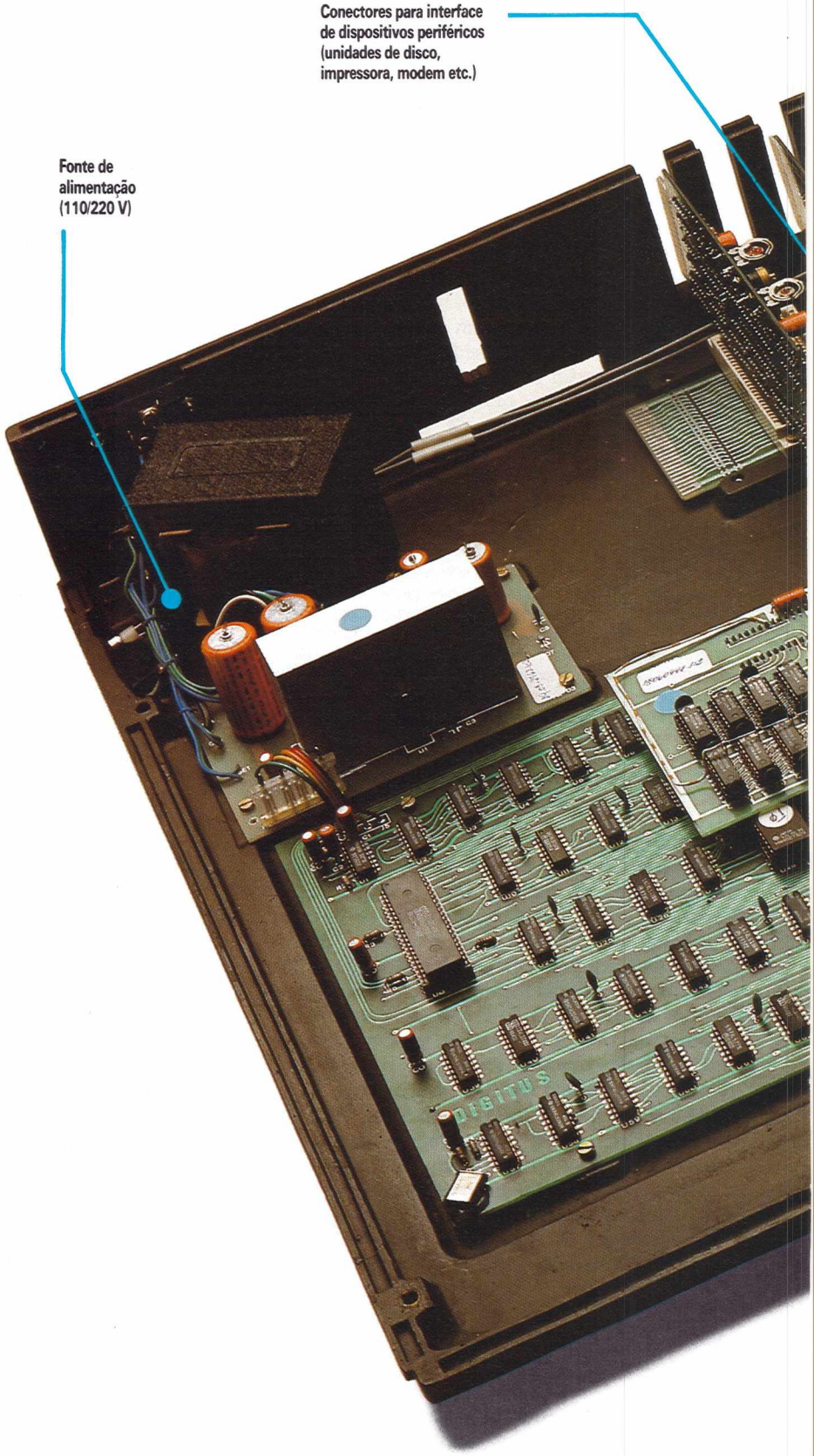
O módulo básico é a CPU com 16, 48 ou 64 Kbytes de RAM, 16 Kbytes de ROM, teclado semelhante ao das máquinas de escrever, teclado numérico separado para facilitar a entrada de dados e conectores para gravador cassete e monitor de vídeo.

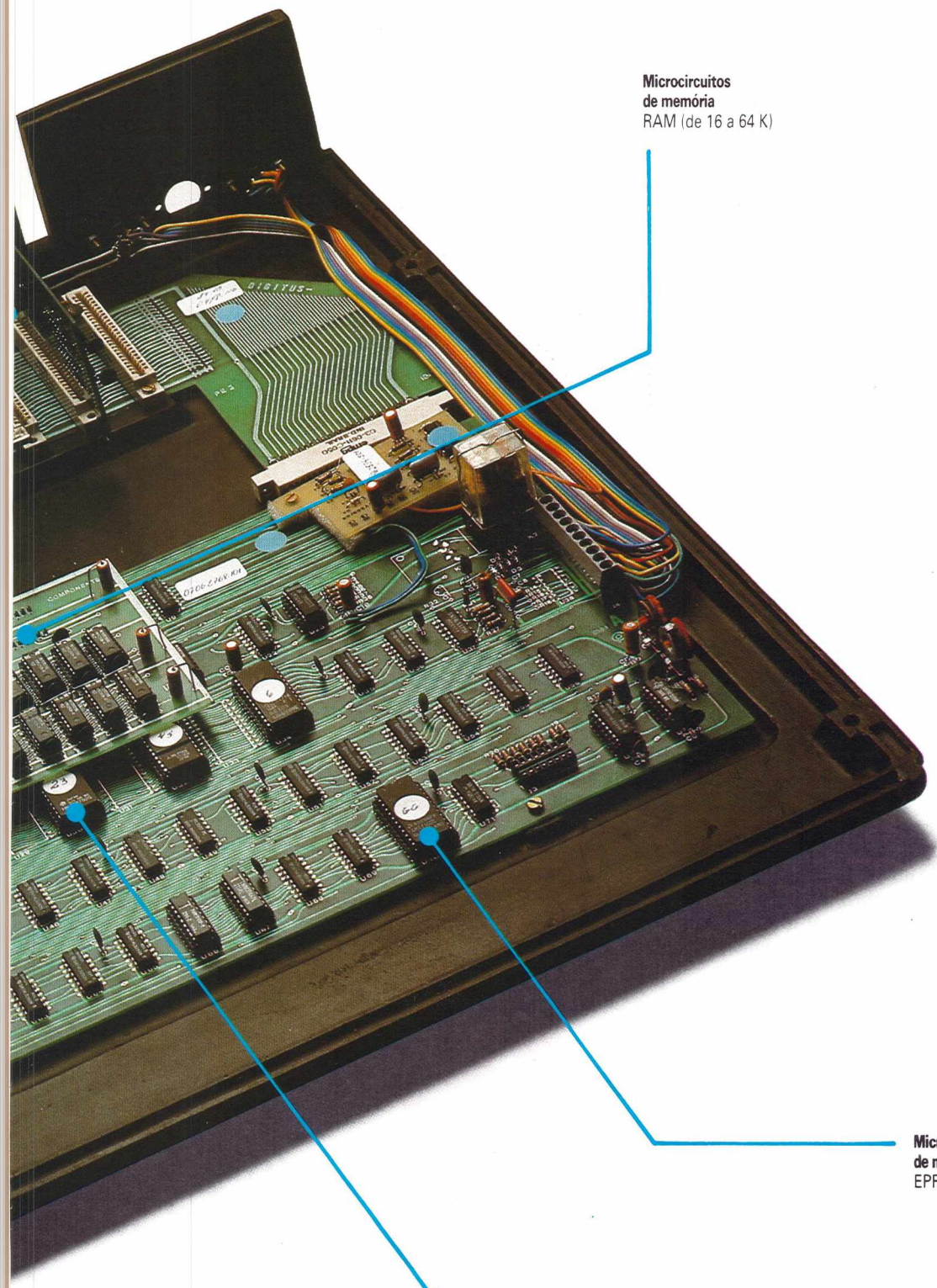
A linguagem residente (contida na ROM) é o BASIC, mas há um programa monitor, também residente, para quem quiser programar em linguagem de máquina.

A configuração mínima (CPU, cassete e monitor de vídeo) pode ser aperfeiçoada com o acréscimo de até quatro unidades de disquetes de 5 1/4 pol. (densidade dupla, face simples ou dupla), com 92 Kbytes por face, impressora serial ou paralela tipo Centronics, monitor colorido de alta resolução gráfica (16 cores e 192 x 256 pontos), sintetizador de voz e modem. Para isso basta acrescentar as interfaces apropriadas. Com interfaces, o DGT tem possibilidade de acesso ao Videotexto da Telesp e ao sistema de comunicações do projeto Ciranda, da Embratel.

Além da vasta linha de programas para os micros TRS-80 (e os compatíveis com eles), o DGT-1000 pode usar programas feitos para o sistema operacional CP/M, bastando para isso acrescentar a placa com o sistema DGP/M, feita pela Digitus. Desse modo, o micro pode também usar outras linguagens de programação, inclusive as compiladas, como FORTRAN e COBOL. Outro recurso do DGT-1000 é o equipamento Digplex, por meio do qual ele pode comunicar-se com dezesseis outros, e ser usado para fins educacionais.







**Microcircuitos de memória**
RAM (de 16 a 64 K)

**Microcircuito de memória**
EPROM de vídeo (2 K)

**Microcircuitos de memória**
EPROM de linguagem e sistema operacional (16 K)

## DGT-1000

**MICROPROCESSADOR**

Z80

**CLOCK**

2,5 MHz

**MEMÓRIA**

Versões de 16, 48 e 64 Kbytes de RAM, com 16 K de EPROM para o sistema operacional e o BASIC residente.

**VÍDEO**

De fósforo verde ou branco, com 16 linhas e 64 ou 32 colunas de texto; resolução de 128 x 48 elementos; interface opcional para monitor colorido (16 cores) com resolução gráfica de 192 x 256 elementos.

**TECLADO**

Profissional, letras maiúsculas e minúsculas, teclado numérico separado, teclas com auto-repetição.

**LINGUAGENS**

BASIC (residente); COBOL, FORTRAN, PASCAL, PL1, CBASIC e outras são disponíveis, se usado o sistema operacional DGP/M (compatível com CP/M).

**PERIFÉRICOS**

Cassete, monitor de vídeo, até quatro unidades de disco de 5 1/4 pol., impressora, modem.

**DOCUMENTAÇÃO**

Os manuais que acompanham o equipamento contêm noções básicas de operação do micro, de programação em BASIC e os esquemas de hardware da CPU.

# O visual dos caracteres

**O "gerador de caracteres" é a parte da memória do computador que define a forma dos caracteres na tela. Em alguns sistemas, é possível ao usuário desenhar seus próprios símbolos.**

Já vimos no curso de programação BASIC (ver p. 214) que todos os caracteres alfanuméricos — e os símbolos gráficos, caso seu computador os tenha — são armazenados na memória RAM na forma de códigos de 8 bits (geralmente em ASCII); assim sendo, cada caractere ocupa 1 byte.

Quando a informação é impressa na tela, os códigos para cada caractere são depositados numa área reservada da memória chamada RAM de vídeo. Se, por exemplo, a letra A é impressa no alto, no canto esquerdo da tela, o primeiro byte da RAM de vídeo irá conter o código 65 (ASCII para A). Se um C é impresso embaixo do A, e o computador tem uma tela de quarenta colunas, o valor 67 será encontrado na 41.ª locação da RAM de vídeo, e assim por diante. Como o computador converte o valor 65 ao padrão de pontos que formam o caractere A na tela? Isto fica a cargo do dispositivo chamado gerador de caracteres.

Um gerador de caracteres nada mais é do que uma coleção de configurações ou padrões depositados como bits na memória. Os microcomputadores têm o gerador de caracteres armazenado em ROM, permitindo uma aparição imediata de caracteres assim que a máquina é ligada. O gerador de caracteres pode ser incorporado à ROM que contenha o interpretador BASIC e o sistema operacional, ou estar no próprio chip da ROM. Quando isto ocorre, é comum encontrar fornecedores independentes que oferecem a recolocação de ROMs que produzirão um grupo de caracteres de língua estrangeira, por exemplo, ou uma série de símbolos especializados para, digamos, engenharia ou matemática. Entretanto, um número crescente de computadores permite que o gerador de caracteres seja transformado em RAM — admitindo que o programador projete seus próprios caracteres e símbolos.

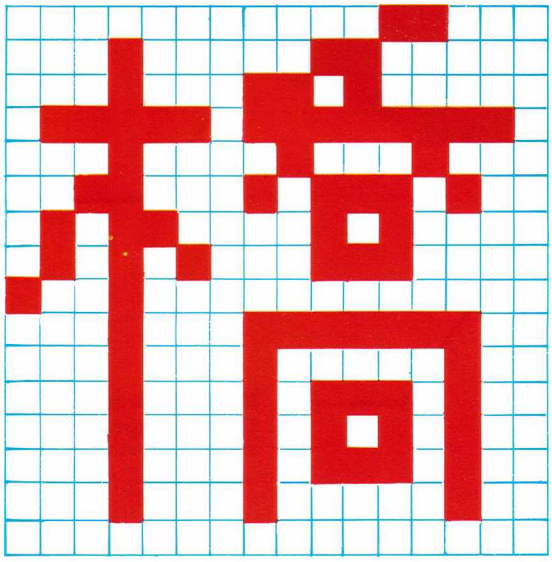
Todos os caracteres são construídos numa matriz de pontos, que num microcomputador é de 8 x 8, embora uma matriz maior resulte em melhor legibilidade e em maior variedade de caracteres a ser mostrada. Os caracteres são formados com o preenchimento de quadriculados na retícula. O quadriculado preenchido representa o dígito 1, e o quadriculado em branco o dígito 0, dando o total de 64 bits, ou 8 bytes para cada caractere.

O primeiro byte no gerador de caracteres representaria o padrão de bit para a linha do alto do primeiro caractere na tabela. Se o computador só apresenta caracteres em ASCII, com códigos de 0 a 127, então o gerador de caracteres vai requerer 128 x 8 bytes (1 Kbyte de memória).

A dificuldade para o computador é que, quando o varredor na tela de televisão está gerando a linha mais ao alto do vídeo, ele precisa produzir a linha mais ao alto de pontos para o caractere posicionado no alto da tela, à esquerda; em seguida, produz a linha mais ao alto do caractere posicionado a sua direita, e assim por diante, varrendo toda a tela. Dessa maneira, quando o varredor começa sua segunda passagem, precisa encontrar e mostrar a segunda fileira de pontos para cada um dos caracteres da linha anterior.

O circuito de vídeo atinge isso por meio de dois contadores independentes. Um contador mantém a pista da posição de memória de vídeo correspondente ao ponto por onde o varredor está passando. O outro conta as linhas do varredor, começando do zero para a primeira linha e atingindo o sete para a oitava, recomeçando do zero para a nona, e assim por diante. Dessa forma, o computador consulta o ASCII ou código de display na memória de vídeo, multiplica-o por 8 e adiciona ao valor do contador de linha naquele momento. Isto fornece um endereço do gerador de caracteres para um padrão de 8 bits que corresponde à linha correta do caractere que está sendo examinado.

Tomemos o exemplo da geração do caractere A, que tem um código ASCII de 65. Podemos calcular que a primeira linha do caractere será armazenada no byte de número 520 (65 × 8 + 0); a segunda linha no byte 521 (65 × 8 + 1); a terceira linha no byte 522 (65 × 8 + 2) etc. Resta apenas para o circuito de vídeo converter aqueles 8 bits numa seqüência de voltagens que ligará ou desligará o raio de elétrons varredor para mostrar os caracteres na tela.





**Caracteres complexos**
Os caracteres japoneses, para quem não os conhece, são bastante complicados, e a matriz normal 8 × 8 não mostra detalhes suficientes para que eles sejam lidos. O caractere para "ponte" (ao lado) só pode ser lido em matriz de 16 × 16. Obtém-se maior clareza com matriz de pontos de 24 × 24.

# Questão de segurança

**A "paridade par" garante que o número de bits de valor 1 em 1 byte seja sempre par. Assim, detectam-se erros de transmissão.**

Uma das principais vantagens dos computadores digitais em relação aos dispositivos analógicos é que as falhas e imprecisões comuns nos circuitos elétricos não se acumulam à medida que os sinais passam por muitos circuitos (ver p. 239). Todavia, quando os dados são transmitidos a qualquer distância — seja por uma interface serial e um par de fios, seja por uma linha telefônica —, o "ruído" elétrico de fundo na linha pode às vezes ser suficiente para alterar um único bit de 0 para 1, e vice-versa. Normalmente, o equipamento receptor não tem meios de distinguir se isto aconteceu e aceitará os dados errôneos como corretos.

Observe o que ocorre se um bit no código ASCII para a letra Q for adulterado:

[ ] 1010001 (Código ASCII transmitido para Q)
[ ] 1000001 (Código ASCII recebido para A)

Um erro como esse na transmissão dos dados será, no mínimo, desagradável e, potencialmente, uma catástrofe. Entretanto, você se recorda de que os códigos ASCII são atribuídos apenas para valores até o total de 127, o que requer a utilização de apenas sete bits (numerados de 0 a 6). O bit mais significativo (bit 7) é, portanto, freqüentemente utilizado como bit "de paridade", para detectar a ocorrência de erros.

Há duas convenções para a utilização de bits de paridade: "paridade par" e "paridade ímpar". Examinaremos a primeira. A expressão "paridade par" significa que o bit de paridade (o bit 7, em código ASCII) é fixado de modo que o número total de bits de valor 1 no byte seja sempre um número par. Eis como as letras A e Q seriam representadas com paridade par:

[0] 1000001
(o código ASCII para A, pela paridade par)
[1] 1010001
(o código ASCII para Q, pela paridade par)

Há dois bits de valor 1 no código ASCII para A; assim, o bit de paridade assume o valor 0, de modo que o total de todos os oito bits seja par. No código ASCII para Q há três bits de valor 1; assim, o bit de paridade é constituído como um dígito 1. Isto eleva para 4 (que é par) o número total de bits de valor 1.

Agora examinemos o que acontecerá se o bit 4 em nossa letra Q, em código ASCII, for adulterado, como no exemplo acima.

[1] 1000001 (Q em ASCII adulterado)

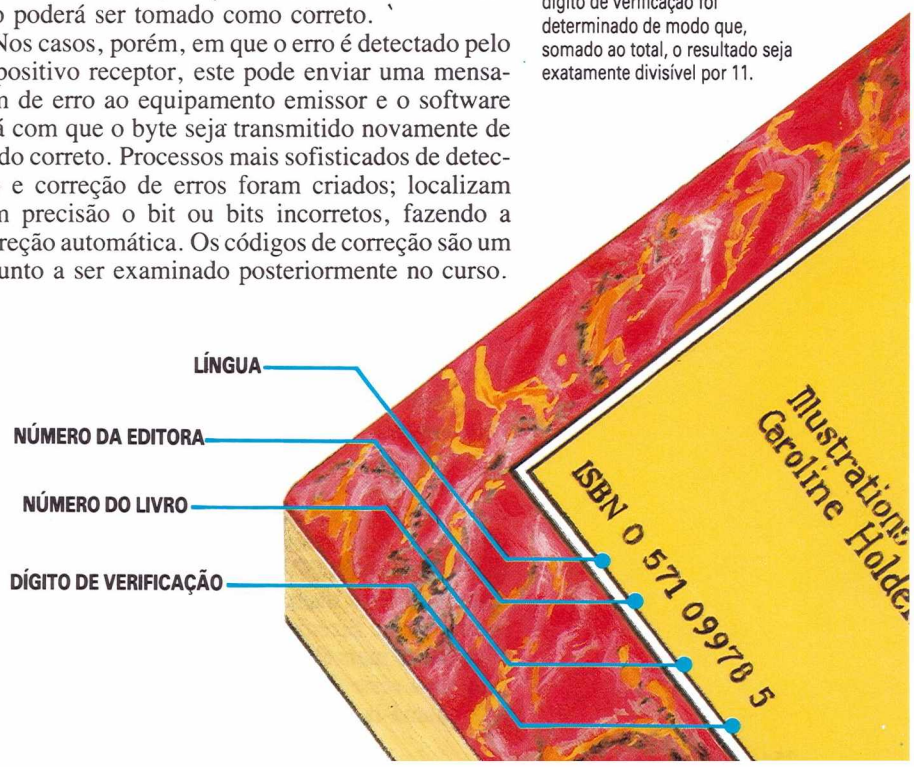
Quando a paridade do byte for verificada (seja pelo software ou por hardware especial), será observado que a letra Q correta tem um número par de valores 1 (inclusive o bit de paridade). A letra Q adulterada, entretanto, teve o bit 4 acidentalmente alterado de 1 para 0, mas o bit original de paridade — bit 7 — permanece com o valor 1. Ao ser conferida a paridade deste byte adulterado, será verificado que ele possui um número ímpar de bits 1 e, desse modo, será detectada a adulteração do byte, que poderá então ser rejeitado. Se você refletir sobre isso, notará o seguinte: mesmo que o próprio bit de paridade venha a se adulterar na transmissão, a ocorrência do erro será detectada pelo processo de verificação de paridade, e o byte será rejeitado.

Ao observar o código ASCII utilizado por seu próprio computador, você notará que o bit 7 (o bit mais significativo, ou MSB, Most Significant Bit) é efetivamente utilizado, mas não como um bit de paridade. Isso ocorre a fim de possibilitar ao computador a recepção de um conjunto de caracteres adicionais (geralmente um conjunto de caracteres gráficos), e também porque os erros de transmissão de dados no *interior* do computador são muito raros. A paridade em geral é empregada apenas na transmissão de dados através de longas distâncias ou no registro de dados em superfícies magnéticas de gravação (como fita ou disco), que também é suscetível de "erros de bit".

A verificação de paridade é excelente para indicar a transmissão incorreta de um byte, mas não identifica qual dos bits foi incorretamente transmitido; assim, o erro não pode ser corrigido pelo equipamento receptor. Pior ainda, se dois bits em um byte forem adulterados, o byte incorretamente transmitido poderá ser tomado como correto.

Nos casos, porém, em que o erro é detectado pelo dispositivo receptor, este pode enviar uma mensagem de erro ao equipamento emissor e o software fará com que o byte seja transmitido novamente de modo correto. Processos mais sofisticados de detecção e correção de erros foram criados; localizam com precisão o bit ou bits incorretos, fazendo a correção automática. Os códigos de correção são um assunto a ser examinado posteriormente no curso.

**Apenas para verificação**
O último dígito do Padrão Internacional de Codificação de Livros (International Standard Book Number, ISBN) consiste em um dígito de verificação, equivalente à paridade em um computador. Multiplique o primeiro dígito (aqui, 0) por 10, o segundo (5), por 9, e assim sucessivamente, e a seguir some os resultados. Você notará que o dígito de verificação foi determinado de modo que, somado ao total, o resultado seja exatamente divisível por 11.



# Campos e registros

**Continuando o projeto de programação para desenvolvimento de uma agenda de endereços computadorizada, veremos agora como um arquivo pode ser subdividido em registros e campos.**

Na parte anterior do curso de programação em linguagem BASIC, deixamos como tarefa a elaboração dos elementos do exercício de programação através de um ou mais níveis de "pseudolinguagem", até o ponto em que os exemplos pudessem ser codificados em BASIC. Iniciaremos hoje pela revisão desse exercício e pela sugestão de algumas soluções possíveis. A primeira "Apresentação dos objetivos", por exemplo, foi:

**ENTRADA (INPUT)**
Um nome (em qualquer formato)
**SAÍDA (OUTPUT)**
1. Um prenome
2. Um sobrenome

Em nosso primeiro nível de elaboração, verificamos que estes objetivos poderiam ser subdivididos em seis estágios (posteriormente vimos que o último deles podia ser eliminado). Eram os seguintes:

1. Ler o nome (*LER*)
2. Converter todas as letras em maiúsculas (*CONVERTER*)
3. Encontrar o último espaço (*ESPAÇO*)
4. Ler o sobrenome (*LERSOBRENOME*)
5. Ler o prenome (*LERPRENOME*)
6. Eliminar os caracteres não alfabéticos do prenome

Estamos tratando todos esses procedimentos como sub-rotinas e os nomes a elas atribuídos são fornecidos entre parênteses. Infelizmente, a maioria das versões de BASIC não permite a chamada de sub-rotinas por nome e será necessário, quando o programa chegar ao estágio final de desenvolvimento, inserir números de linha após as instruções GOSUB correspondentes. Durante a fase de desenvolvimento, todavia, é muito mais fácil fazer referência a sub-rotinas por meio de nomes. Esses nomes poderão depois ser incorporados nas instruções REM. Estamos indicando essa utilização de sub-rotinas com nomes colocando-os entre asteriscos. Em linguagens que podem chamar sub-rotinas pelos nomes (como a PASCAL), esse tipo de sub-rotina é denominado "procedimento".

Apesar de a linguagem BASIC não poder operar com procedimentos, é interessante você supor que isso é possível, enquanto a programação ainda estiver no estágio de pseudolinguagem. De modo semelhante, sua versão de BASIC pode não ser adequada para lidar com nomes longos de variáveis, como CONTAR ou NOMEDARUA$, mas em nível de pseudolinguagem é mais conveniente designá-los desse modo. Tente fazer com que eles sejam descritivos. Fica muito mais claro chamar uma variável auxiliar, para uma série de caracteres, de VARIAVELAUX$ do que chamá-la de VX$. Felizmente, muitas das versões de BASIC agora permitem nomes mais longos de variáveis.

Também desenvolvemos o segundo dos estágios (converter todas as letras em maiúsculas) através de um segundo e terceiro níveis de elaboração e criamos um pequeno programa para executar esta tarefa. Agora, tentaremos o mesmo com relação aos outros estágios:

```
2.ª ELABORAÇÃO
3. (Encontrar o último espaço)
INICIAR
Executar um LOOP enquanto permanecerem
caracteres não examinados na variável NOME$
   IF (SE) o caractere = "  "
      THEN (ENTÃO) registrar a posição em uma
      variável
      ELSE (CASO CONTRÁRIO) nada executar
   ENDIF (ENCERRAR)
ENDLOOP (ENCERRAR O LOOP)
END (ENCERRAR)

3.ª ELABORAÇÃO
3. (Encontrar o último espaço)
INICIAR
READ (LER) NOMECOMPLETO$
Executar um LOOP (executado enquanto
permanecerem caracteres não examinados)
   FOR L=1 para a extensão de
   NOMECOMPLETO$
   READ (LER) o caractere proveniente de
   NOMECOMPLETO$
   IF (SE) o caractere = "  "
      THEN LET (ENTÃO) CONTAR = posição do
      caractere
      ELSE (CASO CONTRÁRIO) nada executar
   ENDIF (ENCERRAR)
ENDLOOP (ENCERRAR O LOOP)
END (ENCERRAR)
```

Estamos agora em condição de codificar, da pseudolinguagem para a linguagem de programação:

```
10 INPUT "FORNECER NOME COMPLETO ";
   NOMECOMPLETO$
20 FOR L=1 TO LEN (NOMECOMPLETO$)
30 LET CARACT$ = MID$
   (NOMECOMPLETO$,L,1)
40 IF CARACT$ = "  " THEN LET CONTAR = L
50 NEXT L
60 PRINT "O ULTIMO ESPACO ESTA NA
   POSICAO ";CONTAR
70 END
```

Observe que a linha 10 é uma entrada fictícia para testar a rotina; a linha 60 é uma saída fictícia, também para teste; e a instrução da linha 70 deverá ser

alterada para RETURN, quando a rotina for utilizada como sub-rotina.

Vamos tentar agora o mesmo procedimento no estágio quatro:

```
2.ª ELABORAÇÃO
4. (Ler sobrenome)
INICIAR
Atribuir a SOBRENOME$ os caracteres à direita do
último espaço
END (ENCERRAR)

3.ª ELABORAÇÃO
4. (Ler sobrenome)
INICIAR
   READ (LER) NOMECOMPLETO$
   Localizar o último espaço (chamar a sub-rotina
   *ESPAÇO*)
   Executar o LOOP enquanto os caracteres
   permanecerem na série, após o espaço
      READ (LER) os caracteres e acrescentá-los a
      SOBRENOME$
   ENDLOOP (ENCERRAR O LOOP)
END (ENCERRAR)
```

Antes de iniciar a codificação em BASIC, você deve observar alguns enganos possíveis. Ao localizar o último espaço, na última elaboração acima apresentada, a pseudolinguagem exige a utilização da sub-rotina *ESPAÇO*, mas não será possível escrevê-la em BASIC e testá-la, se a sub-rotina *ESPAÇO* já não tiver sido escrita. De modo geral, não é interessante codificar cada módulo em BASIC (ou qualquer outra linguagem de alto nível) antes de o programa ser desenvolvido em pseudolinguagem. Todavia, se você deseja testar um módulo, talvez seja necessário escrever alguns valores fictícios de variáveis, bem como algumas entradas e saídas fictícias. No exemplo acima, CONTAR é a variável que contém o número da posição do último nome, na variável NOMECOMPLETO$. Ao testar, podemos trapacear um pouco com a suposição de que a rotina funciona de modo adequado:

```
10 LET NOMECOMPLETO$ = "TOM JOBIM"
20 LET CONTAR = 4
30 FOR L = CONTAR + 1 TO LEN
   (NOMECOMPLETO$)
40 LET SOBRENOME$ = SOBRENOME$ + MID$
   (NOMECOMPLETO$,L,1)
50 NEXT L
60 PRINT "O SOBRENOME E "; SOBRENOME$
70 END
```

Segue-se agora o processo para encontrar o prenome (estágio cinco). Lembre-se de que definimos o prenome como uma concatenação de todos os caracteres alfabéticos, até o último espaço do nome. Sinais de ponto, apóstrofos, espaços etc. tiveram de ser eliminados.

```
2.ª ELABORAÇÃO
5. (Ler prenome)
INICIAR
Executar o LOOP até o último espaço, enquanto
os caracteres permanecerem em
NOMECOMPLETO$
   Examinar os caracteres
   IF (SE) o caractere não for uma letra
      THEN (ENTÃO) nada executar
      ELSE (CASO CONTRÁRIO) acrescentar
      o caractere a PRENOME$
   ENDIF (ENCERRAR)
ENDLOOP (ENCERRAR O LOOP)
END (ENCERRAR)

3.ª ELABORAÇÃO
5. (Ler prenome)
INICIAR
Executar o LOOP até CONTAR, enquanto
permanecerem os caracteres
   LET CARACTAUX$ = elésimo caractere na série
   IF (SE) CARACTAUX$ não for uma letra
      THEN (ENTÃO) nada executar
      ELSE (CASO CONTRÁRIO) LET PRENOME$ =
      PRENOME$ + CARACTAUX$
   ENDIF (ENCERRAR)
ENDLOOP (ENCERRAR O LOOP)
```

Agora temos condição de codificar em BASIC, mas, como estágio intermediário, utilizaremos instruções BASIC, não numeradas, em um formato estruturado, de modo que você possa comparar a estrutura com o estágio acima:

```
CODIFICAÇÃO
5. (Ler prenome)
REM INICIAR
REM LOOP
   FOR L = 1 TO CONTAR - 1
      LET CARACTAUX$ = MID$
      (NOMECOMPLETO$,L,1)
      LET CARACT = ASC(CARACTAUX$)
      IF CARACT>64 THEN PRENOME$ =
         PRENOME$ + CHR$(CARACT)
      REM ENDIF
   NEXT L: REM ENDLOOP
REM END
```

Em linguagem BASIC, isto seria:

```
10 FOR L = 1 TO CONTAR - 1
20 LET CARACTAUX$ =
   MID$(NOMECOMPLETO$,L,1)
30 LET CARACT = ASC(CARACTAUX$)
40 IF CARACT>64 THEN PRENOME$ = PRENOME$
   + CHR$(CARACT)
50 NEXT L
60 END
```

Entretanto, do modo como se apresenta, o programa não funcionará. Há três problemas com ele: a variável CONTAR precisa receber um valor; não está prevista a entrada dos nomes (atribuição de uma série de caracteres à variável NOMECOMPLETO$); e não há "saída" na forma de uma instrução para impressão, de modo que o usuário possa verificar se o programa funcionou de forma adequada.

Se essa rotina fosse parte de uma sub-rotina, os parâmetros a ela transferidos (a entrada) e os parâmetros dela provenientes (a saída) teriam de ser manipulados em outra parte do programa. Esta é uma questão importante: o fluxo dos dados no interior do programa deve ser sempre avaliado com cuidado antes do início da codificação em BASIC. Isso é especialmente importante quando utilizamos variáveis (CONTAR, por exemplo) e o mesmo nome de variá-

vel é usado em diversas partes do programa. Não adianta chamar uma sub-rotina que utiliza uma variável como CONTAR, se a sub-rotina não possui meios de conhecer o valor que ela supostamente possui. Se uma sub-rotina atribui um valor inicial à variável CONTAR, esse valor permanecerá o mesmo, a menos que um novo valor seja atribuído posteriormente — talvez em outra sub-rotina. Este é o motivo por que não é boa prática de programação saltar do meio de um loop, uma vez que o valor da variável no loop será desconhecido. Examine as conseqüências de deixar esses dois fragmentos de programa como partes de diferentes sub-rotinas em um programa:

**Parte da sub-rotina X**

```
FOR L = 1 TO LEN(PALAVRA$)
LET CARACT$ = MID$(PALAVRA$,L,1)
IF CARACT$ = " . " THEN GOTO 1550
NEXT L
```

**Parte da sub-rotina Y**

```
FOR Q=1 TO LIMITE
LET A(L) = P(Q)
NEXT Q
```

Esta parte da sub-rotina Y lê valores em uma tabela indexada, na qual o índice é a variável L. Se a sub-rotina Y for chamada após a sub-rotina X e se a condição de teste na sub-rotina X tiver sido obtida (que um dos caracteres seja um " . "), o valor da variável L será completamente imprevisível; assim, não saberemos a que elementos da tabela os valores estarão sendo atribuídos na sub-rotina Y. Além do erro de desviar para fora do loop, esta sub-rotina também utiliza uma instrução GOTO, procedimento que também é desaconselhável. Instruções GOTO dificultam muitas vezes a compreensão do programa e devem ser utilizadas com cuidado.

A fim de evitar confusões, um bom procedimento é fazer uma lista de todas as variáveis, nos estágios de pseudolinguagem, acompanhada de observações que informem para que estão sendo utilizadas. Algumas linguagens (mas não o BASIC) permitem a apresentação das variáveis como "locais" ou "globais", isto é, possuem valores que são utilizados em apenas uma parte do programa (locais) ou através de todo o programa (globais). Muitas variáveis, como as utilizadas nos loops (por exemplo, o L, na instrução LET L = 1 TO 10), são quase sempre locais; desse modo, é geralmente conveniente dar um valor inicial à variável, antes de sua utilização (por exemplo, LET L = 0). Algumas linguagens, como PASCAL, exigem isto; e, embora o BASIC sempre assuma que o valor inicial da variável é 0 (a menos que um outro seja especificado), ainda assim é recomendável fornecer o valor inicial.

Até aqui formulamos uma definição razoável de um nome, para as finalidades de nossa agenda de endereços computadorizada, e desenvolvemos algumas rotinas que podem lidar com nomes de vários modos, e que serão utilizadas em nosso programa completo. Agora vamos novamente nos afastar dos detalhes de codificação do programa e examinar a estrutura dos "registros" em nosso "arquivo" da agenda de endereços.

Os termos "registro", "arquivo" e "campo" têm significado relativamente específico na área de computação. *Arquivos* são conjuntos completos de dados relacionados. Em sistemas de computação, correspondem aos itens identificáveis armazenados em disco flexível (disquete) ou em fita cassete, com um nome determinado. Podemos considerar toda a nossa agenda de endereços como um arquivo e a denominaremos AGEEND.

No interior dos arquivos, temos *registros*, que são também conjuntos de dados relacionados. Se imaginarmos nossa agenda de endereços como uma caixa de fichário, o arquivo será a caixa repleta de cartões e os registros serão os cartões individuais — cada qual com seu próprio nome, endereço completo e telefone.

Em cada registro temos *campos*. Os campos podem ser considerados como uma ou mais linhas de dados relacionados, no interior do registro. Cada um dos registros em nosso arquivo AGEEND terá os seguintes campos: NOME, ENDEREÇO e TELEFONE. Um exemplo de registro é o seguinte:

```
Pedro Lameira
Rua Antonio Celso 73
São Paulo
SP
011-5402588
```

Neste registro há três campos: o campo do nome, que inclui as letras do alfabeto (e, talvez, o apóstrofo, como Pedro d'Ávila); o campo do endereço, que inclui alguns números e muitas letras; e o campo do telefone, que inclui apenas números, desprezando-se os hifens, como neste exemplo: 011-258-1191). Antes de começar a escrever um programa que lide, de maneira flexível, com informação complexa como esta, precisamos determinar o modo de representar os dados no interior do computador. Uma maneira pode ser considerar toda a informação no interior de um registro como sendo simplesmente uma longa série de caracteres. O problema com este tipo de procedimento é que a obtenção de dados específicos é extremamente difícil. Vamos admitir que a seguinte entrada é apenas uma longa série de caracteres:

```
PERCIVAL R. BURTON
1056 AVENUE OF THE AMERICAS
RIO DEL MONTENEGRO
CALIFORNIA
U.S.A.
(415) 884 5100
```

Se percorrermos os registros para encontrar o número telefônico de PERCIVAL R. BURTON, será seguro supor que os 14 últimos caracteres do registro representam o número? O que acontecerá se incluirmos o código de discagem direta internacional, da seguinte forma: 0101 (415) 884 5100? O número teria então um total de 19 caracteres. Para superar esta dificuldade, é atribuído ao número telefônico um campo separado, e o programa nos fornecerá todos os caracteres (ou números) nesse campo, quando for solicitado.

A dificuldade com este tipo de procedimento está na necessidade de algum meio para relacionar os vários campos separados, de modo que a referência a um campo (o campo do nome, por exemplo) possa igualmente nos fornecer os outros campos do registro. Um meio de resolver isso consiste em possuir

um campo complementar vinculado ao registro apenas para finalidades de indexação. Se um registro for, por exemplo, o 15.º do arquivo, o campo para o índice deverá conter o número 15. Isto então poderia ser utilizado para indicar os elementos em uma série de tabelas. Para ilustrá-lo, suponha que um registro se apresente da seguinte forma:

| | |
|---|---|
| Jaime Fonseca | campo do NOME |
| Rua Feliciano 59 | campo do ENDEREÇO |
| Sorocaba | campo da CIDADE |
| SP | campo do ESTADO |
| 0152322303 | campo do TELEFONE |
| 015 | campo do ÍNDICE |

Se soubermos o nome da pessoa e desejarmos o número de seu telefone, teremos apenas de percorrer os elementos da tabela, mantendo os nomes, até que seja encontrado o correspondente. Procuraremos, a seguir, aquele elemento da tabela no qual está o nome — neste caso, o número 15. Assim, tudo que nos resta fazer será encontrar o 15.º elemento da tabela, TELEFONE, para obter o número correto do telefone.

Se tivermos amigos na região Nordeste do país, poderemos querer que o programa detecte cada ocorrência do nome "Recife" no campo CIDADE. O programa examina os campos CIDADE e registra a posição de cada ocorrência do nome Recife. Assim, para a impressão dos nomes e endereços desses amigos, será necessário unicamente recuperar todos os elementos que têm o mesmo número em todas as tabelas para cada registro "Recife". A utilização deste procedimento elimina a necessidade de pesquisar o campo ÍNDICE, e a técnica apresenta a vantagem de ser uma operação relativamente simples.

Na próxima parte do nosso curso, veremos alguns problemas vinculados à pesquisa de listas para encontrar itens específicos.

## Exercícios

■ Admitir que os registros com os seguintes campos serão adequados à nossa agenda de endereços computadorizada:

campo do NOME
campo do ENDEREÇO
campo da CIDADE
campo do ESTADO
campo do TELEFONE

Supor que uma das opções oferecidas por um menu na agenda de endereços é a seguinte:

5. CRIAR UMA NOVA ENTRADA

Você digita o número 5 e o programa desvia para a parte em que novos registros são criados (você pode admitir que ainda não há entradas na agenda de endereços). Uma vez que o programa é completamente dirigido por menu, você estará sempre sendo advertido para as entradas previstas — com indicações como FORNECER O NOME, FORNECER A RUA e assim sucessivamente. Eis aqui uma lista dos resultados esperados.

1. Um elemento em uma tabela para o nome
2. Um elemento em uma tabela para o endereço
3. Um elemento em uma tabela para a cidade
4. Um elemento em uma tabela para o estado
5. Um elemento em uma tabela para o telefone

Sua tarefa é desenvolver estes procedimentos, por um processo de programação top-down e usando uma pseudolinguagem, até o ponto em que a conversão direta para o BASIC se torne possível. A pseudolinguagem pode seguir as regras que você estabelecer; sugerimos apenas o emprego de letras maiúsculas para palavras-chaves, como IF, LOOP etc., e de letras minúsculas para descrições em linguagem comum das operações a serem efetuadas.

## A propósito...

Apresentaremos, a partir de agora, uma relação dos principais comandos, instruções e funções do BASIC. Serão descritas instruções de quatro versões dessa linguagem: o BASIC da Microsoft, muito utilizado em microcomputadores comerciais; a versão Applesoft, encontrada nos micros compatíveis com o Apple II (Micro Engenho, Unitron, TK2000, Elppa etc.); a versão TRS-80, disponível em micros como CP 300, CP 500, DGT 1000 e outros; e a versão Sinclair ZX81, compatível com as linguagens dos micros TK83, TK85, CP 200, Ringo e outros.

| Comando/Instrução/Função | Ação/Resultado | MS BASIC | Applesoft | TRS-80 | Sinclair |
|---|---|---|---|---|---|
| ABS(x) | Dá o valor absoluto de x | + | + | + | + |
| ACS(x) | Dá o arco-cosseno em radianos de x | | | | + |
| APPEND "arquivo" | Abre um arquivo e posiciona o pointer no final do mesmo | | + | | |
| ASC(x$) | Dá o código ASCII do primeiro caractere de x$ | + | + | + | |
| ASN(x) | Dá o arco-seno em radianos de x | | | | + |
| ATN(x) | Dá o arctangente (em radianos) de x | + | + | + | + |
| AUTO número, incremento | Gera número de linhas automaticamente | + | | + | |
| BEEP | Executa um bip no alto-falante | + | | | |
| BLOAD "arquivo", saída | Carrega dados binários na memória | + | + | | |
| BSAVE "arquivo", saída | Grava dados binários | + | + | | |
| CALL variável | Chama um programa em linguagem de máquina | + | | | |
| CDBL(x) | Converte x para precisão dupla | + | | + | |
| CHAIN "arquivo" | Chama um programa e lhe transfere variáveis | + | | | |

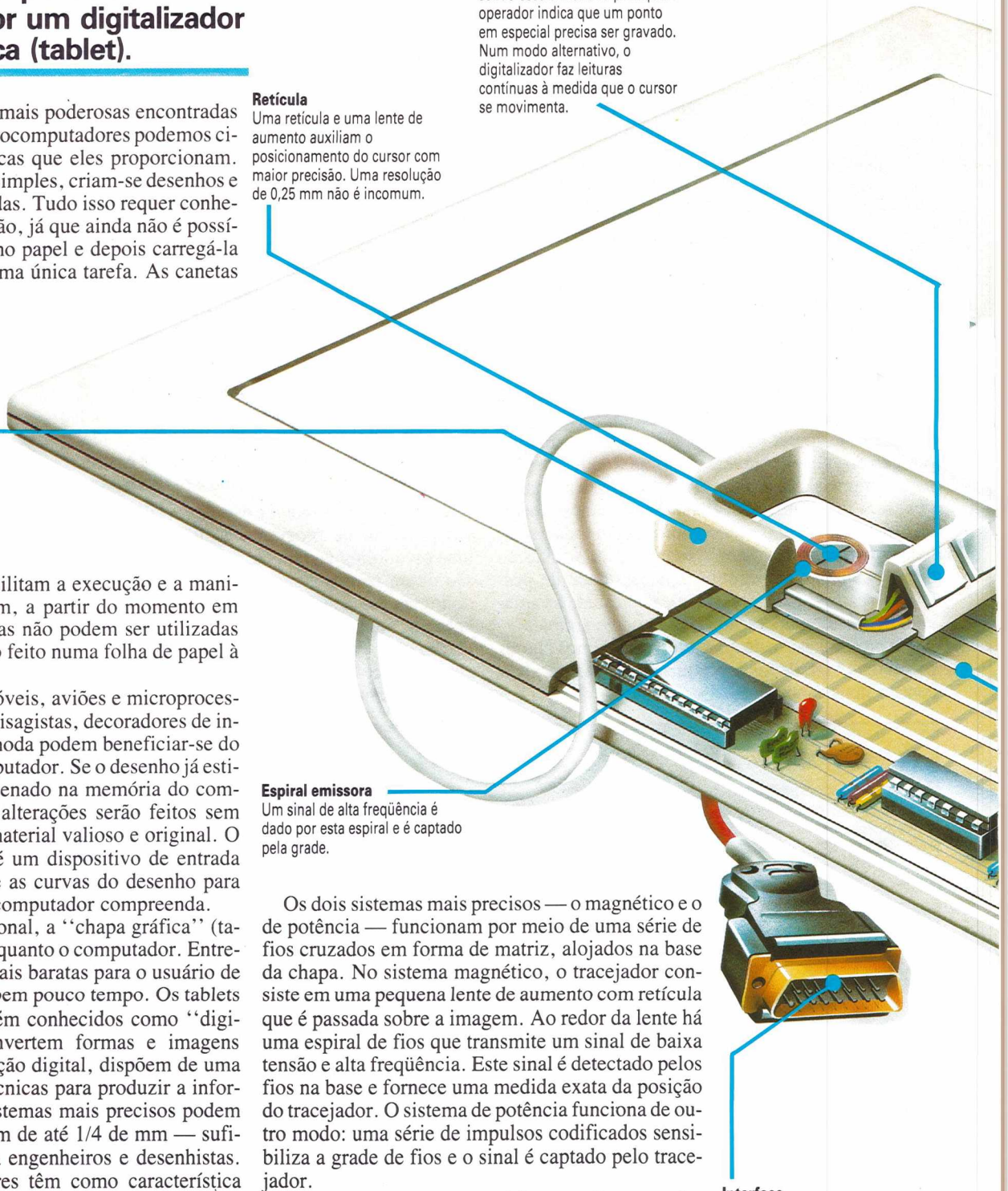
# Traços eletrônicos

## Imagens desenhadas no papel são transferidas para seu computador por um digitalizador ou chapa gráfica (tablet).

Entre as características mais poderosas encontradas na geração atual de microcomputadores podemos citar as habilidades gráficas que eles proporcionam. Com alguns comandos simples, criam-se desenhos e padrões em cores variadas. Tudo isso requer conhecimentos de programação, já que ainda não é possível criar uma imagem no papel e depois carregá-la no computador como uma única tarefa. As canetas ópticas (ver p. 156) facilitam a execução e a manipulação de uma imagem, a partir do momento em que ela está na tela, mas não podem ser utilizadas para copiar um desenho feito numa folha de papel à parte.

Projetistas de automóveis, aviões e microprocessadores, assim como paisagistas, decoradores de interiores e estilistas de moda podem beneficiar-se do sistema gráfico do computador. Se o desenho já estiver seguramente armazenado na memória do computador, acréscimos e alterações serão feitos sem que se perca nenhum material valioso e original. O que se faz necessário é um dispositivo de entrada que traduza as linhas e as curvas do desenho para uma linguagem que o computador compreenda.

No mercado profissional, a "chapa gráfica" (tablet) é quase tão antiga quanto o computador. Entretanto, as alternativas mais baratas para o usuário de micros apareceram há bem pouco tempo. Os tablets de alta precisão, também conhecidos como "digitalizadores", pois convertem formas e imagens analógicas em informação digital, dispõem de uma grande variedade de técnicas para produzir a informação desejada. Os sistemas mais precisos podem determinar uma imagem de até 1/4 de mm — suficientemente exata para engenheiros e desenhistas. Todos os digitalizadores têm como característica uma superfície de base, onde é posta a imagem desenhada ou pintada. Um tracejador, que pode ser uma caneta comum ou um dispositivo eletrônico sofisticado, percorre a superfície da imagem. A posição do tracejador é detectada pelo digitalizador e transmitida como um par de coordenadas mutáveis para o computador.



**Botões de entrada de dados**
A maioria dos cursores possui mais de um botão de pressão, sendo essa a maneira pela qual o operador indica que um ponto em especial precisa ser gravado. Num modo alternativo, o digitalizador faz leituras contínuas à medida que o cursor se movimenta.

**Retícula**
Uma retícula e uma lente de aumento auxiliam o posicionamento do cursor com maior precisão. Uma resolução de 0,25 mm não é incomum.

**Cursor**
Este dispositivo é movido manualmente para traçar a imagem que está sendo digitalizada.

**Espiral emissora**
Um sinal de alta freqüência é dado por esta espiral e é captado pela grade.

Os dois sistemas mais precisos — o magnético e o de potência — funcionam por meio de uma série de fios cruzados em forma de matriz, alojados na base da chapa. No sistema magnético, o tracejador consiste em uma pequena lente de aumento com retícula que é passada sobre a imagem. Ao redor da lente há uma espiral de fios que transmite um sinal de baixa tensão e alta freqüência. Este sinal é detectado pelos fios na base e fornece uma medida exata da posição do tracejador. O sistema de potência funciona de outro modo: uma série de impulsos codificados sensibiliza a grade de fios e o sinal é captado pelo tracejador.

Uma alternativa a estes dois procedimentos é o sistema acústico. O tracejador recebe uma carga eletrostática e, no momento em que toca a base, emite uma pequena faísca. O tempo que a onda acústica criada pela faísca leva para atingir dois microfones indica a posição do tracejador. Entre outras coisas, este processo oferece a possibilidade de se digitali-

**Interface**
Os digitalizadores são geralmente ligados ao computador através de uma porta serial padrão ou paralela.

**Base**
A imagem a ser digitalizada é colocada nesta base. Em alguns sistemas, uma carga eletrostática é aplicada a esta base para "colar" temporariamente a folha de papel. É importante que a imagem não se mova em relação à base.

zar em três dimensões, por meio de um sinal que passa pelo objeto.

Na parte inferior da escala, há uma chapa sensível à pressão: a imagem é colocada sobre ela e a seguir é desenhada com o tracejador. Isso exige maior pressão que os outros sistemas. Duas folhas condutoras de eletricidade são separadas por um isolador celular, e dois diferentes sinais de alta freqüência são alimentados nas camadas. O sinal detectado pelo tracejador, no instante em que ele faz a conexão elétrica entre as duas folhas, informa a sua posição. Os problemas típicos encontrados neste tipo de sistema incluem mudanças na resistência da superfície, devido a desgaste ou a diferentes pressões manuais. Dado o limitado nível de resolução de gráficos dos microcomputadores, a precisão deste método pode ser considerada aceitável.



**Uso de mapas**
Um dos usos profissionais mais conhecidos de digitalizadores é a coleta e análise de dados de mapas e pesquisas. Aqui, o computador está sendo utilizado para descobrir a localização de novos campos de petróleo a partir de dados geológicos digitalizados.

**Circuito de processamento**
Este PCB contém um microprocessador, alguma memória ROM e alguma RAM. Essas são as condições necessárias ao dispositivo para que ele possa apresentar ao computador informações em forma de coordenadas X-Y.

**Grade receptora**
Na chapa da base existe uma grade de fios que pode captar os sinais emitidos pela espiral. O espaço da grade é consideravelmente menos preciso que a resolução do digitalizador, já que o circuito de processamento pode fazer uma interpolação da distância relativa do sinal captado pelos fios adjacentes.

Os digitalizadores mais simples e baratos são os pantógrafos, baseados no tradicional sistema de desenho através de braços interligados. Eles usam coordenadas geométricas para fornecer uma medida direta da posição do tracejador. Resistências variáveis montadas nas duas articulações fornecem voltagens proporcionais aos ângulos de abertura dos braços do pantógrafo. O nível de resolução do pantógrafo é limitado pela precisão das resistências variáveis e das juntas mecânicas, e situa-se em torno de 5%. No entanto, pantógrafos mais sofisticados, baseados em medidas ópticas da rotação das juntas, podem fornecer resultados muito melhores, apesar de ainda inferiores aos sistemas magnético e de potência.

Os tablets ópticos utilizam uma grade de raios infravermelhos para detectar o posicionamento do tracejador. Eles não se mostram tão sensíveis como os outros sistemas, mas são inteiramente adequados a que se utilize o próprio dedo para selecionar um item de um "menu" do programa. Em alguns casos, as fontes de infravermelho e os detectores são colocados nas bordas da unidade de apresentação visual, fornecendo assim uma tela interativa onde as imagens podem ser desenhadas com o simples movimento de um dedo.

Os dados reais produzidos pelo tablet ou digitalizador devem ser convertidos em informações adequadas para projeção na tela e, para isso, a maioria dos equipamentos disponíveis é equipada com o software adequado. Entretanto, a simples entrada de dados não representa o único uso dos tablets. Desde que a informação esteja armazenada no computador, o tablet pode ser utilizado como um instrumento de elaboração, permitindo que novas cores sejam acrescidas ou modificadas, e que se tracem as formas de outra maneira. Programa-se a superfície do tablet para atuar como um menu de opções padronizadas, e assim o teclado é utilizado apenas para selecionar as principais funções. Todos os sistemas de animação por computadores (ver p. 181) possuem tablets de excelente qualidade como forma principal de entrada.

# Gottfried Leibniz

**1646**
Nasce em julho, em Leipzig.

**1661**
Matricula-se na Universidade de Leipzig. Forma-se aos 17 anos.

**Década de 1660**
Desenvolve o princípio da razão suficiente, em Paris.

**1673**
Máquina de calcular apresentada à Royal Society na Inglaterra.

**1675**
Inventa o cálculo infinitesimal independentemente de Newton.

**1676**
Estuda a dinâmica através da energia cinética.

**1678**
Bibliotecário e conselheiro do duque de Hanover.

**1679**
Desenvolve a matemática binária.

**1683**
Publica o panfleto "O mais cristão senhor da guerra", em ataque a Luís XIV.

**Década de 1690**
Sua genealogia da Casa de Hanover expande-se para o estudo da história do mundo. Desenvolve interesse pela lingüística e pela origem das línguas.

**1700**
Organiza a Academia de Ciências de Berlim.

**1714**
Responsável pelo estabelecimento do direito de sucessão de Jorge I ao trono inglês após a morte da rainha Ana.

**1716**
Morre em Hanover a 14 de novembro.



## Cientistas envolvidos com a quinta geração de computadores interessam-se pelo trabalho deste pensador do século XVII.

Gottfried Wilhelm Leibniz foi uma das luzes científicas de sua época — o período conhecido como a Idade da Razão. Nasceu na cidade de Leipzig em 1646 e morreu em Hanover em 1716. Durante seus setenta anos de vida (o tipo de número exato que você espera de um matemático) descobriu princípios de cálculos, estudou a dinâmica dos corpos e fez contribuições para a geologia, a teologia, a história, a lingüística e a filosofia. Além do mais, desenvolveu idéias que viriam a ser fundamentais para a criação do computador.

Iniciou suas viagens aos 20 anos, depois de a Universidade de Leipzig ter-lhe recusado o doutorado em direito, por causa de sua pouca idade. Não dispondo de recursos particulares para sustentar-se, teve de submeter-se a trabalhos que não tinham relação com suas pesquisas científicas. Aos vinte e poucos anos, trabalhou como advogado e diplomata; mais tarde, passou a ser bibliotecário e conselheiro da realeza.

Suas pesquisas eram amplamente diversas, e sua natureza cosmopolita levou-o a extensas viagens pela Europa, o que lhe possibilitou travar amizade com os grandes pensadores da época.

Sua primeira contribuição importante para a filosofia ocorreu em 1672, quando formulou o princípio da razão suficiente. Esse princípio sustentava a idéia de que "nada existe sem uma razão de ser" e "todas as coisas estão para o melhor no melhor possível dos mundos".

Voltando-se para a matemática, lançou-se ao trabalho de aperfeiçoamento da máquina de somar pascalina, inventada por Blaise Pascal, em 1642 (ver p. 86). Leibniz procurou torná-la também capaz de multiplicar e dividir. Para isso, projetou um dispositivo mecânico chamado cilindro de Leibniz (ver ilustração abaixo). O aparelho foi uma grande novidade para a época.

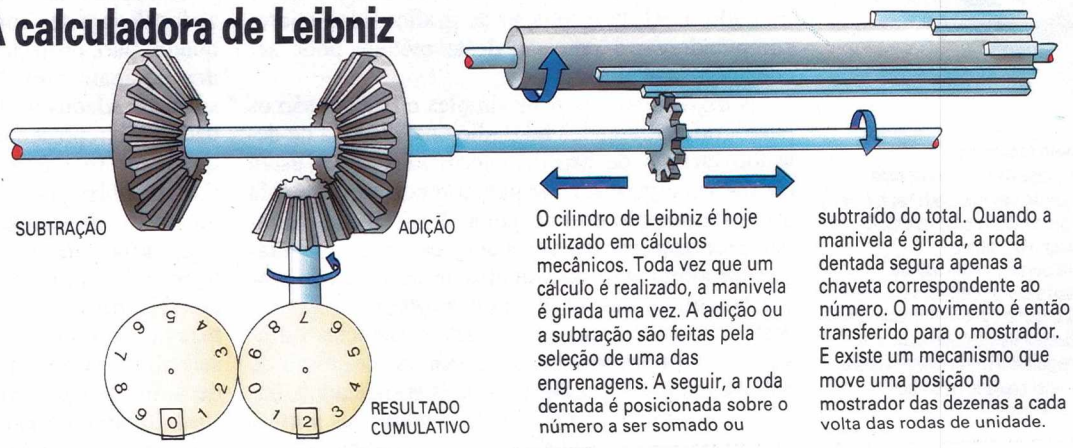
Anteriormente, pela complexidade de operação dos numerais romanos, a multiplicação só era ensinada nas escolas especializadas. Uma máquina que podia multiplicar mecanicamente tornava a aritmética mais acessível. Uma vez aperfeiçoado esse aparelho, Leibniz passou da base aritmética 10 para o exame formal da matemática binária.

A maior ambição de Leibniz era idealizar uma linguagem universal que utilizasse a clareza e a precisão da matemática para solucionar os problemas enfrentados pela humanidade. Essa linguagem caracterizava-se pelo uso de símbolos abstratos para representar os "átomos" fundamentais da compreensão, incluindo também um conjunto de normas para o emprego desses símbolos.

Sua tentativa não obteve resultado satisfatório, mas suas idéias foram retomadas de maneira mais modesta, no início do século XX, por Bertrand Russell, que procurou explicar a matemática em termos de uma "linguagem" de lógica formal.

Nos anos recentes, cientistas envolvidos no projeto a longo prazo de desenvolvimento da quinta geração de computadores voltam a demonstrar interesse pelo trabalho de Leibniz. Espera-se que estas máquinas possam solucionar qualquer problema da atividade humana, com a mesma velocidade e correção com que os computadores atuais executam cálculos matemáticos. Para isso, eles vão precisar de um tipo de linguagem totalmente novo.

## A calculadora de Leibniz




O cilindro de Leibniz é hoje utilizado em cálculos mecânicos. Toda vez que um cálculo é realizado, a manivela é girada uma vez. A adição ou a subtração são feitas pela seleção de uma das engrenagens. A seguir, a roda dentada é posicionada sobre o número a ser somado ou subtraído do total. Quando a manivela é girada, a roda dentada segura apenas a chaveta correspondente ao número. O movimento é então transferido para o mostrador. E existe um mecanismo que move uma posição no mostrador das dezenas a cada volta das rodas de unidade.

# Um livro de figuras

**O Lisa, da Apple, reproduz em sua tela os objetos usualmente dispostos sobre uma mesa de escritório. Mas muitas de suas funções estão passando para os micros domésticos.**

O Lisa, produzido pela Apple Computer, é uma máquina de uso exclusivamente comercial. Seu preço, mesmo sem impressora, é elevado. Você, então, obviamente, estará indagando o que essa máquina tem a ver com o MICROCOMPUTADOR — CURSO BÁSICO. A resposta é simples: decidimos dar tamanha atenção ao Lisa porque se trata de um produto pioneiro e várias de suas funções começam a ser incorporadas aos micros não-comerciais. A própria Apple já lançou um equipamento menor e mais barato com as mesmas características — o Macintosh. E os concorrentes se preparam para competir com a capacidade de desempenho desse produto.

A novidade em torno do Lisa não é o seu hardware e sim o software padronizado que o acompanha. O desenvolvimento de um software sofisticado vem sendo uma tendência generalizada de todos os fabricantes de micros. Hoje são despendidos menos homens-hora para projetar e construir uma nova máquina do que para escrever um complexo jogo eletrônico ou um programa de uso comercial. O software tornou-se o elemento mais importante de qualquer sistema de computador e também o mais caro. Usuários de microcomputadores constatam que é possível gastar, por ano, com jogos, programas aplicativos e utilitários o mesmo dinheiro que pagaram pela máquina.

Mesmo assim, analisaremos primeiro o hardware do Lisa, cujo design foi fundamentalmente ditado pelas exigências do software. A memória-padrão do Lisa é de 1 megabyte de RAM (isto é, mil vezes a memória-padrão de um Sinclair ZX81). Uma memória tão grande exige que o microprocessador gaste muito tempo com "administração de memória" — movimentando os dados e controlando os lugares em que eles estão armazenados. O processador é um Motorola 68000, capaz de processar 16 bits de informação de cada vez, ao passo que as CPUs da maioria dos microcomputadores usuais processam 8. Pelos padrões dos microcomputadores, é um processador muito rápido, com um conjunto de instruções realmente avançado. Para armazenamento permanente, o sistema Lisa inclui duas unidades de disco flexível e uma unidade de disco rígido independente e com poucas funções externas. O disco rígido é necessário tanto pela sua capacidade (5 megabytes) quanto pela velocidade — o Lisa faz uso de um grande número de programas que freqüentemente precisam ser trocados entre a RAM e o disco. Discos rígidos serão estudados com maior profundidade neste curso, pois já estão aparecendo unidades de baixo custo no mercado de microcomputadores.

Outra característica marcante do hardware do Lisa é o display monocromático embutido no mesmo módulo da CPU, que tem uma resolução de 720 x 364 pixels. Ele proporciona uma variedade de tipos diferentes para elaboração de textos, além dos gráficos descritos mais adiante neste artigo. O Lisa dispõe de chips e circuitos especiais com a função específica de dirigir esse display e mover rapidamente as imagens.

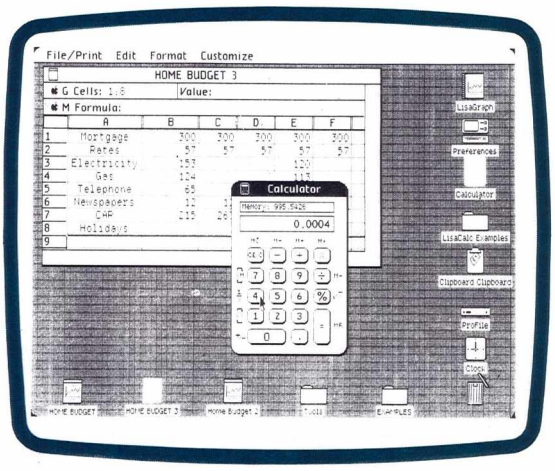


**Fácil para o usuário**
O Lisa foi projetado para ser usado pelos que trabalham em escritório e não têm experiência com computadores. O manejo do "mouse" permite que o teclado seja usado com menos freqüência do que em outros sistemas.

Ligado a uma impressora apropriada — do tipo matriz de pontos, de alta qualidade e velocidade — é possível reproduzir no papel qualquer coisa exibida na tela. Se a impressora não for compatível com o alto nível de resolução da tela, o Lisa produzirá apenas uma imagem impressa, de qualidade razoável.

O teclado do Lisa é separado da unidade principal e tem um bom layout. Contudo, é menos usado que o de outras máquinas, porque o Lisa possui um "mouse". Um mouse é uma das muitas alternativas para introduzir informações na tela sem usar o teclado — entre outros métodos, incluem-se joysticks, canetas ópticas e unidades de reconhecimento de voz. Do tamanho de um maço de cigarros, o mouse, ligado ao computador por um cabo de conexão, é uma caixa que movimentamos manualmente sobre a

**A vantagem do "mouse"**
Toda função é representada por um símbolo chamado "ícone" (figura). Para executá-la, o mouse é movido até o cursor ficar sobre a figura escolhida e aí o botão SELECT do mouse é apertado. Isto "abre" a aplicação, que é exibida com detalhes na tela do Lisa.




superfície de uma mesa. Movendo o mouse, impulsionamos um "cursor", ou ponteiro, que "anda" pela tela. Deste modo, apontando o cursor para a informação ou comando que necessitarmos, apertamos o botão SELECT existente no mouse e a informação será selecionada de imediato ou o comando executado. O teclado só é usado quando novos dados em forma de texto ou de números tiverem de ser introduzidos no computador.

Neste ponto, podemos analisar o software do Lisa e novamente ressaltar que, embora as aplicações usuais dessa máquina se concentrem totalmente na área comercial, os princípios sob os quais elas operam acabarão, sem dúvida, sendo transmitidos para as aplicações dos microcomputadores pessoais (menores) e domésticos.

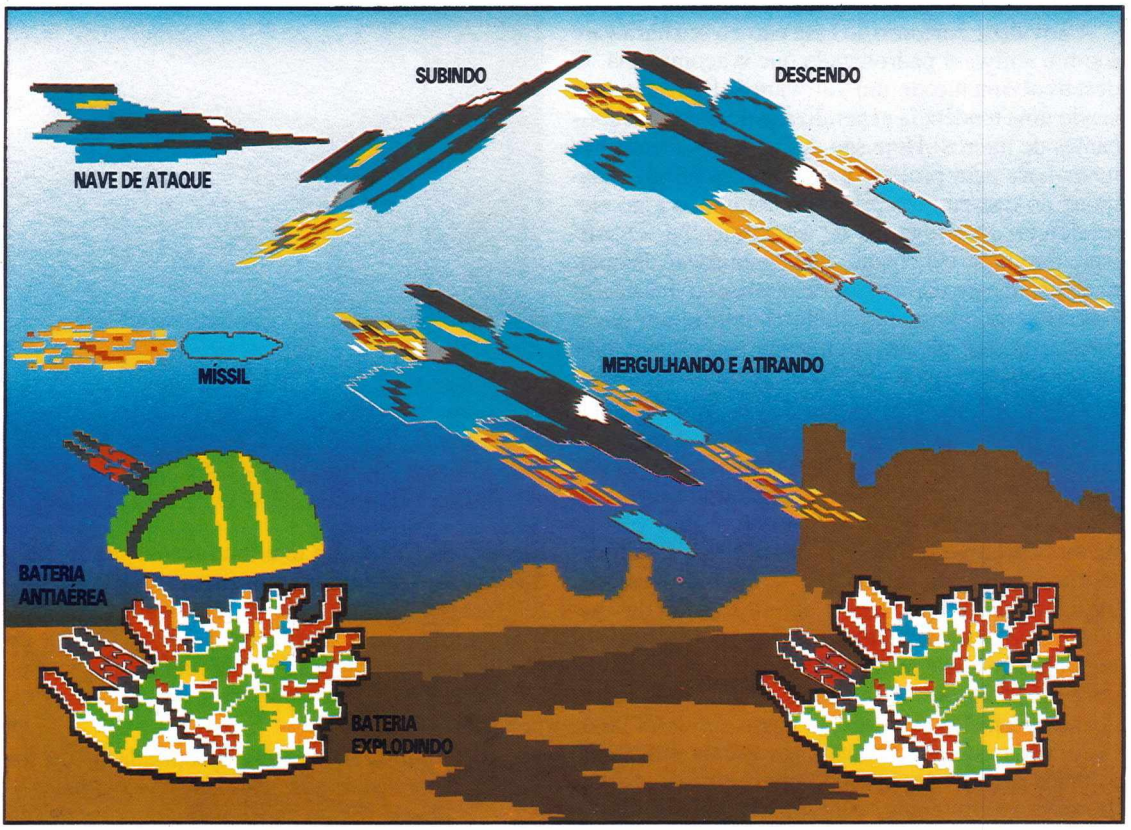
Quando você liga a máquina, a imagem que aparece no Lisa representa o tampo de uma mesa de trabalho com diversos objetos dispostos sobre ela. De fato, quase tudo que você faz no Lisa aparece representado na tela como se você estivesse trabalhando sem o auxílio do computador. Esta é a principal razão de os principiantes acharem tão mais fácil enfrentar o Lisa do que o hardware e o software convencionais. Cada um dos objetos arrumados sobre a mesa é chamado de "ícone" (figura) e representa uma função específica, geralmente assinalada abaixo.

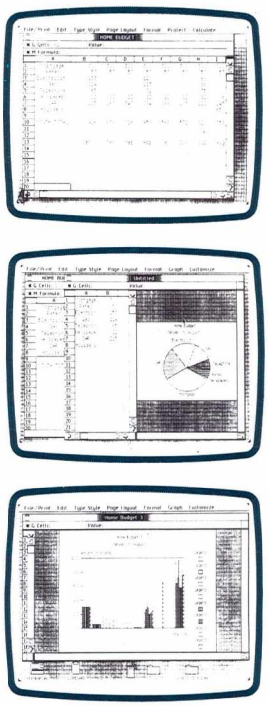
Tomemos como exemplo a figura do relógio. Movendo o cursor sobre o pequeno relógio por intermédio do mouse e apertando o botão SELECT, aparecerá um relógio maior na tela, junto com a data. Se você não quiser que o relógio grande fique atravancando a mesa, ele pode ser simplesmente "fechado" para o tamanho original. Da mesma forma, selecionando a figura de uma calculadora, você "abrirá" uma calculadora maior, que pode ser usada para fazer cálculos aritméticos. Se você não estiver satisfeito com a disposição das figuras na mesa, pode mudá-las de lugar movimentando o mouse com o botão SELECT pressionado. Um dos efeitos mais divertidos, e que ilustra a que ponto o Lisa é modelado de acordo com o nosso modo de trabalhar, é a figura do cesto de lixo. Quando você não precisa mais de alguma parte do trabalho, joga-a no lixo usando o mouse. Com este procedimento, torna-se muito difícil apagar alguma informação acidentalmente. Você pode até examinar o conteúdo do cesto de lixo e recuperar o que for necessário, contanto que o Lisa não tenha sido desligado nesse meio tempo.

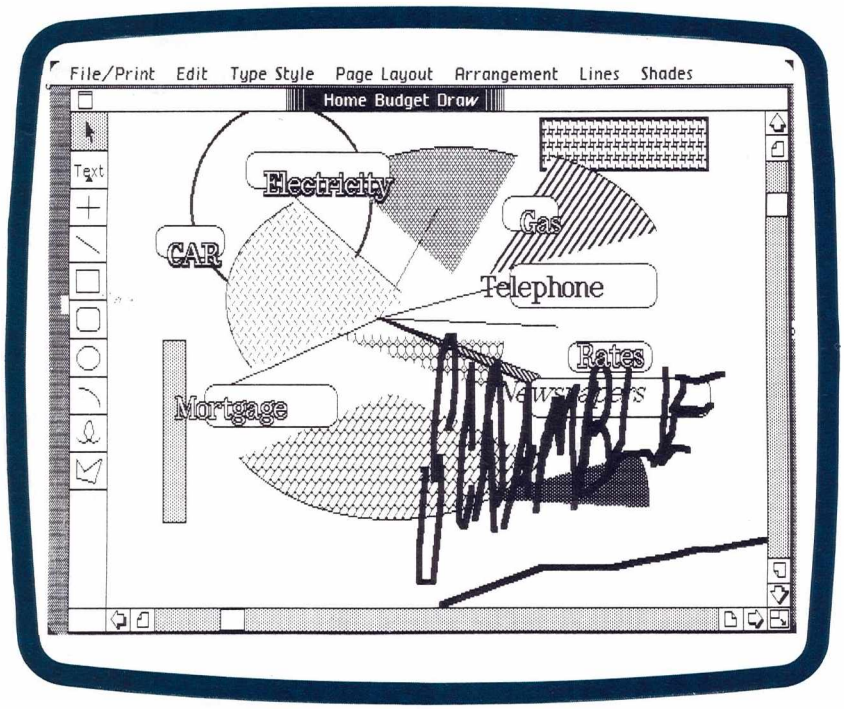
A maior parte do trabalho no Lisa é feita com a utilização de seis sistemas de aplicações: LisaWrite, um processador de palavras; LisaCalc, uma folha eletrônica; LisaGraph, um sistema para desenhar gráficos; LisaList, um gerenciador de banco de da-

**Programação rápida**
Para criar um jogo como "Defensor" em programação convencional, você desenharia numerosos modelos para o layout da tela, e então escreveria um programa a partir do zero, para controlar o jogo. Usando a programação de objeto orientado do Lisa, você pode concentrar-se em cada elemento individualmente.

Começando pela nave de ataque, você estabelece que ela andará sempre da esquerda para a direita; que quando o joystick for movimentado para a frente ou para trás, a nave andará respectivamente para cima e para baixo; quando o botão FIRE for acionado, um míssil será lançado. A seguir, você define a forma do míssil e determina que ele continue na mesma direção até entrar em contato com outro objeto, o que fará com que ele desapareça. A bateria antiaérea é definida por uma forma simples, que se transforma numa explosão, se atingida pelo míssil. Passe as três definições para linguagem adequada, ponha-as na forma de instruções, e jogue.








**Passando informação adiante**
Uma das características do Lisa é a capacidade de fazer a informação passar de uma aplicação para outra, usando a função COPY. Ela armazena, temporariamente, informação pronta para ser afixada em outra "janela". Por exemplo, comecemos por analisar alguns dados usando LisaCalc (a folha eletrônica).
Os números resultantes são copiados no LisaGraph, que com eles produz automaticamente um gráfico em forma de barras ou círculo. Finalmente, a imagem é transferida para o LisaDraw, onde é acabada com títulos, setas, diagramas etc.
O resultado final pode então ser impresso.

dos/listas; LisaProject, um auxiliar para planejamento de projetos; e LisaDraw, um instrumento sofisticado para criar qualquer tipo de imagem gráfica. As figuras ou ícones para estas aplicações são simples blocos de papel. Para fazer um cálculo na folha eletrônica, por exemplo, o cursor é posicionado sobre o bloco de papel riscado com linhas e colunas chamado LisaCalc. Apertando o botão SELECT, uma folha deste bloco é "arrancada" e colocada em outro lugar da mesa, podendo receber um título como "planejamento de vendas", por exemplo.

No caso de o usuário precisar ter várias aplicações de folhas eletrônicas na mesa ao mesmo tempo, ele deverá resselecionar a mesma figura. Num sistema de computador normal, você teria de passar pelo processo de carregar o programa de folha eletrônica e então especificar o arquivo de dados com o qual quer trabalhar. No Lisa, entretanto, programa e dados são inseparáveis. Este é um outro exemplo de programação de objeto orientado, que vimos no artigo que trata do Programa de Criação de Fliperama (Pinball Construction Set) (ver p. 241).

Outra importante característica do "ambiente operacional" (como é chamado) do Lisa é sua capacidade de apresentar "janelas". Quando uma aplicação é selecionada, ela aparece como uma grande folha de papel sobre a mesa. O tamanho dessa folha pode ser especificado com a utilização do mouse. Caso haja mais de uma aplicação "aberta" ao mesmo tempo, as folhas ficam sobrepostas, e a que estiver sendo usada no momento ficará no alto da pilha, exatamente como se elas estivessem sobre a mesa. Pode acontecer que a aplicação na qual você esteja trabalhando, o processador de palavras, por exemplo, necessite de maior espaço do que o da folha que você especificou. Neste caso, a folha passa a atuar como uma janela sobre a aplicação e pode ser movimentada de um lado para outro, mostrando qualquer parte do documento completo. O princípio das janelas foi amplamente explicado quando examinamos folhas eletrônicas (ver p. 158).

É possível passar informação de uma aplicação para outra utilizando-se os ícones ou figuras, e esta é outra importante função do Lisa. Digamos que você esteja fazendo uma análise do seu volume de vendas mensal, usando LisaCalc. Por meio da função COPY, que é selecionada de um menu de funções especiais listadas no alto da tela, você faz uma cópia temporária dos resultados da folha eletrônica. Então, escolhendo um pedaço de papel LisaGraph, esses resultados são introduzidos na seção INPUT DATA da aplicação para traçar gráficos, bastando para isso recorrer à opção PASTE do menu. Caso seja solicitado, o LisaGraph produzirá um gráfico em forma de círculo (ou de barras, ou de linhas), totalmente marcado e graduado. Agora, usando de novo as opções COPY e PASTE, que se encontram no alto da tela, pode-se copiar esta imagem num pedaço de papel LisaDraw.

Esta última aplicação lhe permite completar o gráfico com setas e diagramas, ou substituir os tipos dos títulos e legendas por uma variedade de modelos diferentes. O resultado final pode ser impresso e copiado para um retroprojetor de slides ou usado como arte final para um relatório ou artigo de revista.

Como dissemos, os princípios da programação de objetos orientados e dos ambientes operacionais no estilo do Lisa serão brevemente transferidos até para as máquinas mais baratas, principalmente se elas se tornarem mais avançadas quanto à velocidade do processador e ao tamanho da memória RAM. Procure imaginar: se houvesse janelas múltiplas na tela do seu microcomputador, você poderia escrever um programa em uma delas e observar o output em outra. Em seguida, você poderia "chamar" os auxiliares de programação, bastando apontar para uma figura em forma de caixa de ferramentas, e mudar as sub-rotinas de lugar na sua listagem simplesmente movendo o mouse sobre sua mesa de trabalho. Esperemos que tal capacidade de desempenho do software não esteja muito distante.

# Janelas para o mundo

**Viewdata é uma das poucas áreas da computação em que os padrões são internacionais. Isto permite, em princípio, o acesso de seu micro a uma rede mundial de bancos de dados.**

Sistemas tipo viewdata, como o serviço Prestel, da British Telecom ou o Videotexto da Telesp, nos possibilitam acesso a uma variedade de bancos de dados por meio de um aparelho de televisão modificado e um telefone. Estes sistemas podem ser de três tipos: público, aberto e de interesse geral, como o Prestel ou o Videotexto; aberto a todos mas específico no conteúdo, como alguns bancos de dados financeiros ou de bolsas de valores; e privado.

Os sistemas tipo viewdata, no entanto, não devem ser confundidos com sistemas de teletexto, atualmente distribuídos na Europa e nos Estados Unidos através dos canais de televisão e que usam caracteres e layout de página semelhantes. Os sistemas de teletexto, como o Ceefax da BBC inglesa, são televisionados como sinais subsidiários sobrepostos aos programas normais de televisão. Os sinais que definem uma página de teletexto são transmitidos entre os quadros de uma apresentação normal de televisão. Um dispositivo decodificador os separa e cria a imagem do teletexto. Assim, o texto e os gráficos resultantes podem substituir a imagem transmitida pela televisão, ou sobrepor-se a ela, de modo que as duas imagens aparecerão juntas, como no caso em que o teletexto é usado para colocar legendas destinadas aos deficientes auditivos.

Os sistemas de teletexto não são interativos, isto é, o espectador não dispõe de meios para mudar o conteúdo de uma página de informação ou fazer uma réplica. Todavia, o acesso a eles é livre. Tudo o que você precisa para usufruir as vantagens de um grande volume de informação é de um aparelho de televisão adaptado adequadamente.

Os sistemas tipo viewdata, por outro lado, usam o aparelho de televisão comum apenas como um monitor para transmitir informações recebidas através da rede telefônica. Normalmente, o usuário não pode alterar o banco de dados, mas pode utilizar o teclado para introduzir respostas às questões do sistema. Todos os sistemas tipo viewdata são providos de menu, isto é, a tela sempre mostra um conjunto de opções para o usuário. Ele faz a escolha digitando um número no teclado numérico. Com isso, o controle passa para o submenu selecionado e o processo é repetido através dos níveis da hierarquia até que o usuário chegue à página de informação desejada.

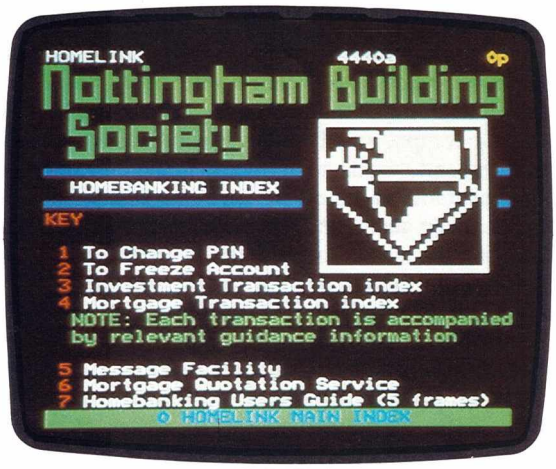
Cada página é identificada por um número único, e pode-se ir diretamente a uma determinada página digitando seu número no teclado. Este é o método mais rápido (e mais barato) para aqueles que consultam freqüentemente páginas determinadas.

Para dar um exemplo, vejamos como um usuário faz reservas para suas férias de verão pelo sistema inglês Prestel. O quadro "sign-on" (a primeira imagem que ele vê quando liga o sistema) manda pressionar a tecla com o símbolo # para chegar ao índice principal. Daí em diante, ele segue as sugestões do menu, página por página. A primeira parada é no quadro de Informações Gerais. O quarto item dessa página tem o título: Férias, Transporte, Viagens — portanto, ele aperta a tecla 4.

Como resposta, o usuário recebe várias alternativas: Transporte Ferroviário, Transporte Aéreo, Transporte Rodoviário, Outros Transportes, Férias e Turismo e Carros e Motos. Há também mais quatro "saídas", cada uma levando a um quadro de informações (uma espécie de desvio, como uma nota de rodapé num livro), de onde ele pode voltar ao corpo principal do



**Informação pública**
O serviço inglês Prestel de viewdata, da British Telecom, que compreende cerca de 250.000 páginas de informação, pode ser usado em qualquer parte do país. Para ter acesso ao banco de dados, basta uma linha telefônica e um receptor de televisão.




**Transacionando na tela**
A Nottingham Building Society, juntamente com o Bank of Scotland e a British Telecom, desenvolveram um sistema que permite aos assinantes controlar suas transações financeiras sem sair de casa. Os clientes podem examinar seus saldos bancários e de poupança, pagar contas, e até comprar uma série de bens e serviços.

texto. Após pressionar a tecla correspondente a Férias e Turismo, o veranista tem de decidir se vai reservar passagem e acomodações separadamente ou optar por um plano de excursão, e sua preferência determinará a rota de saída da página. Como um meio alternativo para selecionar as páginas, o Prestel lançou uma Lista de Opções, onde o usuário pode fazer uma escolha imediata — por exemplo, ele vai diretamente à página em que consta uma determinada empresa aérea ou rede de hotéis.

Foi tomado aqui o item "viagem" como exemplo por ser o mais usado em todo o sistema nas ilhas britânicas. As agências de viagens, em particular, são os usuários mais entusiasmados do Prestel.

Criado nos laboratórios de pesquisa da British Telecom em 1971, o primeiro sistema viewdata entrou em operação em 1976, numa experiência de caráter interno, e passou a ser distribuído para o público como Prestel no fim de 1979. A British Telecom chamou seu primeiro sistema de Viewdata, mas foi obrigada a mudar para Prestel, quando ficou estabelecido que viewdata era um termo genérico.

O viewdata talvez seja um sistema sem igual, no sentido de que se tornou um padrão aceito tanto na Inglaterra quanto internacionalmente desde o princípio. Numerosos fabricantes de computadores e companhias de telecomunicações começaram a produzir sistemas próprios que se utilizam dos protocolos e estruturas de dados do Prestel. O resultado foi a criação de uma rede de bancos de dados de âmbito mundial, com acesso livre para qualquer assinante.

O sistema tipo viewdata necessita de uma linha telefônica aberta durante todo o tempo em que estiver sendo usado e o usuário incorre nesse encargo, além da despesa de conexão com o serviço. A par disso, existe a possibilidade de outra despesa, referente a algum quadro que tenha sido retido, mas isto fica a critério de quem transmite a informação. Há também o pagamento de uma pequena assinatura anual. Na tentativa de diminuir as contas telefônicas dos usuários, o Prestel instalou diversos "concentradores locais" — linhas de troncos telefônicos exclusivos do sistema —, que fazem a ligação de uma chamada de Glasgow, por exemplo, com um número em Londres, com uma taxa de chamada local.

O hardware necessário para os serviços de viewdata divide-se em três tipos principais. O mais sofisticado e também mais caro é o terminal especial para viewdata, preferido pela maioria dos usuários comerciais. A segunda alternativa é instalar um adaptador para aparelho comum de televisão — há grande variedade desses adaptadores, que vão desde um simples teclado numérico com discagem telefônica manual até um teclado tipo máquina de escrever, com discagem totalmente automática. Porém, no núcleo de todos estes dispositivos está um microcomputador que decodifica os sinais que entram e saem. O terceiro método, conhecido dos usuários de microcomputadores, consiste em comprar software de viewdata para um micro standard. Este método tornou-se mais atraente desde a criação de um serviço chamado Micronet 800, que proporciona aos assinantes a facilidade de carregar os programas no computador diretamente pelo telefone. Além do mais, esses adaptadores permitem que uma página do Prestel seja guardada em disco, eliminando desta maneira a necessidade de se manter uma ligação telefônica constantemente aberta.

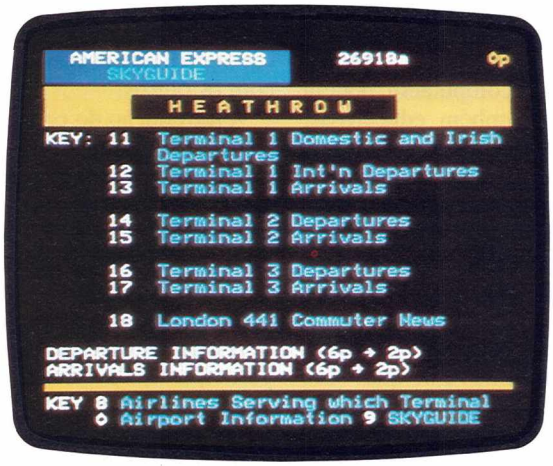
O Prestel também oferece um serviço de mensagens chamado Mailbox, pelo qual os assinantes podem deixar recados. O assinante é avisado de que há um recado para ele assim que liga seu terminal Prestel ou, se este estiver sendo usado, logo que terminar a chamada. Ele também pode procurar por recados durante uma chamada. Para mandar uma mensagem não é necessário ter acesso a um teclado alfanumérico completo, como imaginariam alguns, pois há uma variedade de "formulários de recados" padronizados que podem ser preenchidos usando-se tão-somente números.

Em meados de 1983, havia em torno de 35.000 assinantes do Prestel na Grã-Bretanha — que dispunham de mais de 250.000 páginas de informação —, e um número desconhecido de empresas e organizações utilizando as especificações do viewdata para fazer suas próprias pesquisas nos bancos de dados.

## Telesoftware

O Videotexto da Telesp, que funciona com tecnologia da empresa francesa Matra e contava inicialmente com 2.000 assinantes, começou a estender os seus serviços a partir de meados de 1984. Como o Prestel, ele passou a permitir que os usuários de micros ligassem seus equipamentos à rede. Assim, o número de assinantes do Videotexto deve chegar a cerca de 15.000 em 1985. E um serviço adicional pôde ser criado: o telesoftware, ou seja, o usuário de microcomputador tem condições agora de adquirir de alguns dos fornecedores do sistema Videotexto jogos e programas aplicativos, armazenando-os em fita cassete ou disco.






**Turismo eletrônico**
Uma das aplicações comerciais mais populares do Prestel é a das agências de viagens. As empresas aéreas, em particular, mantêm sistemas altamente avançados para reservas e emissões de passagens, embora todas elas se utilizem de computadores separados. O Prestel permite que as agências tenham acesso à maioria desses sistemas, podendo reservar e vender passagens diretamente. Os usuários também são avisados sobre as mudanças de horário dos vôos.

# Jogando pelo correio

**Nem todos os jogos exigem reações instantâneas — nos jogos postais, um lance pode levar mais de um mês e dezenas de jogadores participam da mesma partida.**

Embora os novos compradores de micros declarem que pretendem aprender programação, não há dúvida de que a aplicação mais popular para estas máquinas são os jogos. Como já tivemos oportunidade de demonstrar neste MICROCOMPUTADOR — CURSO BÁSICO, os jogos baseados em computador podem ser não só divertidos como também educativos. Você não precisa ficar restrito a *games* de fliperama do tipo Space Invaders. O computador criou novos conceitos de jogo, tais como simulações educativas e de aventuras (ver pp. 81 e 161), sem mencionar as versões computadorizadas de jogos populares que oferecem uma variedade de modos diferentes de jogar.

Todavia, existe um tipo de jogo baseado em computador que ainda não mencionamos e do qual você talvez ainda não tenha ouvido falar. Em extraordinário contraste com os jogos de fliperama, que exigem decisões e reflexos quase instantâneos, este jogo é realizado lentamente — cada lance leva semanas para ser jogado! Ao contrário dos outros jogos de computador, que na maioria são feitos para atuação individual, estes *games* podem incluir várias dezenas de participantes ao mesmo tempo.

Referimo-nos aos jogos postais, comuns na Inglaterra, cujos participantes estão espalhados por todo o país (e, no caso de jogos internacionais, por todo o mundo). Cada um deles especifica seus lances no papel, a intervalos predeterminados. Os lances são enviados pelo correio a um coordenador do jogo, que os introduz num microcomputador (a maioria dos jogos postais não exige que os jogadores tenham microcomputador próprio). O coordenador então envia a cada jogador o material impresso pelo computador, mostrando sua própria posição e outras informações relevantes, tais como as posições dos outros jogadores com quem ele tem contato.

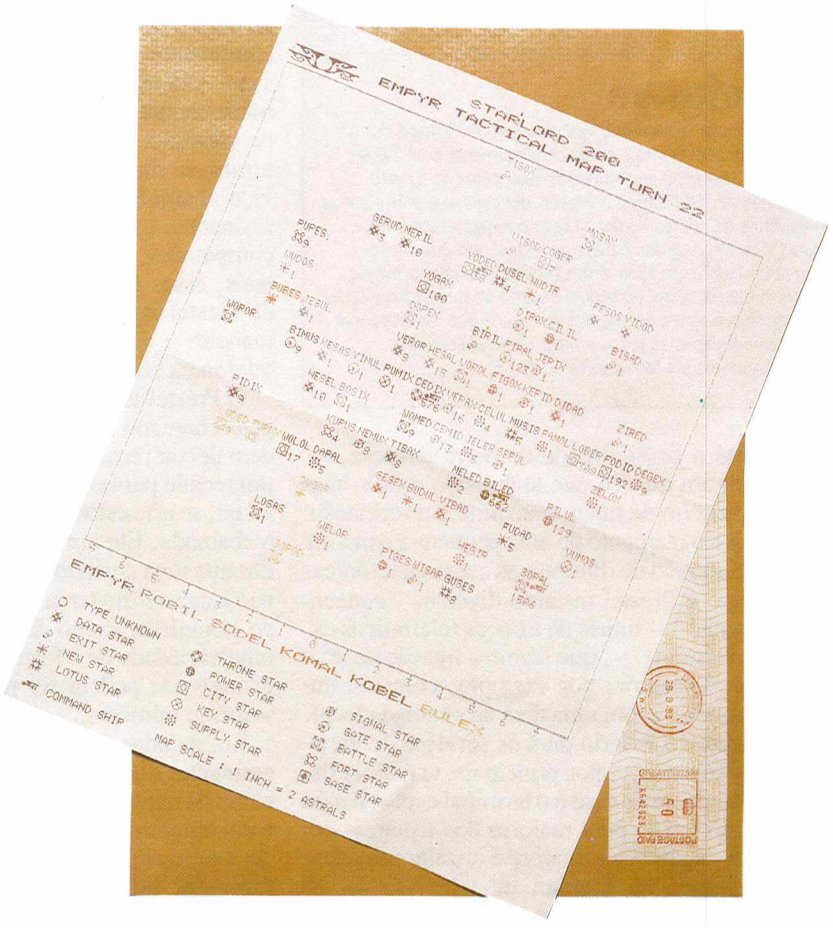
Estas partidas costumam durar meses, ou um tempo indefinido; todavia, é permitido aos jogadores entrar ou sair do jogo em qualquer etapa. É cobrada uma taxa de inscrição para cobrir as despesas iniciais com material e manual de instruções; daí em diante, uma taxa é paga a cada lance. Estes jogos não são em absoluto um passatempo barato. Com um lance por semana, a partida é considerada rapidíssima, mas jogos internacionais podem ter uma demora de seis semanas entre uma rodada e outra.

Um jogo postal típico atende a várias dezenas de jogadores. Se novos participantes quiserem inscrever-se, o coordenador começa um segundo jogo paralelamente ao primeiro, usando os mesmos programas em seu computador, mas com discos que contêm dados diferentes.

Os jogos postais já existiam bem antes do computador. Havia associações de xadrez postal e também o popular jogo inglês Diplomacia podia ser feito pelo correio — nesta competição, os representantes de sete nações européias têm por objetivo dominar o continente e para isso fazem e rompem alianças uns com os outros.

A introdução do computador, para fazer todos os cálculos e administrar o jogo, foi sinal de que o campo de ação destes jogos aumentou, assim como sua engenhosidade e refinamento. Alguns apresentam vastas galáxias, através das quais os jogadores manobram suas frotas de espaçonaves; outros envolvem terras fabulosas e reinos beligerantes; e existem também versões computadorizadas do Diplomacia.

A singularidade a respeito destes jogos é que os participantes se influenciam reciprocamente — eles não ficam só fazendo explorações, como nos jogos de aventuras. As alianças entre dez ou mais jogadores não são incomuns. Uma medida da qualidade destes jogos é que a faixa etária dos assinantes é bem mais ampla do que a dos outros tipos de jogos de computador.




**Guerra nas galáxias**
"Starlord" foi o primeiro jogo postal a ser administrado por computador na Grã-Bretanha. O objetivo dos jogadores é encontrar a Throne Star (Estrela Trono) e tornar-se imperador. A cada lance eles recebem um mapa mostrando a área mais próxima em torno de suas forças e uma lista de quem controla as estrelas vizinhas. O computador que controla a partida trabalha com um disco de 7,5 megabytes devido ao amplo número de programas que dirigem o jogo e à grande quantidade de informação que deve ser mantida para atender a mais de 700 jogadores.

# Comportamento simulado

**A simulação é uma técnica em computador que permite experimentar situações que em outras condições seriam muito perigosas ou dispendiosas. Simulações em microcomputadores podem ser muito instrutivas.**

Uma das mais importantes utilizações do computador situa-se na área da simulação. Trata-se de um método de planejamento em que se examina um modelo da situação a ser analisada, simulando-a no computador. O modelo fornece uma visão simplificada da situação específica, retendo os aspectos importantes do problema e descartando os detalhes de menor influência nos resultados finais.

Para analisar uma situação, tomemos um exemplo de dois copos, um com vinho branco e o outro com vinho tinto. Coloca-se uma colher de vinho tinto no vinho branco e, depois de misturados, uma colher desse líquido é posta no copo de vinho tinto. Qual o copo que fica com maior "impureza"?

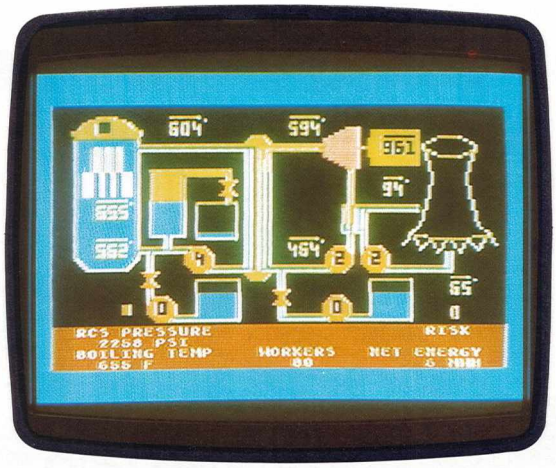
Há várias maneiras de resolver esse problema, mas uma das mais simples é utilizar um modelo. Por exemplo, podemos estabelecer um modelo em que o volume de vinho em cada copo seja exatamente de uma colher. Será fácil perceber que ambos os copos ficarão com o mesmo grau de impureza (uma mistura igual de tinto e branco). Se ampliarmos esse modelo para maior quantidade de vinho, chegaremos à mesma conclusão.

Há três formas principais de modelos. Um modelo que podemos chamar de pictórico ou bem expressivo — por exemplo, uma fotografia ou até mesmo um mapa — mostra os arranjos e relações espaciais entre os elementos da figura. Outra forma de modelo é aquele cujos componentes se comportam, um em relação ao outro, de modo semelhante aos elementos reais que ele representa no problema — por exemplo, os problemas solucionados por computadores analógicos (ver p. 238). O terceiro tipo é o modelo que usa símbolos abstratos e relações matemáticas para representar uma situação. Os computadores digitais usam este último tipo na simulação.

Destacam-se quatro situações de problemas que podem ser solucionadas por simulação no computador. A primeira é a que apresenta alto risco e impede experiências reais; por exemplo, a determinação do nível seguro de radiatividade numa usina nuclear.

Na segunda situação, da qual um bom exemplo é a economia de um país, seria quase impossível encontrar solução matemática simples para o conjunto de equações que compõem o problema. Será melhor dispor as equações na forma de um modelo no computador e observar os efeitos de diferentes ações e eventos sobre ele.

O terceiro caso se dá quando o problema a ser analisado envolve tanta despesa que alguns ajustes devem ser feitos no modelo, para evitar um erro na versão final. No projeto de um novo aeroporto em Londres, por exemplo, um longo trabalho de simulação foi realizado para solucionar questões de planejamento e engenharia. Essa simulação fez pesquisas sobre o barulho e outras condições relacionadas ao meio ambiente, assim como o fluxo de pessoas e o tráfego em torno da área escolhida.



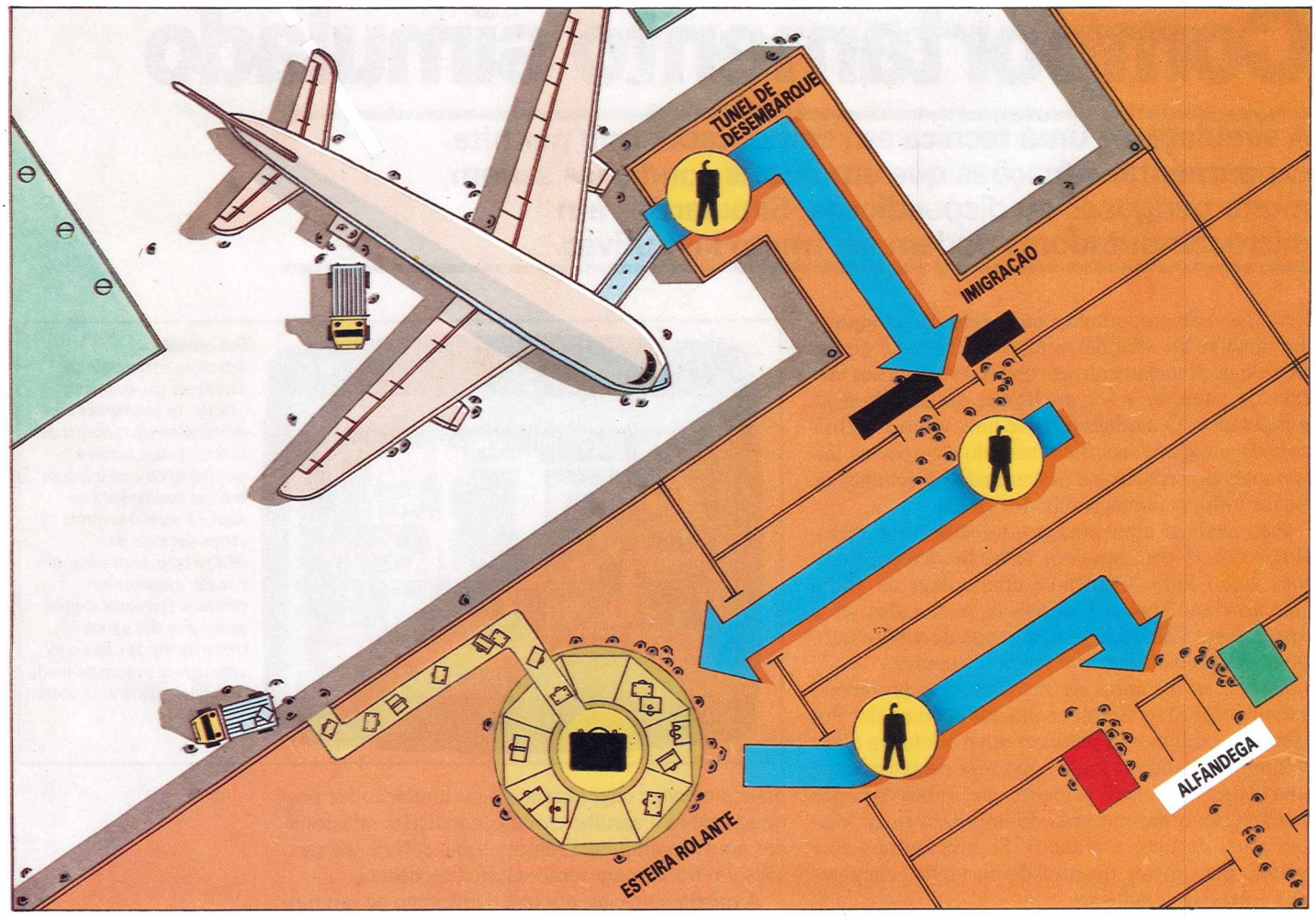

**Sob controle**
Aplicação importante da simulação dá-se na área escolar: os estudantes são exercitados em modelos de sistemas reais, como a simulação de uma usina de energia nuclear feita no Atari. O usuário controla os vários sistemas de refrigeração, para evitar que o reator esquente em demasia. O manual explica as funções dos vários mecanismos da usina e há uma demonstração do modo pelo qual o programa opera.

A quarta situação, em que o emprego de um modelo assume grande importância, ocorre quando um problema se restringe ao campo teórico e é impossível a experiência *in loco*. Os astrofísicos, por exemplo, especulam sobre a formação das estrelas utilizando modelos para avaliar uma teoria cosmológica — digamos, a teoria do "Big Bang" — em relação a uma outra hipótese.

Em toda simulação, a primeira coisa a fazer é organizar um modelo e decidir quais são os elementos importantes e como estão interligados.

Os sistemas e seus modelos dividem-se em dois grupos: sistemas fechados ou determinísticos, e sistemas estocásticos ou de conclusões abertas. Em um orçamento doméstico, as despesas podem ser distribuídas de várias formas, mas o livro de contas deve fazer o balanço no final — trata-se, neste caso, de um sistema determinístico. Entretanto, se a família gastar dinheiro de forma aleatória, sem considerar a quantia disponível em seu balanço, o sistema é chamado estocástico.

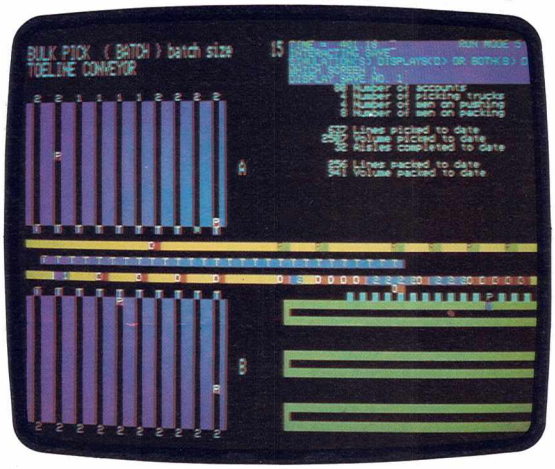
Por meio de simulações teóricas não se pode ter certeza absoluta de que o modelo escolhido é o mais correto. Imaginava-se que a Terra era o centro do universo, até Copérnico apresentar um modelo matemático bem mais simples, tendo o Sol no centro. Atualmente, os astrônomos, ao observarem as galáxias, constatam que todas estão se afastando de nós a velocidades crescentes, o que sugere, mais uma vez, que estamos no centro do universo. Mas, se adotarmos um modelo cósmico com a Terra no centro, cometeremos um equívoco, como mostra o seguinte




**Um problema Jumbo**
A simulação em computador foi usada no projeto de um novo terminal do aeroporto de Heathrow, em Londres. O programa simulava, primeiro, o tráfego de aviões, com o número dos passageiros e bagagens. Depois simulava os "processos" a que os passageiros seriam submetidos. O modelo permitiu, assim, adequar as dimensões do terminal projetado.

modelo alternativo. Se você respingar uma bexiga de plástico com pontos de tinta para representar as estrelas e, a seguir, inflar a bexiga, vai notar, evidentemente, que os pontos de tinta se afastam uns dos outros, e nenhum deles está no centro do balãozinho.

Entretanto, há muitas vantagens no uso de modelos. Eles ajudam a formular melhores teorias, abreviam o tempo das análises, permitem novas tentativas e modificações e, o mais importante, são bem mais baratos que uma experiência real. O uso de modelos também permite chegar a novas descobertas. O laser, por exemplo, foi inventado quando se pesquisava um aspecto de um modelo matemático para o qual não se havia dado anteriormente atenção.




**Veja por quê**
A BL Systems, uma divisão da Leyland inglesa, desenvolveu um pacote de modelos para microcomputador, com projetos de novas linhas de produção e instalações. Esse pacote é chamado "Veja por quê". Ele se caracteriza pela saída gráfica na forma de diagramas esquemáticos. Os números ao lado de cada etapa do processo indicam a progressão do trabalho e salientam qualquer problema.

A BL Systems, subsidiária da Leyland inglesa, pôs à venda um sistema de simulação de múltipla finalidade, que foi originalmente desenvolvido para projeto de linhas de produção e depósitos automatizados, em duas cidades. O sistema é chamado "Veja por quê", e utiliza saídas gráficas para mostrar resultados em vez de tabelas comuns de estatística. O sistema tem tido boa aceitação e foi usado pela Administração de Aeroportos Ingleses para modelar um novo terminal em Heathrow.

Um problema típico que pode ser examinado por meio da simulação é a ordem de chegada de aviões. Num aeroporto, se o vento mudar de repente, e apenas uma das pistas de pouso estiver disponível, os aviões terão inevitavelmente de formar fila. Eles dispõem de reserva limitada de combustível, e leva algum tempo até cada um poder aterrissar. Assim, as regras de enfileiramento serão programadas no sistema. Nessas simulações, geradores de números randômicos (ver p. 209) são usados para criar eventos inesperados, como pousos casuais.

A simulação é uma área importante de aplicação para os computadores digitais e resultou em novas linguagens, especialmente escritas para a simulação de projetos (por exemplo, GASP, SIMSCRIPS e GPSS). À medida que as atividades humanas se tornam mais complexas, o uso de modelos para simular problemas tem-se revelado cada vez mais importante.

# Apple IIe

**Muitos se admiram de sua longevidade, mas o Apple II ainda está entre os microcomputadores mais versáteis, e a quantidade de software disponível para ele é maior do que para qualquer outro.**

Sob muitos aspectos, o Apple II é o equipamento com que tudo começou, pois, apesar de não ter sido o primeiro microcomputador disponível, foi com o resultado de seu lançamento que se tornaram acessíveis ao usuário de microcomputadores recursos tais como cor, gráficos de alta resolução e incorporação de som. Embora estas características sejam importantes, as que de fato são significativas no Apple II, tornando-o tão popular, devem ser ressaltadas.

A característica mais notável é o padrão da documentação, deliberadamente estabelecido para que todos os aspectos do equipamento sejam tão claros quanto possível para seus usuários. Esta atitude foi oposta à das poucas empresas do ramo nessa época, que mantinham (e em alguns casos ainda mantêm) sigilo absoluto em relação ao interior de seus produtos. Como resultado direto dessa informação livremente acessível, outra vantagem importante do Apple II tornou-se bem conhecida.

Esta segunda vantagem é a fileira de conectores de expansão (slots) que ocupa a parte de trás da placa principal do equipamento. Foi a flexibilidade de organização destas fendas que possibilitou a grande variedade de produtos adicionais disponíveis para a máquina. Tal diversidade, por sua vez, levou ao uso do Apple II sob as mais diferentes formas.

Muitos computadores têm conector de expansão, em geral um ou dois, que se apresentam como uma área relativamente pequena da memória. O Apple possui sete conectores na versão mais recente (o IIe) e oito nos modelos mais antigos (II e II Plus), e cada um deles representa para a CPU duas unidades de memória (uma bem pequena e outra muito útil, de 2 Kbytes).

Em conseqüência, um cartão periférico que seja inserido no Apple pode dispor de 2.048 bytes para o programa de controle na placa. Isto simplifica bastante o uso desse cartão, já que não há necessidade alguma de conectar programas de controle especiais ou de reescrever o sistema operacional.

Existe hoje uma ampla seleção de cartões para o Apple II, que vão desde as relativamente simples interfaces de entrada/saída até cartões bastante elaborados, tais como canetas ópticas e cartões de grande capacidade de RAM, que podem expandir a memória para 1 megabyte ou mais.

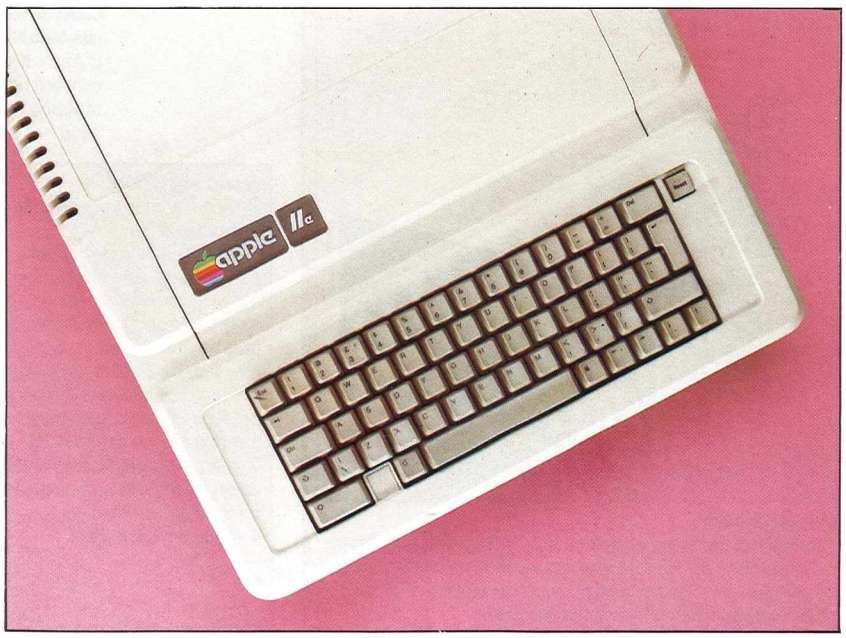
Entretanto, como o microprocessador 6502 só pode endereçar 64 Kbytes, a memória está disposta em "bancos" de 64 Kbytes, que são selecionados um de cada vez. Computadores completos, com elaborados chips microprocessadores de 16 ou 32 bits, podem até ser ligados para funcionar em paralelo com o 6502 da máquina, e criam um sistema que possui tanta (ou mais) capacidade de computação quanto um minicomputador.

Naturalmente, uma ampla variedade de software tem sido desenvolvida para utilização desta riqueza de hardware, e hoje a biblioteca de programas Apple é sem dúvida a maior do mundo — maior até do que as listas de programas que usam o sistema operacional padrão CP/M.

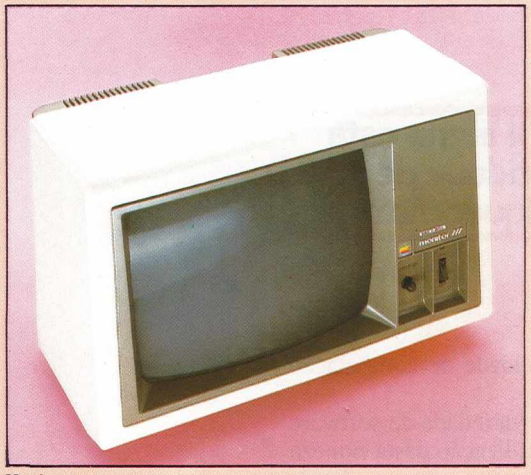
Na verdade, a biblioteca para CP/M pode ser incluída na lista do Apple, porque, embora o Apple possua um processador 6502, ele pode rodar programas CP/M com o auxílio de um dos acessórios mais populares — um cartão Z80 (um CP/M só pode ser rodado com um microprocessador Z80). Assim que um destes é inserido em um conector do Apple, a máquina funciona como um computador CP/M perfeitamente normal.

Apesar de ser bem pequeno e de aparência despretensiosa, o Apple II é um equipamento com tendência a ser objeto de superlativos. É o computador pessoal com mais sistemas operacionais (onze), diferentes linguagens (pelo menos 27), e editores de textos (doze ou mais) do que qualquer outra máquina.

A história do Apple II está longe de ser concluída; o lançamento de um sistema operacional inteiramente novo para a máquina, chamado ProDOS, tem muitas características que farão o desempenho do Apple II ultrapassar o de alguns dos mais caros sistemas.




**Teclado do Apple**
O teclado foi projetado nos mais altos padrões, tendo em conta o processamento de palavras. As teclas são adequadamente moldadas e de perfil meio côncavo, para comodidade de uso. As duas teclas marcadas com o logotipo Apple nos dois lados da barra de espaço são de controle, utilizadas em programas aplicativos. À primeira vista, a tecla Reset (zerar) pode parecer próxima demais da tecla Delete (suprimir). Entretanto, para zerar o computador, é necessário manter pressionada a tecla CTRL, e então pressionar Reset.



**Monitor**
O Apple IIe trabalha com qualquer monitor adequado, mas, é claro, combina melhor com seu design a própria unidade da Apple. Apresenta 80 colunas de texto e é equipado com filtro antibrilho.

**CPU 6502**

**Conector para alto-falante**

**Conector auxiliar**
No Apple IIe, o slot 3 é duplicado por esta fenda ligeiramente maior, que tem todos os sinais normais de bus (barramento) e também alguns adicionais para trabalhar com o segundo banco de 64 Kbytes que pode ser acrescentado.

**RAM**
O Apple IIe possui exatamente 64 Kbytes de RAM, mantidos em oito chips 4164, mas o mapeamento efetivo é mais complexo, já que os endereços dos chips periféricos são colocados no meio da RAM.

**ROM de vídeo**
Os caracteres impressos na tela são gerados por este ROM, disponível em vários conjuntos de linguagens.

**ROM do teclado**
O mapeamento do teclado, controlado por este ROM, é modificado de acordo com as necessidades de cada país. No Brasil, o Unitron e o Micro Engenho têm teclado adaptado.

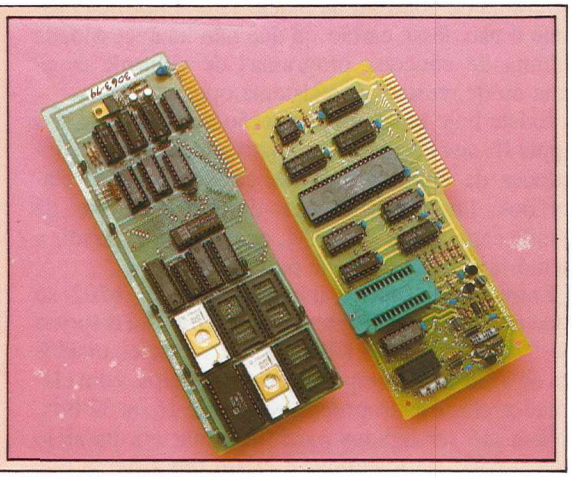
**Conector para teclado numérico reduzido**
O Apple II pode ser equipado com um teclado numérico reduzido, que é conectado aqui.

**Conector do teclado**



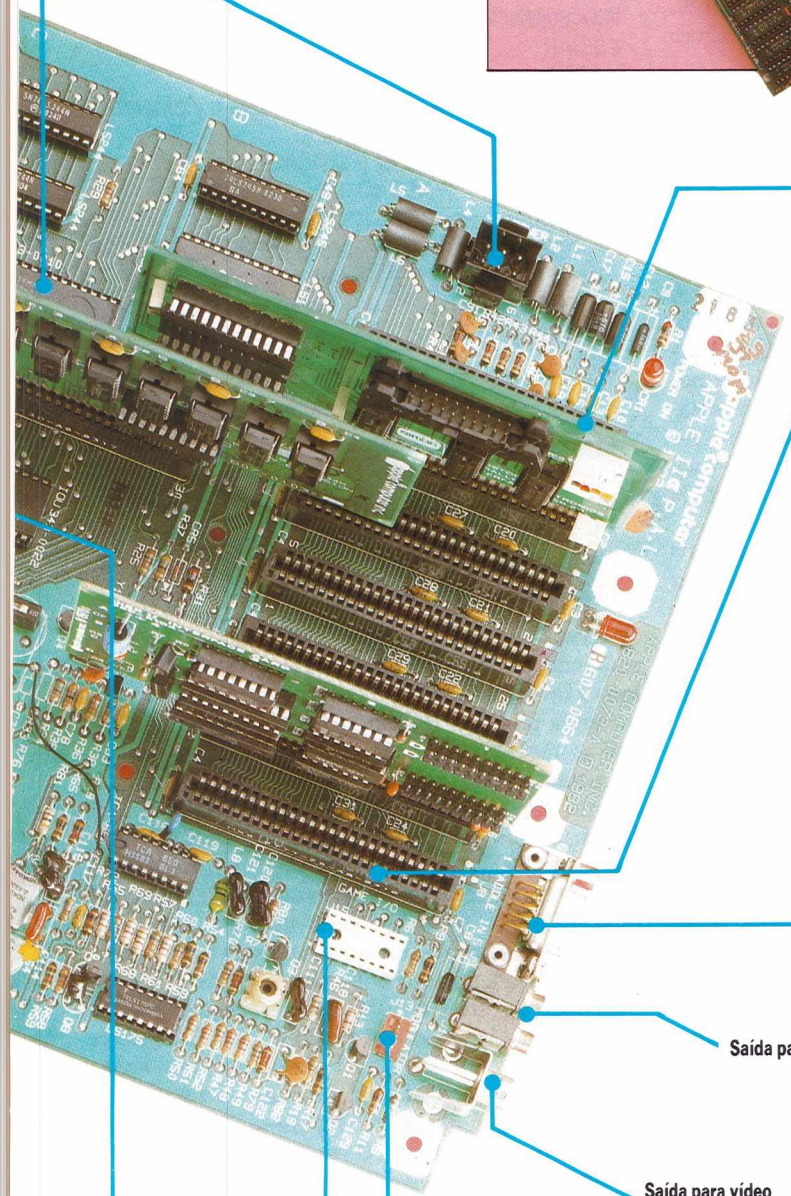
**Unidade de discos**
As unidades de discos Apple não são "inteligentes", de modo que um cartão de controle deve ser inserido na placa principal do computador. Nos padrões atuais, a capacidade de 143 Kbytes é baixa, mas ainda adequada.

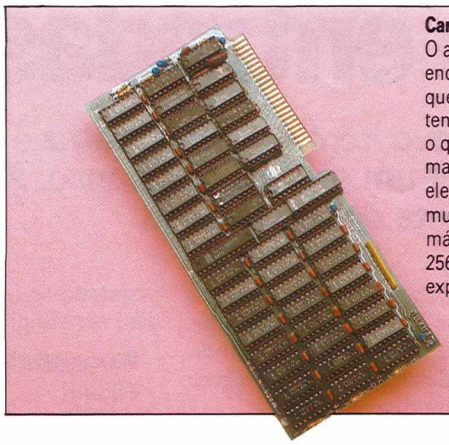
**Gravador de EPROM**
Devido à grande variedade de instrumentos de desenvolvimento que existe para o Apple, este dispositivo é ideal para uso com um sistema de desenvolvimento para novos programas. O gravador de EPROM pode ser utilizado por qualquer um para produzir EPROMs que então se inserem em um cartão ROM. Quando o programa tiver sido aperfeiçoado, ele será posto na forma ROM, no caso de ser vendido em quantidade.



**Unidade de controle de memória**
Uma das duas ULAs, que constituem a diferença principal entre o Apple II e o Apple IIe. Esta unidade controla a tela de 80 colunas, bem como o segundo banco de RAM de 64 Kbytes.

**Conector de fonte de alimentação**





**Cartão RAM de 256 Kbytes**
O avançado mapeamento de endereços do Apple permite que múltiplos de até 2 Kbytes tenham seu banco comutado, o que resulta em um acesso mais complicado. Entretanto, ele permite o uso de blocos muito grandes de RAM na máquina. Este cartão comporta 256 Kbytes, mas pode ser expandido para 1 Mbyte!

**Slot 1**
Este é normalmente ocupado por uma interface de impressora paralela.

**Slot 7**
Sinais especiais de vídeo são disponíveis apenas neste slot, de modo que cartões como os de canetas ópticas e moduladores de cor são ligados aqui.

**Tomada DIL de entrada para jogos**
Uma das características mais inovadoras do Apple é a entrada para jogos, que proporciona uma mínima, mas útil, forma de entrada analógica.

**Saída para cassete**

**Saída para vídeo composto**

**Conector auxiliar de vídeo**

**Conector-D para jogos**
A tomada de 16 pinos da porta de jogos é frágil para uso diário, e por isso o Apple II tem uma pequena tomada-D em paralelo.

**Unidade de entrada/saída**
A ULA, que manipula o endereçamento do conector auxiliar.

## APPLE IIe

**DIMENSÕES**
460 × 385 × 115 mm

**CLOCK**
1 MHz

**MEMÓRIA**
16 Kbytes de ROM e 64 Kbytes de RAM. Possibilidade de expansão para 128 Kbytes, ou mais, com comutação de "banco".

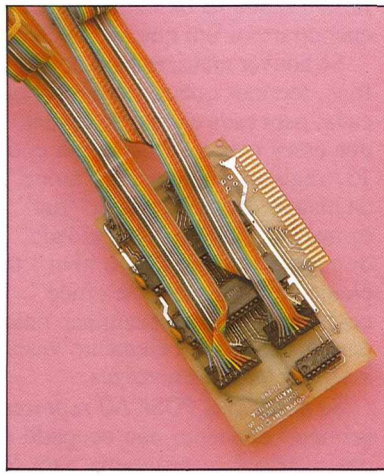
**VÍDEO**
24 linhas de 40 caracteres, apenas monocromático. Gráficos de baixa resolução de 48 × 40 em 16 cores. Gráficos de alta resolução de 192 × 280 em 6 cores.

**TECLADO**
62 teclas de alta qualidade

**LINGUAGENS**
BASIC Applesoft
A maioria das linguagens opcionais comuns e algumas menos comuns são disponíveis.

**PERIFÉRICOS**
Cassete, vídeo composto, 7 slots para expansão, entrada para jogos.

**DOCUMENTAÇÃO**
A documentação que acompanha o equipamento é de alto padrão, embora o avançado material, necessário para uma boa compreensão da máquina, tenha de ser adquirido em separado, e é razoavelmente caro. Há grande variedade de livros para todos os níveis de interesse, e a máquina, provavelmente, é a que recebe maior cobertura sob este aspecto.



**Cartão I/O para uso geral**
Às vezes um cartão tem tantas funções possíveis que se torna obrigatório uma ROM de controle para restringir seu uso. Este cartão, com dois versáteis adaptadores para interface 6522, é um exemplo disso. Possui 40 linhas I/O controláveis separadamente, dois registros de comutação, para converter dados da forma paralela em serial, e quatro relógios de 16 bits.

# Novas entradas

**Para inserir um novo item em uma matriz é necessário primeiro encontrar um espaço em branco. A busca binária é um modo eficiente de consegui-lo.**

Vimos na parte anterior como um arquivo de dados é constituído de registros, que são divididos em campos, podendo cada um deles ter acesso a outros, através de um campo de indexação. Vejamos agora algumas técnicas de pesquisa disponíveis.

Não é difícil a criação de registros para nossa agenda de endereços. Suponhamos que exista uma variável alfanumérica para cada um dos campos no registro. Estes podem ser denominados NOMECOMP$ (para referência a ''nome completo''), RUA$, CIDADE$ e TEL$ (posteriormente explicaremos nossa utilização aqui de variáveis alfanuméricas, em vez de variáveis numéricas para o campo do número telefônico). Na lista das oito funções necessárias ao programa da agenda de endereços, a de número seis possibilitava a inclusão de novas entradas. Se essas oito opções forem apresentadas na tela, no início do processamento do programa, a escolha da função de número seis levará você a uma rotina de entrada semelhante à apresentada como exercício.

Admitamos que haja uma série de entradas na agenda de endereços, mas você não pode se lembrar de quantas. É fundamental que as novas entradas não se sobreponham às já existentes; desse modo, uma das tarefas do programa poderá ser a de buscar os elementos em uma das matrizes, a fim de encontrar o primeiro deles que não contenha dados.

A busca em uma matriz para verificar se um elemento está ''ocupado'' não é difícil. Como as variáveis numéricas, também as variáveis alfanuméricas podem ser comparadas, na linguagem BASIC. A instrução IF A$ = "RESIDÊNCIA" THEN... é tão válida como IF A = 61 THEN..., pelo menos na maioria das versões do BASIC. Se qualquer das matrizes em nossa agenda de endereços já tiver uma entrada, esta consistirá em pelo menos um caractere alfanumérico. Os elementos ''vazios'' não conterão caracteres alfanuméricos; desse modo, tudo que necessitamos fazer é examinar os elementos, desde o primeiro, até encontrarmos um que não contenha caracteres.

Se houver matrizes para nome, rua, cidade e telefone, teremos quatro tabelas, com um elemento em cada, para cada campo do registro. Uma vez que todos esses campos ''estão em correspondência'', o 15.º registro terá seu item ''nome'' no 15.º elemento da matriz ''nome''; o item ''rua'', no 15.º elemento da matriz ''rua''; o item ''cidade'', no 15.º elemento da matriz ''cidade''; e o item ''telefone'', no 15.º elemento da matriz ''telefone''. Desse modo, necessitamos apenas examinar uma dessas matrizes para encontrar um elemento vazio; não precisamos verificar todas as matrizes.

Se a variável POSIÇÃO representar o número do primeiro elemento não ocupado em qualquer uma das matrizes, o programa para localizar POSIÇÃO (supondo-se que ela não seja ainda conhecida) poderá ser simplesmente o seguinte:

```
PROCEDIMENTO (encontrar o elemento não ocupado)
INICIAR
  LOOP
    REPEAT UNTIL (REPETIR ATÉ) que o elemento
    não ocupado seja localizado
    READ (LER) a matriz (POSIÇÃO)
    POSIÇÃO = POSIÇÃO + 1
    IF (SE) a matriz (POSIÇÃO) = " "
      THEN (ENTÃO) registrar a POSIÇÃO
      ELSE (CASO CONTRÁRIO) nada executar
    ENDIF
  ENDLOOP
END
```

Em BASIC, isto seria simplesmente o seguinte:

```
1000 FOR L = 0 TO 1 STEP 0
1010 LET POSICAO = POSICAO + 1
1020 IF NOMECOMP$ (POSICAO) = " " THEN LET
  L = X
1030 NEXT L
1040 REM resto do programa
```

Observe que o valor da variável X na linha 1020 corresponde ao valor exigido para o encerramento do loop FOR-NEXT e este valor varia de acordo com a máquina utilizada (os compatíveis Sinclair exigem que as variáveis sejam predefinidas). É igualmente importante notar que este é um fragmento de programa que supõe que a variável 1 NOMECOMP$( ) está DIMensionada e que foi dado um valor inicial à variável POSIÇÃO. Para processar este fragmento independente, você deve DIMensionar a variável NOMECOMP$( ) e dar um valor inicial às variáveis POSIÇÃO e X, em algum ponto, antes da linha 1000.

Embora já tenhamos utilizado a técnica FOR X = 0 TO 1 STEP 0, esta é uma boa ocasião para examinar mais detalhadamente seu modo de operar. Geralmente, um loop FOR-NEXT na linguagem BASIC ''sabe'' com antecedência quantas vezes o fragmento do programa deverá repetir-se. Se você quiser repetir alguma coisa 30 vezes, a forma FOR X = 1 TO 30 será muito conveniente. Todavia, desta vez, estamos simulando um loop REPEAT...UNTIL. Apesar de as versões comuns do BASIC não terem procedimento REPEAT...UNTIL, é bastante fácil simulá-lo usando a seqüência FOR-NEXT. Enquanto o teste da linha 1020 for negativo, a variável L (o contador do loop FOR-NEXT) permanecerá com o valor 0, sendo-lhe acrescentado o valor 0 a cada interação (repetição do loop), enquanto a linha 1010 fará com que a variável POSIÇÃO seja aumentada em uma unidade a cada interação. Quando o teste da linha 1020 apresentar o resultado ''verdadeiro'' (isto é, quando for encontrado um elemento vazio da variável

NOMECOMP$( ), a variável L assumirá o valor X e o loop FOR-NEXT será encerrado na linha 1030. Isso faz com que a variável POSIÇÃO indique o primeiro elemento livre na variável NOMECOMP$( ).

POSIÇÃO é uma variável cujo valor provavelmente precisará ser determinado no início, toda vez que o programa da agenda de endereços for utilizado, e que também necessitará ser atualizada diversas vezes durante a utilização do programa. Assim, ela será uma das nossas variáveis "globais" e o estabelecimento de seu valor deverá ser parte de uma rotina de "inicialização". Isto pode ser feito toda vez que o programa for processado, ou um "flag" pode ser criado para indicar se o valor da variável POSIÇÃO foi alterado ou não desde o último processamento do programa. Esta última abordagem não é difícil; porém, neste estágio, cria uma complexidade desnecessária. Para deixar simples o procedimento, nossa primeira tarefa será encontrar o valor da variável POSIÇÃO, sempre que o programa for processado.

Vamos revisar os procedimentos que desejamos que a agenda de endereços computadorizada execute, examinando também se é possível chegar a uma estratégia global de programa. Desta vez, seremos um pouco rigorosos, supondo que cada uma das atividades será tratada como uma sub-rotina separada (cujo nome virá indicado entre asteriscos).

| | |
|---|---|
| 1. Encontrar um registro (pelo nome) | *ENCREG* |
| 2. Encontrar nomes (a partir de nomes incompletos) | *ENCNOMES* |
| 3. Encontrar registro (a partir da cidade) | *ENCNOMCID* |
| 4. Encontrar registros (a partir de iniciais) | *ENCINICIAL* |
| 5. Listar registros (todos) | *LISTREGS* |
| 6. Acrescentar registro | *ACRESCREG* |
| 7. Modificar registro | *MODREG* |
| 8. Eliminar registro | *ELIMREG* |
| 9. Sair do programa (gravar) | *SAIRPROGR* |

Sabemos agora, em termos gerais, quais são as "entradas" e "saídas" do programa que desejamos; desse modo, podemos começar a pensar em termos de programa principal. Todos os detalhes podem ser executados pelo processo de programação "top-down" e codificados em várias sub-rotinas. Também sabemos que deveremos dar um valor inicial a vários elementos, inclusive o valor da variável POSIÇÃO. Sabemos igualmente que, como o programa é dirigido por menu, receberemos um conjunto de escolhas toda vez que o programa for processado. Sabemos ainda que, qualquer que seja a resposta às opções apresentadas, desejaremos que pelo menos uma delas seja executada.

Assim, o corpo do programa principal já pode assumir uma forma:

**PROGRAMA PRINCIPAL**
INICIAR
  ATRIBUIR UM VALOR INICIAL (procedimento)
  SAUDAÇÃO (procedimento)
  ESCOLHA (procedimento)
  EXECUÇÃO (procedimento)
ENCERRAR

O programa acima seria escrito em BASIC da seguinte forma (com os números de linha substituídos para os nomes das sub-rotinas):

```
10 REM PROGRAMA AGENDA DE ENDERECOS
20 GOSUB *INICIALIZAR*
30 GOSUB *SAUDACAO*
40 GOSUB *ESCOLHA*
50 GOSUB *EXECUCAO*
60 END
```

A sub-rotina ou procedimento *SAUDAÇÃO* apresenta, por alguns segundos, uma saudação na tela seguida pelo menu. A saudação provavelmente terá a seguinte forma:

*BEM-VINDO AO*
*MICROCOMPUTADOR — CURSO BASICO*
*AGENDA DE ENDERECOS COMPUTADORIZADA*
(PRESSIONAR A BARRA ESPACEJADORA, QUANDO PRONTO PARA CONTINUAR)

Em resposta à solicitação para pressionar a barra espacejadora, o programa desviará para a sub-rotina *ESCOLHA* e a tela se apresentará ao usuário da seguinte forma:

*VOCE DESEJA*
1. ENCONTRAR UM REGISTRO (pelo nome)
2. ENCONTRAR NOMES (por trecho de um nome)
3. ENCONTRAR REGISTROS (de uma cidade)
4. ENCONTRAR REGISTROS (de uma inicial)
5. LISTAR TODOS OS REGISTROS
6. ACRESCENTAR UM REGISTRO
7. MODIFICAR UM REGISTRO
8. ELIMINAR UM REGISTRO
9. SAIR E GRAVAR

*ESCOLHER DE 1 A 9*
*SEGUIDO DE RETURN*

Neste ponto, o programa desviará para a sub-rotina correspondente, de acordo com o número fornecido. A estrutura do programa começa agora a tomar forma. Todas as alternativas, exceto a de número 9 (para SAIR e GRAVAR), necessitarão encerrar-se com uma instrução para retornar à sub-rotina *ESCOLHA*. Mas há muitos detalhes de organização interna dos dados que não examinamos. Trataremos deles posteriormente.

Vamos admitir que estamos processando o programa, que ele já possui todos os registros de que precisamos e que desejamos encontrar um registro completo, dando entrada a um único nome. Isto exige a opção 1 — ENCONTRAR UM REGISTRO (*ENCREG*). Antes de tentar elaborar esta parte do programa, examinemos alguns dos problemas vinculados às rotinas de busca.

## Busca

Manuais sobre técnicas de programação tendem a tratar conjuntamente dos procedimentos de busca e ordenação (sort). Os leitores devem lembrar-se de que já falamos sobre o procedimento de ordenação em um programa elaborado para dispor nomes em ordem alfabética (ver p. 134). Tanto o procedimento de busca como o de ordenação levantam aspectos interessantes sobre como os dados são organizados — em computador ou em outro sistema de informação.

Se uma agenda de endereços comum for constituída por uma caderneta sem índice alfabético e se os itens forem sendo acrescentados à medida que se quiser incluir novos nomes e endereços, sem ordem alfabética, teremos uma estrutura de dados denominada "pilha". Pilhas são conjuntos de dados reunidos segundo a ordem de chegada. É evidente que a forma de pilhas é a menos eficiente para organizar dados. Para encontrar o endereço e o número do telefone de alguém, seria preciso procurar em toda a agenda. O mesmo, geralmente, acontece com os sistemas de computação, embora haja ocasiões em que os critérios de acesso aos dados sejam tão incomuns que o procedimento por pilhas pode mostrar-se conveniente.

Uma estrutura de dados mais organizada e de mais fácil utilização, tanto para os usuários como para o próprio computador, é obtida pela organização dos dados de acordo com um método conhecido e simples. A lista telefônica é um bom exemplo de conjunto de dados (nomes, endereços e números telefônicos) em que o campo do nome é ordenado de acordo com as regras simples da seqüência alfabética. Os números, para todos os efeitos, são ordenados aleatoriamente, mas os nomes — que são mais "significativos" —, organizados de acordo com regras fáceis de serem observadas.

Uma vez examinada a organização interna dos dados em nossa agenda de endereços computadorizada, os dados são organizados como uma pilha, armazenando um primeiro registro no elemento × da matriz nome, elemento × da tabela rua, e assim sucessivamente; e o registro seguinte, no elemento × + 1 da matriz nome, e o elemento × + 1 da matriz rua e assim por diante. A busca de um determinado item de dados — por exemplo, LUIZ SILVA — exigirá a observação do primeiro elemento da matriz nome a fim de verificar se é LUIZ SILVA; repetir esse procedimento com o segundo elemento e assim sucessivamente, até que tenhamos localizado o campo ou até a constatação de que não existe o item LUIZ SILVA.

Se os dados que desejamos encontrar já foram ordenados em uma estrutura identificável, poderemos verificar como esta ordenação simplificará a busca. Suponhamos que você tenha um banco de dados para times de futebol e que um dos campos dos registros seja a contagem de gols em uma semana determinada. Um banco de dados de alta capacidade lhe permite buscar o time ou times que marcaram 11 gols nessa semana. Eis uma tabela que apresenta as contagens de gols para cada time na semana em questão:

1,6,2,2,1,9,0,0,2,1,4,11,4,2,12,5,2,1,0,1

Evidentemente, os gols estão dispostos pela ordem de time e não da própria contagem. São vinte times e apenas um deles efetivamente marcou 11 gols nessa semana: o 12.º time da tabela. No caso de dados não estruturados como estes, o único modo de encontrar a informação que você deseja consiste em examinar o primeiro elemento e verificar se corresponde ao valor 11; se não, repetir o procedimento com o segundo elemento, e assim por diante até o encontro do elemento de valor 11 ou a constatação da inexistência desse valor na tabela.

Analisando esses dados, vemos que há um total de vinte escores que variam de 0 a 12. Esse exemplo é simples e, mesmo que tenhamos de examinar cada item, logo descobriremos que o valor 11 é o 12.º elemento apresentado na matriz. Porém, o que acontecerá se houver milhares de elementos constituindo uma grande matriz? O exame de um grande número de itens de dados não ordenados tornará o programa muito lento e inconveniente.

A solução está em ordenar primeiramente os dados, de modo que as buscas sejam feitas com rapidez. Eis aqui outra vez a matriz de contagens, agora disposta em ordem numérica:

0,0,0,1,1,1,1,1,2,2,2,2,2,4,4,5,6,9,11,12

Sabendo que o total de times é vinte, o modo mais rápido de determinar a posição do escore desejado consiste em dividir a matriz em duas partes e buscar unicamente a parte em que é provável que o número procurado se encontre. Lembre-se: a busca em grandes quantidades de dados tem a possibilidade de tomar muito mais tempo do que as operações aritméticas simples, como a divisão de um número por dois. O algoritmo para localização do escore se apresentará da seguinte forma:

```
Encontrar a matriz que contém os escores
Ler o número que desejamos encontrar
Determinar a extensão da matriz
Determinar o ponto intermediário da matriz
Executar um loop até o número ser localizado
    Se o item no ponto intermediário for igual
    ao número que estamos buscando, então o
    número foi localizado
    Se não, verificar se o número procurado é
    maior ou menor que o número no ponto
    intermediário
    Se o número buscado for maior que o
    número no ponto intermediário, então
    determinar o ponto intermediário da parte
    superior da matriz
    Se o número exigido for menor que o
    número no ponto intermediário, então
    determinar o ponto intermediário da parte
    inferior da matriz
    (Repetir esse procedimento até que o
    número seja localizado)
```

Esse procedimento pode assumir a seguinte formalização:

```
INICIAR
  Encontrar a matriz das contagens
  INPUT (FORNECER) O NÚMERO (buscado)
  Executar um LOOP até que o número seja
  localizado
   IF (SE) O NÚMERO = (ponto intermediário)
    THEN (ENTÃO) registrar a posição do ponto
    intermediário
    ELSE (CASO CONTRÁRIO)
      IF (SE) O NÚMERO > (ponto intermediário)
        THEN (ENTÃO) encontrar o ponto
        intermediário da metade superior
        ELSE (CASO CONTRÁRIO) encontrar o
        ponto intermediário da metade inferior
      ENDIF
    ENDIF
  ENDLOOP
```

```
    SE O NÚMERO tiver sido localizado
      THEN (ENTÃO) PRINT (IMPRIMIR) a posição do
      ponto intermediário
      ELSE (CASO CONTRÁRIO) PRINT (IMPRIMIR)
      "NÚMERO NÃO ENCONTRADO"
    ENDIF
  END
```

Se você refletir sobre este programa em pseudolinguagem, vai notar que ele afinal localizará o número em questão, se este se encontrar na matriz. Desenvolveremos esta pseudolinguagem até o ponto em que possamos chegar a um programa coerente. Este processo de pesquisa por subdivisões sucessivas é denominado "busca binária".

Para você experimentar, apresentamos um programa baseado em BASIC ainda que em pseudolinguagem. A finalidade do programa é criar uma matriz e ler os escores, a partir de uma instrução DATA; a tarefa seguinte é indicar a pesquisa do escore. Se este for encontrado, o programa imprimirá o elemento da matriz em que o número for encontrado.

```
10 REM PROGRAMA PARA LOCALIZAR UM
   NÚMERO EM UMA MATRIZ
20 DIM ESCORES(20)
30 FOR Z = 1TO 20
40 READ ESCORES(Z)
50 NEXT Z
60 DATA 0,0,0,1,1,1,1,1,2,2,2,2,2,4,4,5,6,9,11,12
70 LET L = 20
80 LET BTM = 1
90 LET TP = L
100 INPUT "ENTRAR ESCORE ";N
110 FOR Z = 0 TO 1 STEP 0
120 LET L = TP - BTM
130 LET MD = BTM + INT(L/2)
140 IF N = ESCORES(MD) THEN LET Z = X
150 IF N > ESCORES(MD) THEN LET BTM = MD
160 IF N < ESCORES(MD) THEN LET TP = MD
170 NEXT Z
180 PRINT "O ESCORE ESTÁ NO ELEMENTO
    NO. ";MD
190 END
```

Observe ainda que a variável X tem de receber um valor inicial, de acordo com as exigências de sua máquina particular (como os compatíveis Sinclair, por exemplo).

Se os dados presentes em um arquivo ou matriz tiverem uma distribuição normal, como acontece com as listas telefônicas, nas quais os nomes são distribuídos de acordo com a ordem alfabética, a busca binária será um método eficiente de encontrar um determinado item. Todavia, não é o mais eficiente e há mesmo algoritmos alternativos que podem encontrar dados de modo mais rápido. Tal procedimento constitui a técnica de "informação não significativa", na qual o programa realiza um cálculo aproximado da posição de entrada, refinando-o até que essa posição seja encontrada. Entretanto, tais métodos vão além dos propósitos deste curso e o método de busca binária é suficiente para o que precisamos.

## Exercícios

Se você processar este programa, ele funcionará, desde que seja fornecido um escore que esteja presente na tabela. Fornecendo um escore de valor 3, por exemplo, que não se encontra na tabela, o programa não chegará a se encerrar e não será apresentada nenhuma mensagem de erro. Se você digitar o número 12, que se encontra na tabela, o programa não poderá localizá-lo. O programa também supõe que todos os números na tabela ordenada serão diferentes, mas, como você pode observar pela apresentação dos dados, vários números se apresentam mais de uma vez. O programa não detecta essa eventualidade nem registra todas as posições em que o número aparece.

Sua tarefa consiste em:

1. Analisar o programa e descobrir por que ele não pode localizar o escore de valor 12.
2. Alterar uma linha do programa para corrigir essa deficiência.
3. Determinar o motivo por que o programa não pode lidar com números que não se encontram na tabela e criar um procedimento que supere essa deficiência.

Na página 235 deste nosso curso apresentamos uma série de exercícios de revisão a fim de ajudá-lo a avaliar seu progresso no curso de programação em BASIC. Veja as soluções na página 280.

## A propósito...

| Comando/Instrução/Função | Ação/Resultado | MS BASIC | Applesoft | TRS-80 | Sinclair |
|---|---|---|---|---|---|
| CHR$(n) | Dá o caractere código ASCII n | + | + | + | + |
| CINT(x) | Converte x em um inteiro com arredondamento | + | | + | |
| CIRCLE(x,y),r | Desenha um círculo com centro (x,y) e raio r | + | | | |
| CLEAR | Zera as variáveis do programa | + | + | + | + |
| CLOAD "arquivo" | Carrega um programa da fita | | | + | |
| CLOSE#arquivo | Fecha um arquivo | + | + | | |
| CLS | Limpa a tela | + | | + | + |
| CODE(x$) | Dá o código ASCII do primeiro caractere de x$ | | | | + |
| COLOR=x | Define a cor a ser usada em gráfico | | + | | |
| COMMON lista de variáveis | Passa variáveis a um programa encadeado | + | | | |

# Dicas sobre o som

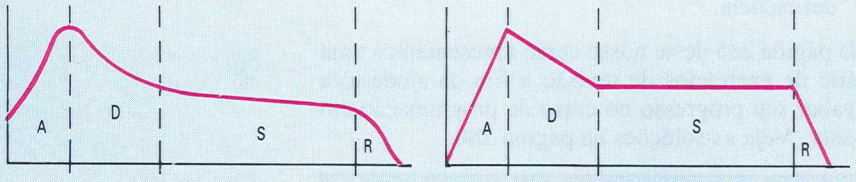
## Continuamos a explicar a música por computação.

Como parte de nossa série sobre a produção de sons, vamos ver agora alguns dos aspectos mais desenvolvidos nos microcomputadores.

### Geradores de envelope

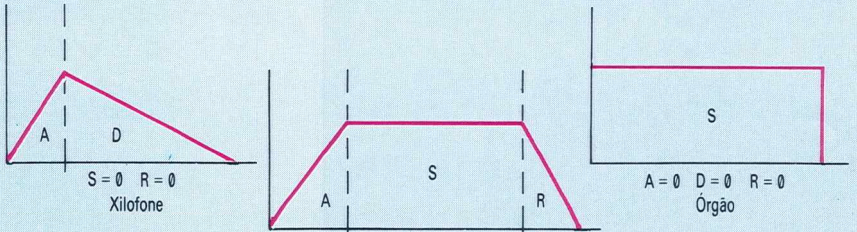
O envelope de um som é o padrão das variações de volume, desde o instante da emissão do som até o momento em que se extingue. Envelopes semelhantes aos de uma nota de piano e de outros instrumentos estão ilustrados abaixo. Os envelopes geralmente se dividem em quatro partes, chamadas Attack (ataque), Decay (decaimento), Sustain (sustentação) e Release (liberação). A sigla ADSR é referente a essas quatro partes. No caso do piano, o volume se eleva rápido ao nível máximo, quando se pressiona a tecla (ataque); a seguir, decresce lentamente (decaimento) para um volume mais ou menos constante (nível de sustentação), enquanto se mantém a tecla pressionada; e, por fim, decai rapidamente para zero (liberação), quando a tecla deixa de ser pressionada. Observe que a sustentação é um nível de volume, enquanto ataque, decaimento e liberação representam intervalos de tempo. O dispositivo que controla os quatro aspectos do envelope é chamado gerador ADSR.

Aliás, qualquer dispositivo que ligue e desligue um som é uma espécie de gerador de envelope.




PIANO NATURAL — PIANO ELETRÔNICO

Na maioria dos computadores, para essa finalidade, não há nada que seja sofisticado: apenas um gerador de "bips", que consiste em um simples interruptor para ligar o som a um volume constante por determinado tempo. No exemplo acima, os elementos A, D e R equivalem a zero.




# Como criar imagens

## Animação simples pelos comandos em BASIC.

Tendo sido vistos os princípios básicos da produção de gráficos por computador, passamos agora a examinar os procedimentos para obtermos animação simples. Primeiro, é necessário detalhar os meios de fazer os caracteres aparecerem na tela e depois controlar seu posicionamento usando um programa em BASIC. Há dois métodos principais a que o programador pode recorrer: o comando POKE e o comando PRINT, com as funções a eles vinculadas.

O comando POKE coloca números em qualquer posição específica da memória. Há um conjunto especial de posições de memória no interior de cada computador, cada qual relacionada a uma determinada posição do caractere na tela. Em telas comuns, de 25 linhas por 40 colunas, há 1.000 posições reservadas para esta finalidade. Cada posição possui um número que corresponde a um caractere particular no conjunto de caracteres da máquina, que pode ser o código standard ASCII do caractere (ver p. 214) ou um código designado pelo fabricante da máquina. Além dessas posições de códigos de caracteres, há geralmente outro conjunto de posições que contém informação sobre a cor do caractere apresentado em qualquer posição da tela.

No Commodore 64, por exemplo, há pouquíssimos comandos para gráficos em BASIC que auxiliam o programador, e o comando POKE é com freqüência utilizado para criar imagens na tela. Os endereços das posições que contêm códigos de caracteres vão de 1024 a 2023, e as posições que contêm informação sobre as cores têm endereços que vão de 55296 a 56295. No Commodore, o caractere A tem o código de tela 1 e a cor preta é representada pelo código de cor 0; desse modo, os comandos exigidos para colocação de uma letra A na cor preta, no canto esquerdo superior da tela são:

```
10 POKE 1024,1
20 POKE 55296,0
30 END
```

Modificações simples podem ser feitas para apresentar uma linha de letras A na cor preta, na primeira linha superior da tela:

```
10 FOR X = 0 TO 39
20 POKE 1024+X,1
30 POKE 55296+X,0
40 NEXT X
50 END
```

As letras A são produzidas pelo loop FOR-NEXT, que aumenta os endereços das posições de caracteres e cores em uma unidade de cada vez. Se incorporarmos um comando que elimine uma letra A anterior, toda vez que uma nova letra A for criada, a letra

parecerá mover-se na tela: será uma forma elementar de animação.

O código de caractere do Commodore para um espaço é 32. É necessário apenas que o programador o coloque na posição adequada, isto é, atrás da nova letra A que será apresentada. Insira a seguinte linha no programa acima:

```
35 POKE 1024+X,32
```

Ou então:

```
15 POKE 1024+(X-1),32
```

O armazenamento de números em posições através do comando POKE, para a produção de gráficos, é um procedimento trabalhoso. Esse método, provavelmente, será mais bem empregado na criação de fundos estáticos, mediante as instruções READ e DATA, para dar entrada aos códigos de caracteres antes de armazená-los na posição correta pelo comando POKE.

A maioria dos microcomputadores tem muitos comandos para gráficos como parte do conjunto standard de instruções em BASIC, o que permite ao usuário criar imagens coloridas e atraentes, apenas com algumas instruções simples. Os comandos para construção de formas geométricas com alta resolução geralmente fazem parte do equipamento. Com essas instruções, o usuário pode formar uma trama de pontos na tela e uni-los com linhas retas; desenhar quadrados, arcos e círculos; e colorir o interior das figuras desenhadas.

É muitas vezes um procedimento simples retraçar as formas anteriores com a mesma cor do fundo, o que resulta no desaparecimento dessas formas. O traçamento, sua eliminação e o retraçamento em nova posição, realizados de modo rápido, são, de fato, a base da animação simples de gráficos. O realismo da ação depende em grande parte da rapidez com que o processo é realizado. Os sprites são muito mais eficazes porque não exigem a eliminação do traçado à medida que a forma muda de posição, e isto aumenta bastante a velocidade em que parecem se mover. Na verdade, o desenvolvimento dos sprites torna possível, pela primeira vez, escrever jogos tipo fliperama em BASIC, mas, anteriormente, o código de máquina era fundamental.

Os princípios de movimentação dos gráficos podem ser empregados em combinação com programas simples em BASIC. Muitos microcomputadores têm comandos que permitem ao usuário imprimir posições específicas na tela, tais como o comando PRINT AT e outros.

**Traçando uma diagonal**

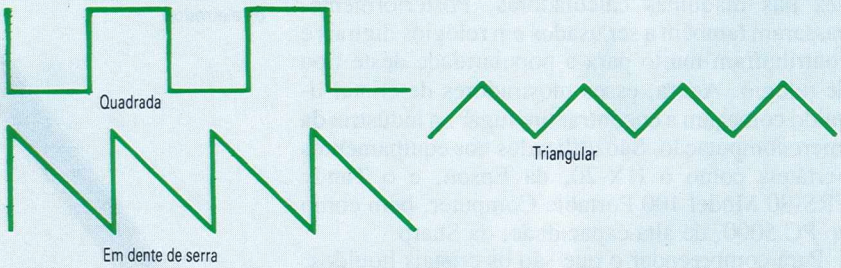
Eis um programa curto na linguagem BASIC compatível com os micros tipo Apple (Unitron, Maxxi, Exato, Micro Engenho e outros) que traça uma linha diagonal à tela, no modo de baixa resolução. As cores geradas em cada ponto são aleatórias:

```
10 REM LINHA DIAGONAL BAIXA RESOLUCAO
20 GR
30 COLOR = RND(16)*16
40 FOR X = 0 TO 39
50 PLOT X,X
60 NEXT X
70 GOTO 30
```

## Forma da onda

A forma da onda é o "formato" ou contorno repetitivo do sinal produzido por um oscilador (ver p. 247) e que dá ao som sua característica própria. Dois instrumentos diferentes tocando notas na mesma altura não produzem sons iguais e isto, em parte, acontece porque as formas das ondas são diferentes. As formas de ondas mais comuns são quadradas (ou de pulso), triangulares e em dente de serra, como é mostrado na ilustração.

A maioria dos microcomputadores proporciona uma única forma de onda, geralmente do tipo pulso. Por isso, nota-se um áspero som sintetizado.




O Commodore 64 é atualmente o computador mais interessante em produção de música, porque permite a escolha entre as três formas básicas de ondas em cada um dos três osciladores. As formas de onda podem ser modificadas com o uso de filtros, que alteram a tonalidade, de modo muito semelhante aos controles grave/agudo dos equipamentos hi-fi e têm o efeito de tornar o som mais suave. Ainda mais útil é a capacidade de trocar os conjuntos de filtro durante a execução da nota. Isso permite a simulação de sons naturais de modo mais aproximado e a produção de sons não-naturais mais interessantes.

## Ruído

O ruído é um tipo complexo de som resultante de vibrações não previsíveis. A audição humana não consegue distinguir entre padrões repetitivos e, assim, não pode discernir nenhuma tonalidade específica. Imagine alguns sons como os da chuva, do vento e do trovão. Esses ruídos não soam de forma igual porque são uma combinação de ruídos (imprevisíveis) em que há a predominância de alguns tons. A maioria dos microcomputadores com recursos para ruídos permite alguma forma de modulação do ruído ou a sua mistura com notas puras. Os efeitos possíveis variam do "vento zunindo" a explosões violentas.

## Saída

A saída é feita, geralmente, através do alto-falante do televisor. Se for este o caso, você poderá ligar o televisor a seu equipamento de hi-fi, através de um videocassete. Alguns computadores, entretanto, só podem emitir sons por um pequeno alto-falante embutido. Nestes, é impossível obter sons de boa qualidade sem fazer uma adaptação do hardware ou adquirir equipamento acessório, externo. Seu computador poderá ter uma saída adequada para ligação direta com o aparelho de hi-fi, o que tornará compensador o esforço exigido na produção de configurações sonoras complexas.

# Claro como cristal

## Displays de cristal líquido, comuns em calculadoras e relógios, começam agora a aparecer em computadores.

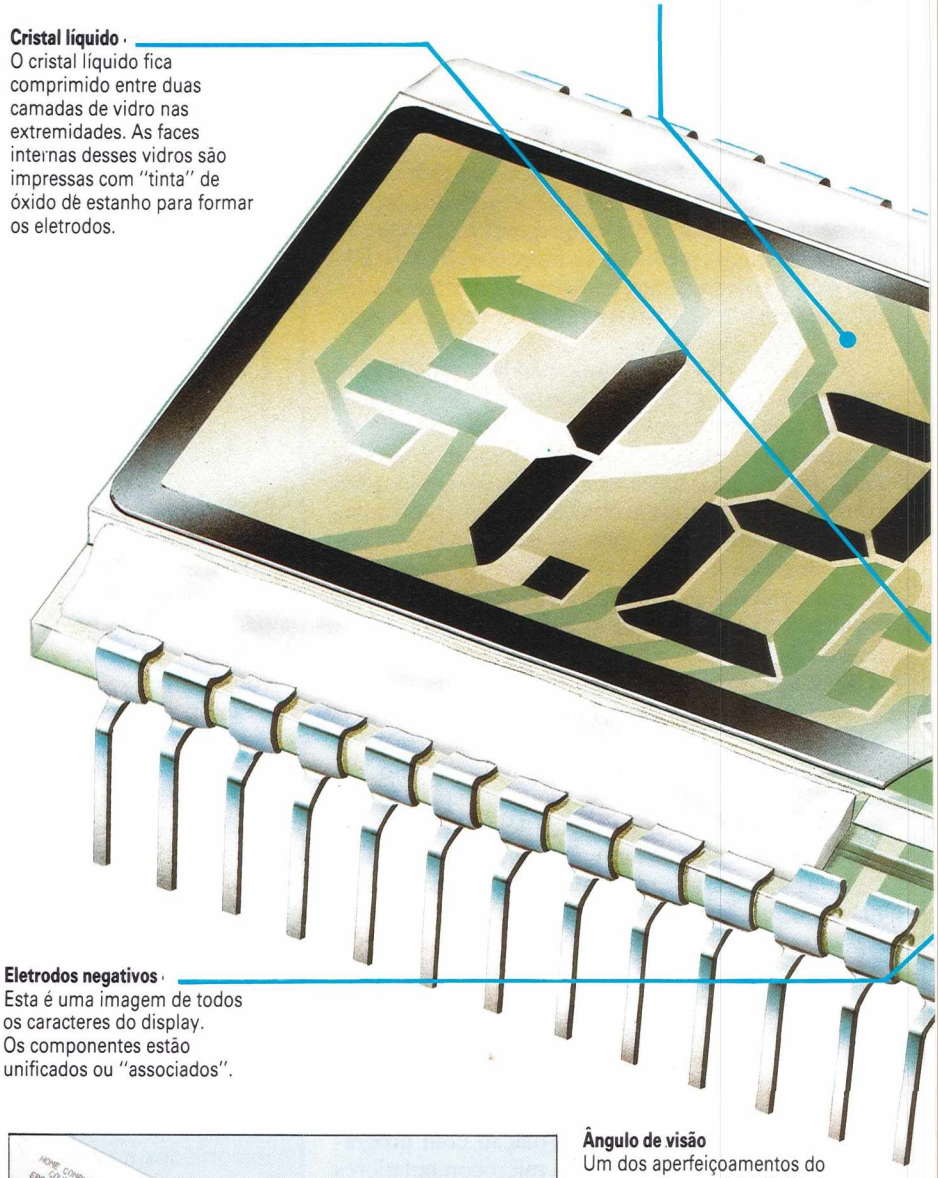
Liquid Crystal Displays (LCDs) existem no mercado desde 1973, quando apareceram pela primeira vez nas máquinas calculadoras. Posteriormente, passaram também a ser usados em relógios digitais e contribuíram muito para a popularidade deste tipo de relógio. Agora, esses mostradores de cristal líquido começam a encontrar um lugar na indústria da microcomputação. São utilizados nos equipamentos portáteis como o HX-20, da Epson, e o Tandy TRS-80 Model 100 Portable Computer, bem como no PC 5000, de alta capacidade, da Sharp.

Para compreender o que são os cristais líquidos, precisamos antes lembrar que toda matéria varia seu estado físico em função da temperatura e da pressão: de sólido (ou cristalino), pode transformar-se em líquido e gasoso. Apenas no estado sólido existe algum alinhamento regular das moléculas da substância. Mas há exceções: no caso de pouquíssimas substâncias esse alinhamento regular se mantém parcialmente no estado líquido. Essas substâncias, cuja constituição é um segredo industrial rigorosamente guardado, denominam-se cristais líquidos.

Até meados da década de 70, os mostradores ou imagens de calculadoras e relógios eram constituídos por diodos emissores de luz (LEDs, Light Emitting Diodes), dispostos de modo a formar um aspecto angular de números ou de letras. Porém, os mostradores LED apresentam inconvenientes: exigem quantidades consideráveis de energia e têm tamanho relativamente grande.

Na busca de outros métodos de apresentação de dados informativos foi descoberto que a disposição das moléculas de cristais líquidos pode ser alterada por corrente elétrica; e, mais ainda, essa alteração é apenas local. Reconhecido este princípio, tornou-se possível estabelecer um modo de apresentar os dados. Para isso, o primeiro estágio consiste na formação de eletrodos na forma de um caractere nas faces internas de duas lâminas de vidro. Uma camada muito fina de cristal líquido fica comprimida entre as lâminas e nela aplica-se uma corrente elétrica. Sob a claridade comum, nada parece acontecer, mas, quando filtros polarizadores (ver ilustração) são aplicados nas partes posterior e anterior, e a estrutura toda é montada contra um fundo refletor, produz-se o efeito desejado — um caractere claramente definido, com um fundo neutro.

O processo de definição dos caracteres exige que a luz atravesse o primeiro filtro e seja, desse modo, polarizada verticalmente; em seguida, desviada em ângulo de 90° e, com isso, bloqueada no filtro posterior. Desse modo, a área do cristal líquido em que a corrente elétrica foi aplicada aparece como uma área escura sólida. É exatamente este o método em-

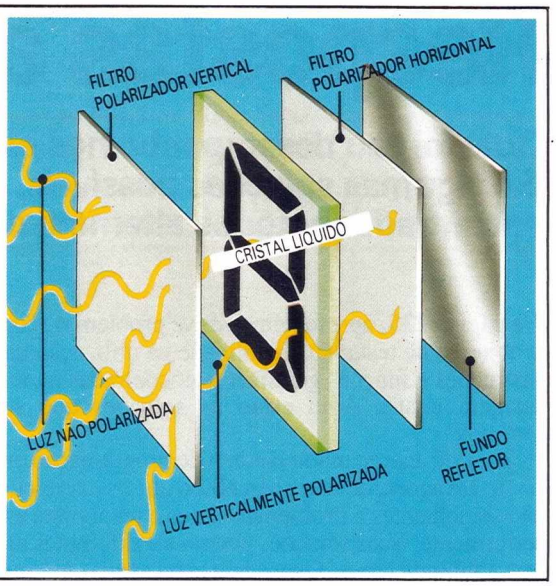






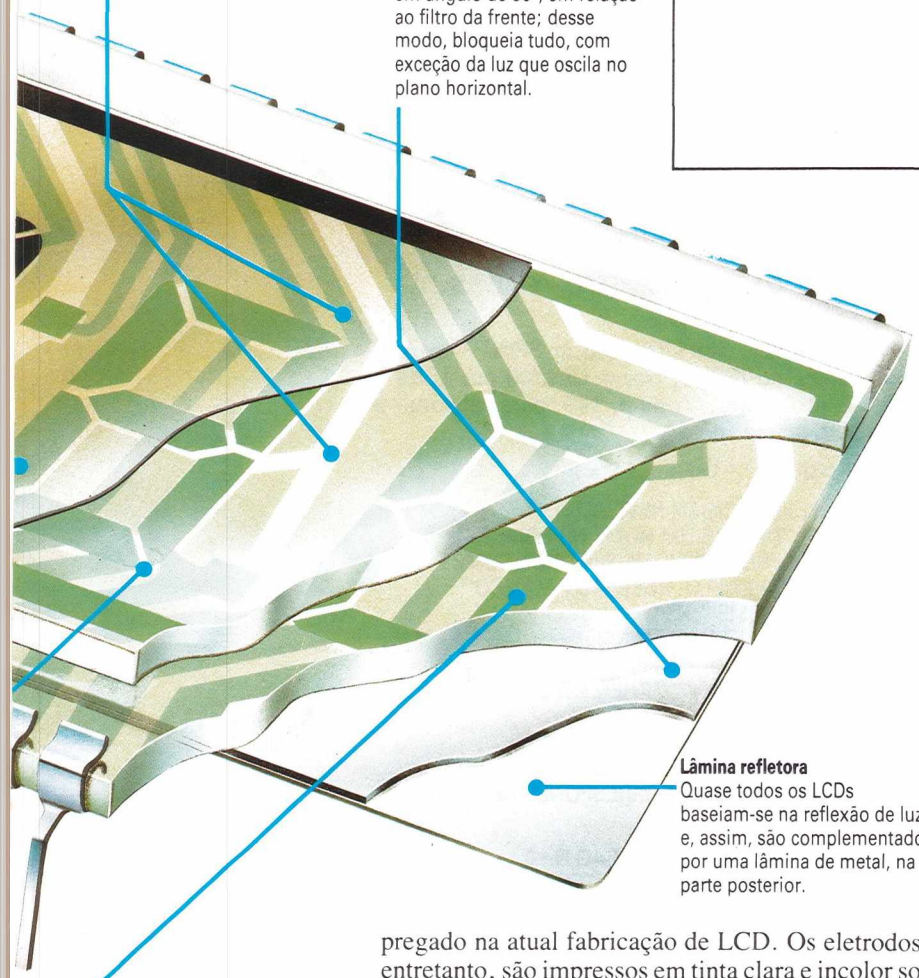
**Ângulo de visão**
Um dos aperfeiçoamentos do computador portátil HX-20, da Epson, é o ajustador do ângulo de visão. Os cristais líquidos são compostos de moléculas estreitas, que possuem pólos magnéticos posicionados no centro de suas laterais alongadas. A aplicação de corrente elétrica em toda a sua extensão faz com que tendam a inverter as extremidades, ao mesmo tempo que a estabilidade natural procura manter a posição original. Quanto maior a corrente, maior a tendência à inversão.

**Polarização**
O filtro situado na frente do LCD só admite a passagem de raios luminosos com oscilação eletromagnética orientada verticalmente. A seguir, os cristais líquidos distorcem a luz polarizada em um ângulo de 90°, permitindo que ela atravesse o filtro de trás (horizontal) e seja refletida em sentido contrário. Quando um segmento do LCD é energizado, cessa a distorção da polarização, o que resulta em uma imagem preta.




**Canais de conexão**
Cada eletrodo é ligado a seu circuito propulsor por meio de uma camada quase invisível de tinta condutora, distribuída na superfície do vidro posterior ou anterior.

**Polarização horizontal**
O filtro atrás do mostrador possui um eixo de polarização em ângulo de 90°, em relação ao filtro da frente; desse modo, bloqueia tudo, com exceção da luz que oscila no plano horizontal.



**Lâmina refletora**
Quase todos os LCDs baseiam-se na reflexão de luz e, assim, são complementados por uma lâmina de metal, na parte posterior.

**Eletrodos positivos**
Nesta representação dos caracteres, cada componente está separado e é endereçado individualmente pelos circuitos condutores.

pregado na atual fabricação de LCD. Os eletrodos, entretanto, são impressos em tinta clara e incolor sobre a superfície do vidro e secam até atingir a quase invisibilidade.

Devido à exigência de produção de uma multiplicidade de caracteres na mesma matriz, o método original — barras curtas combinadas para a composição de caracteres angulares — ainda está em uso, embora as matrizes de ponto LCD estejam progressivamente se tornando comuns. As capacidades gráficas se desenvolveram muito, em virtude da possibilidade de endereçar cada ponto isolado: podem ser produzidos tanto formas contínuas como caracteres gráficos convencionais.

Teoricamente, é possível utilizar LCDs endereçados através de matrizes de pontos com a função de tela de receptor de televisão ou empregá-los como monitor de computador. O principal inconveniente desse método consiste na falta de variação no contraste. Este foi um dos problemas pesquisados durante o desenvolvimento dos vários televisores de vídeo plano que agora são produzidos. O tamanho dos elementos individuais na matriz deve ser reduzido para o tamanho aproximado de um pixel no tubo de raios catódicos, a fim de se obter resolução adequada. Outra alternativa exige a utilização de uma série de LCDs ajustados em um conjunto, que operem simultaneamente. Este método é com freqüência empregado na apresentação de dados informativos complexos, quando há limitação de espaço.

O tempo de resposta de um LCD de alta qualidade, à temperatura normal de operação (20°C), é de cerca de 70 milissegundos para passagem de neutro a preto e de outros 80 milissegundos para retornar ao estado neutro. Esse total de 150 milissegundos requerido pelo LCD coloca este sistema em posição desfavorável, comparado aos 0,00025 de milissegundo da resposta do tubo de raios catódicos. Entretanto, o LCD possui várias vantagens. Uma delas já foi mencionada: o tamanho reduzido; mas o baixo consumo de energia talvez seja a propriedade mais significativa: o LCD consome apenas 10 microwatts por centímetro quadrado de imagem.

Há um efeito interessante na variação da voltagem que atravessa o cristal líquido. A torção das moléculas aumenta e diminui, de acordo com a intensidade da voltagem. É utilizado esse recurso com bons resultados, por exemplo, no computador portátil HX-20 da Epson, dando à imagem um ajustamento conforme o ângulo de visão.

Outra vantagem se evidencia sob a luz solar intensa. O contraste da imagem do tubo de raios catódicos diminui muito em tal circunstância, porém, como os cristais líquidos são percebidos justamente por causa da *ausência* de luz neles refletida, os LCDs parecem tornar-se mais contrastantes e, desse modo, mais legíveis com o aumento da luminosidade externa.

Embora com apenas dez anos de existência, a tecnologia do LCD já tem aplicações significativas nos produtos tradicionalmente reservados aos tubos de raios catódicos. À medida que aumentam as exigências da microminiaturização, espera-se que esta tecnologia se desenvolva ainda mais.

# Respostas aos exercícios

## Qual foi seu desempenho nos exercícios da página 235? Eis algumas soluções possíveis, embora você possa ter encontrado métodos alternativos, com os mesmos resultados.

Na página 235, apresentamos nove problemas, com o objetivo de testar sua habilidade na utilização das instruções e funções geralmente encontradas na linguagem BASIC. Aqui estão as soluções que sugerimos.

Se você acompanhou o curso de programação em BASIC até agora, deve ter percebido que os exercícios são semelhantes a problemas vistos e resolvidos anteriormente. Suas soluções podem ser diferentes das que aqui indicamos. Quase nunca há apenas um método de resolver um problema; o seu pode ser tão bom ou melhor que o nosso.

Se achou difíceis as questões e as soluções, leia novamente a partir da página 149 e estude as soluções indicadas, à medida que for avançando. Se você resolveu bem os exercícios 2, 4, 6 e 8, compreendeu a maior parte das lições apresentadas.

```
100 REM EXERCICIO DE REVISAO N.º 1
200 INPUT "DIGITAR QUALQUER NUMERO";A
300 INPUT "DIGITAR OUTRO NUMERO";B
400 LET C = A + B
500 PRINT "A SOMA DOS NUMEROS E ";C
```

Nos computadores compatíveis Sinclair, colocar um PRINT ''mensagem'' antes do INPUT.

```
100 REM EXERCICIO DE REVISAO N.º 2
200 LET A$ = "PRIMEIRAPALAVRA,"
300 LET B$ = "SEGUNDAPALAVRA"
400 LET C$ = A$ + B$
500 PRINT C$
```

```
100 REM EXERCICIO DE REVISAO N.º 3
200 INPUT "DIGITAR QUALQUER PALAVRA";P$
300 LET C = LEN(P$)
400 PRINT "A PALAVRA QUE VOCE DIGITOU TEM ";C;
    "CARACTERES"
```

```
100 REM EXERCICIO DE REVISAO N.º 4
200 PRINT "PRESSIONAR QUALQUER TECLA"
300 FOR C = 0 TO 1 STEP 0
400 LET A$ = INKEY$
500 IF A$ < > " " THEN LET C = 2
600 NEXT C
700 PRINT "O VALOR ASCII DE " ;A$; " E " ;ASC(A$)
```

Veja a seção ''A propósito...'', nas pp. 175 e 215. Nos computadores Sinclair, substituir a linha 700 por:

```
700 PRINT "O VALOR ASCII DE " ;A$; " E " ; CODE(A$)
```

```
100 REM EXERCICIO DE REVISAO N.º 5
200 INPUT "DIGITAR QUALQUER PALAVRA";P$
300 LET U$ = RIGHT$(P$,1)
400 PRINT "O ULTIMO CARACTERE DA PALAVRA
    E ";U$
```

Veja ''A propósito...'', na p. 149.

Este exercício nos compatíveis Sinclair é resolvido assim:

```
100 REM EXERCICIO DE REVISAO N.º 5
150 PRINT "DIGITAR QUALQUER PALAVRA"
200 INPUT P$
250 LET N = LEN P$
300 LET U$ = P$(N)
400 PRINT "O ULTIMO CARACTERE DA PALAVRA
    E ";U$
```

```
100 REM EXERCICIO DE REVISAO N.º 6
200 PRINT "DIGITAR UM NOME NA FORMA:"
300 PRINT "PRIMEIRONOME SEGUNDONOME"
400 PRINT "EXEMPLO: ROBERTO PACHECO"
500 INPUT "NOME ";N$
600 LET S = 0:LET L = LEN(N$)
700 FOR P = 1 TO L
800 IF MID$(N$,P,1) = " " THEN LET S = P
900 NEXT P
950 PRINT "O ESPACO CORRESPONDE AO ";S; "0.
    CARACTERE"
```

Nos compatíveis Sinclair, colocar PRINT ''mensagem'' antes do INPUT, trocar LEN(N$) por LEN$ e veja ''A propósito...'', na p. 149, para substituir a linha 800, acima, por:

```
800 IF N$(P) = " " THEN LET S = P
```

```
100 REM EXERCICIO DE REVISAO N.º 7
200 PRINT "DIGITAR UM NOME NA FORMA:"
300 PRINT "PRIMEIRONOME SEGUNDONOME"
400 PRINT "EXEMPLO: ROBERTO PACHECO"
500 INPUT "NOME ";N$
600 LET S = 0:LET L = LEN(N$)
700 FOR P = 1 TO L
800 IF MID$(N$,P,1) = " " THEN LET S = P
900 NEXT P
920 IF S = 4 THEN LET X$ = "QUARTO"
940 IF S = S THEN LET X$ = "QUINTO"
950 PRINT "O ESPACO CORRESPONDE AO ";X$;
    "CARACTERE"
```

```
100 REM EXERCICIO DE REVISAO N.º 8
200 INPUT "DIGITAR UMA SENTENCA ";S$
300 LET C = 1
400 FOR P = 1 TO LEN(S$)
500 IF MID$(S$,P,1) = " " THEN LET C = C + 1
600 NEXT C
700 PRINT "A SENTENCA QUE VOCE DIGITOU
    TEM ";C;" PALAVRAS"
```

Veja ''A propósito...'', na p. 149. Nos computadores compatíveis Sinclair, substituir a linha 500, acima, por:

```
500 IF S$(P) = " " THEN LET C = C + 1
```

```
100 REM EXERCICIO DE REVISAO N.º 9
200 FOR C = 128 TO 255
300 PRINT "O CARACTERE NO. ";C;"=";CHR$(C)
400 REM UM PEQUENO INTERVALO AQUI
500 FOR D = 1 TO 500
600 NEXT D
700 REM ENCERRAMENTO DO INTERVALO
800 NEXT C
```

Ver as observações anteriores para os micros tipo Sinclair.

# Seu fiel servidor

## Robôs industriais reconhecem agora visualmente os objetos e aprendem novas tarefas imitando movimentos humanos.

O termo "robô" deriva da palavra tcheca *robota*. Foi cunhado pelo escritor Karel Capek para sua peça teatral escrita em 1920, *R.U.R.* (Rossum's Universal Robots), sendo subseqüentemente adotado com entusiasmo pelos escritores de ficção científica. Apesar dos muitos relatos literários sobre suas capacidades, os robôs não passam de um prolongamento eletromecânico do computador, com todas as limitações e possíveis falhas daí decorrentes.

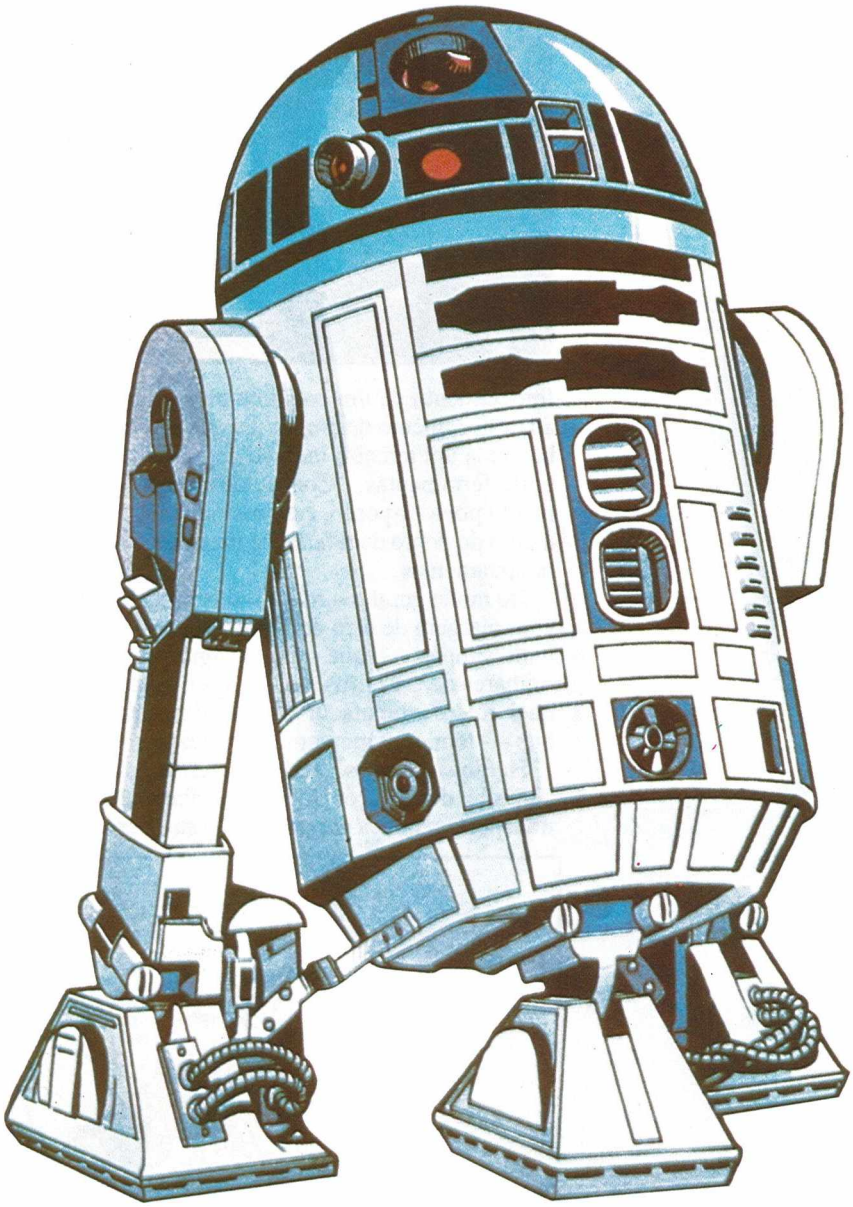
Suas origens remontam à indústria de base da década de 50, quando pela primeira vez empregaram-se máquinas operatrizes com controle numérico. Essas primeiras tentativas eram rudimentares: máquinas controladas por fitas de papel de cinco furos (do tipo empregado em máquinas de telex) que podiam, no máximo, deslocar uma ferramenta de um ponto para outro, próximo ao objeto em que estavam trabalhando.

O estágio seguinte de desenvolvimento desses mecanismos foi o acréscimo da capacidade de substituição de ferramentas durante a realização de um trabalho.

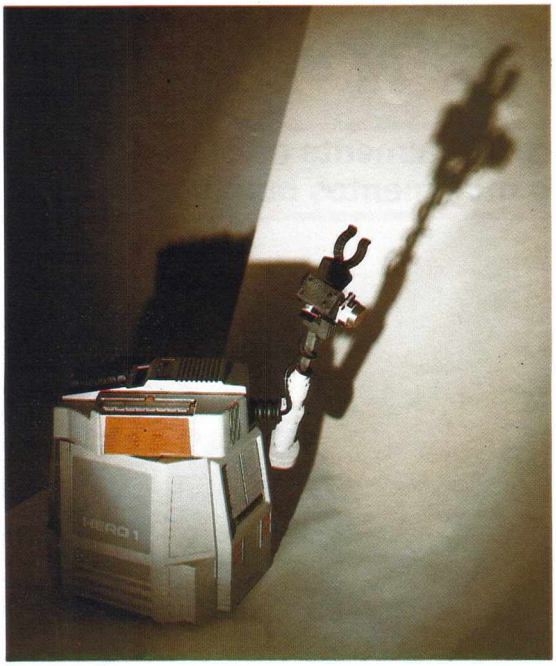
Mesmo com esse aperfeiçoamento, cada tipo de máquina podia realizar apenas uma única tarefa: o torno permanecia um torno, embora tendo a possibilidade de executar toda a seqüência de operações envolvidas em determinado processo. Na mesma época, braços e mãos dirigidos por controle remoto estavam sendo desenvolvidos para executar tarefas em ambientes perigosos, no fundo do mar, por exemplo, ou no manejo de materiais radiativos em laboratórios. Esses dispositivos para manipulação não passavam de prolongamentos das próprias mãos do operador, mas os computadores logo viriam a ser usados para controlá-los diretamente. A denominação mais adequada para os robôs que foram então desenvolvidos seria "braços robóticos", pois consistem em um porta-ferramentas montado em um braço estendido ou articulado.

Para compreendermos como os robôs são programados, devemos primeiro observá-los na relação com o espaço em que operam. A maioria dos robôs industriais tem posição fixa e, desse modo, seu âmbito de ação corresponde a uma esfera achatada na base; assim, podemos considerar a questão do controle do robô como simplesmente um exercício de geometria tridimensional. O centro do esferóide é a articulação do "ombro" do robô e o raio será a extensão do braço alongado, medida a partir do "ombro" até a ponta dos "dedos" — isto é, o dispositivo para segurar, ou porta-ferramentas. Qualquer ponto nesse espaço pode ser expresso por meio de três coordenadas: por exemplo, a distância norte/sul, leste/oeste e para cima/para baixo, a partir de um "ponto de referência" ou posição zero. Neste caso, as coordenadas são chamadas de cartesianas (relativas à geometria desenvolvida pelo matemático e filósofo francês René Descartes, no século XVII). A posição pode também ser expressa em termos de coordenadas esféricas. Em linguagem simples essa relação seria apresentada, digamos, da seguinte forma: "à distância de 2 m, na direção norte/leste, e 30 graus acima da linha horizontal". Neste caso, o "ombro" do robô é o ponto de referência.

Todavia, a tarefa de programação do robô inclui o



**O herói do cinema**
O simpático robô de *Guerra nas Estrelas* — o R2D2 — era, na verdade, controlado por um operador humano. Sua estrutura reflete a concepção vulgarizada de um robô.



**Robô movido a bateria**
O Hero-1 é um robô movido a bateria completamente auto-suficiente, que combina algumas das funções da tartaruga com a capacidade do braço de um robô. Parece um brinquedo caro. Mas, na verdade, é um sistema de computação altamente flexível em razão de suas próprias características, que incluem recursos avançados, como o sintetizador de voz, os sensores de nível de luz, a entrada para som e (devido ao fato de mover-se) um telêmetro ultra-sônico, que pode também funcionar como detector de movimento.

fornecimento de um conjunto de instruções quanto ao deslocamento de um local a outro e, desse modo, há ainda um terceiro método de posicionamento do porta-ferramentas. Conhecido como posicionamento ponto-a-ponto, esse método exige o deslocamento do ponto de referência juntamente com o porta-ferramentas.

De modo geral, os robôs industriais são precisos, com margem de erro de 1 mm. Mesmo os modelos mais simples — que custam algumas centenas de milhares de cruzeiros e podem ser usados em qualquer microcomputador com saídas paralelas de 8 bits — têm uma margem de erro de 2 mm.

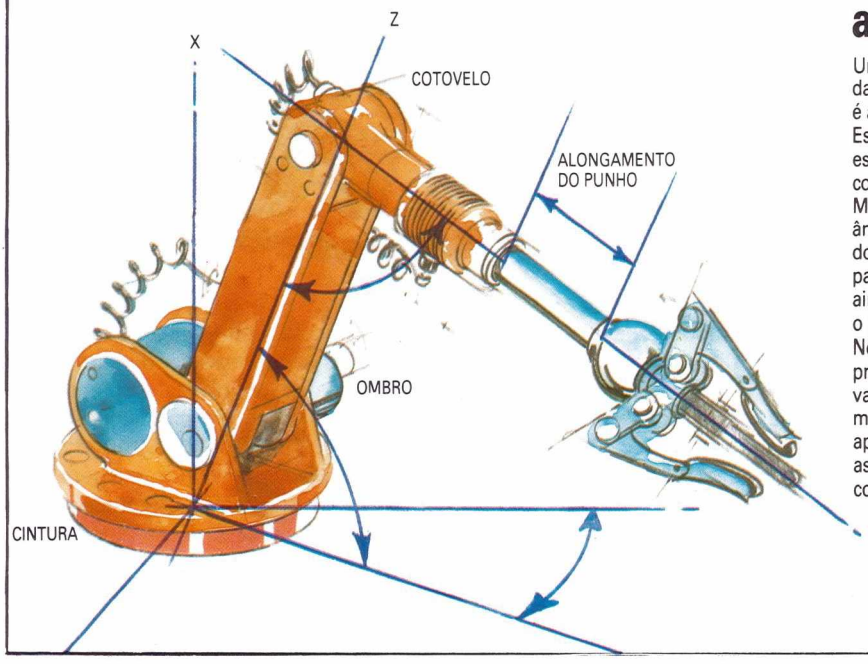
Há dois métodos geralmente empregados para o controle dos braços robóticos. Para aqueles que transportam pouca carga útil, são suficientes os motores que operam por estágios (motores elétricos que se movem a espaços predeterminados, quando recebem corrente elétrica, como os utilizados em unidades de disco, para posicionamento dos cabeçotes de leitura/gravação). Porém, para os braços de robô utilizados em linhas de produção, em que há necessidade do manejo de pesos maiores, é mais comum o emprego de carneiros hidráulicos para mover as várias peças do braço em torno de seus pivôs (os pontos ao redor dos quais as peças giram). É um procedimento bastante simples medir o volume do fluido que passa pelos carneiros hidráulicos e disso deduzir o movimento na outra extremidade, mantendo-se dentro das exigências de precisão operacional.

Os robôs industriais possuem um minicomputador especialmente desenvolvido (ou, nos modelos mais recentes, um microcomputador de alta capacidade) com a única função de controlar o braço e processar uma linguagem de programação especial para essa finalidade. Como a única exigência é indicar coordenadas e emitir comandos simples, como FECHAR A GARRA ou ABRIR A GARRA, a linguagem de programação não contém instruções para lidar com textos. As instruções do programa são fornecidas através de teclas numéricas, ligadas ao computador por um longo "cordão umbilical", de modo que o operador possa mover o braço-robô ao mesmo tempo em que fornece as instruções. As versões mais aperfeiçoadas desses "painéis de controle" incluem um comando joystick de alta precisão.

Outro método de programação, conhecido como "Siga-me", é útil em especial nas tarefas que não exigem a colocação muito precisa da ferramenta, como nas tarefas de pintura por pulverização. Neste caso, o operador segura o porta-ferramentas, levando-o a executar diretamente a tarefa desejada, e assim estará fornecendo a seqüência de movimentos à memória do computador. Com isso, o robô repetirá esses movimentos toda vez que solicitado.

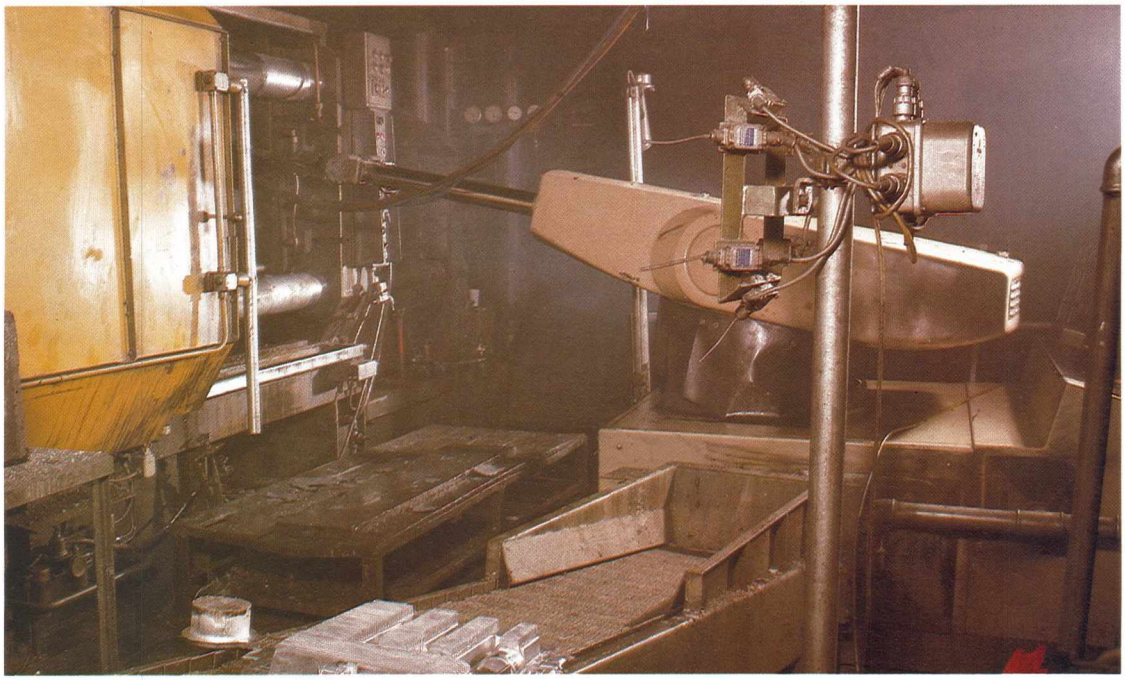
Em todos esses métodos, é a própria posição do

## Movimento angular




Um dos aspectos mais difíceis da programação do braço-robô é a conversão geométrica. Estamos acostumados a especificar as posições pelas coordenadas cartesianas x,y,z. Mas os robôs necessitam de ângulos para as articulações do "cotovelo" e do "ombro", para a rotação da "cintura" e, ainda, para a distância em que o punho deve ser alongado. Nos sistemas simples, os programadores fornecem valores para os quatro movimentos. Os robôs mais aperfeiçoados executam todas as conversões a partir de coordenadas cartesianas.

porta-ferramentas que é definida. O operador não se ocupa com as posições relativas das seções individuais do robô. A linguagem de programação alojada no computador de controle calcula quais devem ser essas posições e também assegura o deslocamento da ferramenta, de um local a outro, pelo menor trajeto possível. A orientação do porta-ferramentas é controlada de forma automática, mantendo-se as posições relativas, tanto a horizontal como a vertical, a menos que haja instruções em contrário. A velocidade do movimento ponto-a-ponto também é automática: o porta-ferramentas é lentamente retirado de



**Legislação de fábrica**
O braço-robô, como o visto aqui em atividade numa fundição, começa a ser usado nos trabalhos insalubres, perigosos e repetitivos da indústria. A limpeza das peças, antes de entrarem em máquina, é um bom exemplo disso. Ao sair do molde, a peça fundida se encontra excessivamente quente para o manuseio humano, sendo posta de lado para resfriamento. Todavia, o robô, não sendo suscetível ao calor, pode enviar a peça imediatamente para a operação seguinte.

onde se encontra, move-se rápido até uma curta distância do ponto de chegada, diminuindo em seguida a velocidade para reencontrar a peça trabalhada no novo local.

Os robôs de que tratamos até agora são capazes apenas de "obediência cega", repetindo com exatidão a tarefa no mesmo local, independente de influências externas. Seu principal emprego é na engenharia mecânica, especialmente na produção de veículos motorizados; organizam-se linhas de produção, nas quais o componente ou o veículo parcialmente montado estão sempre localizados em tempo e espaço precisos. Este fator é fundamental para o bom funcionamento do processo de produção pelo robô, porque, se o componente for colocado em posição incorreta, o robô não adaptará sua atividade adequadamente. Para superar este problema, um sensor pode ser ajustado ao porta-ferramentas. O mais simples dos sensores pode ser um microinterruptor comum de ligar/desligar. Esquemas para eventos acidentais podem ser incorporados ao programa de controle (um comando WAIT, ESPERAR, por exemplo); serão executados se o interruptor não entrar em contato com a peça trabalhada.

Um procedimento alternativo à detecção por pressão poderá exigir o emprego de um fotossensor. Se uma fonte luminosa for posicionada de modo que a peça em montagem a oculte do sensor instalado no porta-ferramentas, este poderá ter seu movimento interrompido antes de atingir o ponto de colisão, ficando em espera (WAIT) até que a peça esteja na posição correta, quando será liberado para continuar a operação. Esse processo também não é inteiramente seguro e, para casos em que se exige total confiabilidade, há o recurso de instalar um sistema de identificação por imagem, que funciona com câmaras de televisão, com dispositivos de carga acoplada (CCD, Charge-Coupled Device). Essas câmaras focalizam a imagem diretamente em um microchip de processamento de matriz (um chip subdividido em uma centena ou mais de fotossensores, cada qual com capacidade de registrar imagens não só em preto e branco, mas também em tons intermediários). Cada um dos sensores requer talvez 1 byte de memória para definir o contraste da escala em cinza. Inicialmente, cada objeto é "fotografado" certo número de vezes e um programa memorizador calcula a média das tonalidades. Por ocasião do processamento, a câmara CCD produz uma imagem do objeto, que é então comparada com a imagem de referência na memória. Se ambas corresponderem, a operação poderá prosseguir. Por esse método é possível verificar se a peça em montagem está presente e em posição perfeitamente correta.

Um outro emprego desse sistema de processamento de imagens está na seleção de componentes misturados. Esse procedimento de "seleção e colocação", como é chamado, está sendo cada vez mais aplicado em robôs de pequeno porte como auxiliar em linhas de produção. Além do seu emprego no próprio processo de produção, os robôs industriais são usados nos estágios de teste e de controle de qualidade, muitas vezes em pares, o que permite maior flexibilidade no posicionamento do produto.

Iniciamos este artigo com uma referência ao robô na literatura de ficção. Poucos casos podem exemplificar melhor a ficção transformada em realidade do que o desenvolvimento dos robôs industriais, e é possível que venham a se tornar os aparelhos autônomos e autodeterminados descritos pela ficção científica. Todavia, isto não se dará enquanto a inteligência artificial permanecer no campo teórico.

# O ressoar do Vic

## Um exame detalhado da produção de som no Vic-20...

Um dos primeiros microcomputadores lançados na Inglaterra foi o Vic-20. Portanto, seus recursos podem parecer um pouco limitados em comparação com equipamentos que surgiram mais recentemente. Além disso, a Commodore não tornou muito fácil a elaboração de programas de som ou música, pois a linguagem BASIC do Vic-20, tanto como a do Commodore 64, não possui comandos especificamente vinculados à produção de som. Todo controle sonoro é obtido por uma série de comandos POKE armazenados nas posições de memória. Este princípio também se aplica ao Commodore 64, e as técnicas aqui descritas para o Vic-20 são úteis ainda ao usuário do Commodore 64. O grau de controle sonoro obtido está restrito ao volume (equivalente ao envelope com A = D = R = 0), à freqüência em três osciladores e a um gerador de ruído. A saída é feita exclusivamente através do alto-falante do televisor. Além disso, devido às imprecisões do método com que o Vic-20 seleciona freqüências, é impossível obter a altura exata para todas as notas da escala musical.

Dispondo só desses recursos, o Vic-20 é pouco útil na produção de música; mesmo assim seus recursos limitados podem ser empregados na criação de "melodias" com dois ou três acordes de notas.

## Controle de som

O Vic-20 é equipado com três osciladores de ondas quadradas e um gerador de ruído. Cada oscilador alcança aproximadamente três oitavas de som, distribuídas nas seguintes freqüências:

| Osc. 1 | Osc. 2 | Osc. 3 | Amplitude da freqüência (Hz) | Oitava |
|---|---|---|---|---|
| ● | | | (65,41-123,47) | 1 |
| ● | ● | | (130,81-246,94) | 2 |
| ● | ● | ● | (261,63-493,88) | 3 |
| | ● | ● | (523,25-987,77) | 4 |
| | | ● | (1046,5-1975,53) | 5 |

Esta distribuição permite ao usuário o alcance de um total de cinco oitavas com a utilização de pelo menos um oscilador em cada uma delas. A oitava três, que se inicia em dó central e possui o padrão de referência A, a 440 Hz, pode ser utilizada com todos os três osciladores.

O controle dos osciladores é executado com a alteração dos conteúdos de cinco posições de memória, da seguinte forma:

| Posição de memória | Oscilador |
|---|---|
| POKE 36874,X | 1 |
| POKE 36875,X | 2 |
| POKE 36876,X | 3 |
| POKE 36877,X | ruído |

Em cada caso, X representa um número inteiro entre 135 e 241 (o valor 0 desliga o oscilador correspondente), que se refere à tabela de equivalentes de valores de notas, na página 73 do manual que acompanha o Vic-20. Antes que a freqüência escolhida possa ser ouvida, o nível do volume deve ser determinado, da seguinte forma:

```
POKE 36878,V
```

onde o valor de V pode ser determinado entre 0 (desligado) e 15 (alto), o que afeta a produção de ruídos e todos os osciladores. Por exemplo:

```
POKE 36874,219:POKE 36875,219:POKE
   36876,219:POKE 36878,7
```

Esse procedimento coloca a referência A (isto é, a nota lá) em 440 Hz, no oscilador 1, em uma oitava acima no oscilador 2 e em outra oitava acima no oscilador 3 — os três osciladores colocados a um volume médio de alcance 7. Lembre-se de armazenar, pelo comando POKE, o valor 0 em cada posição, quando for desligar os osciladores.

## Notas e pausas

Sem uma determinada duração para cada nota e as pausas corretas entre elas, a seqüência torna-se embaralhada. Para simplificar os períodos de "espera", pode-se empregar dois métodos. O primeiro consiste nos loops FOR-NEXT, nos quais a pausa é cronometrada por um loop vazio, como o seguinte:

```
10 POKE 36878,7
20 POKE 36876,203
30 FOR P=1 TO 200
40 NEXT P
50 POKE 36878,0
60 POKE 36876,0
```

Essa seqüência de comandos produz a nota ré # durante a execução de 200 procedimentos FOR-NEXT. Entretanto, esse método depende de uma cuidadosa contagem externa do loop, para se obter precisão. Um modo mais fácil e elegante de determinar as durações e as pausas é usar o relógio embutido no Vic-20, que cronometra em sexagésimos de segundos (jiffys) e pode ser referenciado de dentro de um programa com o emprego da variável TI. Esse procedimento é útil, pois permite a elaboração de um comando para "esperar" durante um período de tempo cronometrado com precisão, como segue:

```
10 POKE 36878,7
20 POKE 36876,203:D=TI
30 IF TI-D<15 THEN 30
40 POKE 36878,0
50 POKE 36876,0
```

Esses comandos tocam as mesmas notas do programa anterior, mas por um período de 15 jiffys (um quarto de segundo). Quando o som é ligado, a variável D recebe o valor TI. A linha 30 conta 15 jiffys antes de prosseguir para a linha 40. As melodias podem ser elaboradas com o emprego do mesmo procedimento para realizar uma pausa antes de tocar uma nota diferente, e assim por diante.

# Esclarecendo o Dragon

## ... e a capacidade gráfica do micro inglês Dragon 32.

O computador Dragon 32 dispõe de um dialeto próprio da linguagem BASIC, conhecido como Microsoft Extended Colour Basic — Basic Ampliado para Cores da Microsoft. Vários outros computadores também seguem essa versão do BASIC, especialmente os da linha Tandy (TRS) que operam em cores.

O BASIC da Microsoft é de uso fácil e possui uma boa variedade de comandos para desenhar linhas, círculos e outras figuras geométricas. Depois de desenhadas, as figuras podem ser coloridas, proporcionando imagens interessantes na tela e pouco empenho na programação.

O Dragon 32 tem sete níveis de resolução, o que dá ao usuário a possibilidade de trabalhar com a tela dividida em 512 pontos individuais, no nível mínimo, e até 49.152 pontos, no nível máximo. Há oito cores disponíveis, mas a escolha pode ser limitada a quatro ou mesmo a duas cores em operações com alta resolução.

## Modos de resolução

A tela comum de dezesseis linhas por 32 colunas de caracteres forma o nível mais baixo de resolução e o comando PRINT@ permite a colocação do caractere em qualquer uma das 512 posições na tela. Além do conjunto normal de caracteres, há dezesseis caracteres gráficos de baixa resolução, disponíveis em oito cores.

O outro modo de resolução divide a tela em 32 linhas e 64 colunas. Assim, o tamanho de cada quadrado corresponde a um quarto do tamanho do caractere normal. Pontos desse tamanho podem ser traçados na tela pelo comando SET e eliminados pelo comando RESET.

Ambos os métodos podem ser utilizados ao mesmo tempo, quando são denominados telas de texto de baixa resolução. Há também cinco níveis de telas de alta resolução, mas não podem ser apresentados simultaneamente ou com telas de baixa resolução. Os cinco modos de alta resolução permitem escolhas de acordo com o padrão de resolução e o número disponível de cores, e são selecionados pelo comando PMODE.

| PMODE | Resolução | Cores disponíveis |
|---|---|---|
| 0 | 128*96 | 2 |
| 1 | 128*96 | 4 |
| 2 | 128*192 | 2 |
| 3 | 128*192 | 4 |
| 4 | 256*192 | 2 |

Naturalmente, há intercâmbio entre a resolução, a cor e a capacidade de memória necessária para armazenar dados da tela e isto deve ser levado em conta quando forem escritos programas extensos em BASIC que também empreguem imagens de alta resolução.

Embora permitindo apenas um número limitado de cores em alta resolução, o Dragon tem o recurso de, entre dois conjuntos de cores, selecionar um. Isso é feito pelo comando SCREEN. Por exemplo, a instrução SCREEN 1,0 escolhe uma tela de alta resolução e o conjunto de cores 0. A instrução SCREEN 1,1, por sua vez, escolhe uma tela de alta resolução, mas com um conjunto alternativo de cores.

**PAINT**
Este comando é muito útil como auxiliar na produção de figuras interessantes. Com o emprego do PAINT, o computador começa a colorir a tela a partir de determinado ponto, até uma linha limite. Isso significa que círculos, triângulos e outras figuras fechadas podem simplesmente ser preenchidas.

**DRAW**
O comando DRAW imita o movimento de um lápis na tela, permitindo ao usuário desenhar linhas em quatro direções. O comando DRAW também possibilita a rotação ou a ampliação da figura terminada pelo usuário.

**GET e PUT**
O comando GET instrui o computador a armazenar uma imagem da tela na memória e o PUT faz com que essa imagem seja reapresentada.

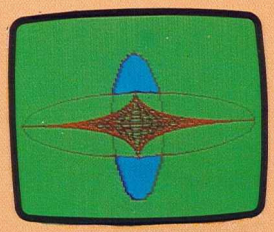
**PSET e PRESET**
Estes são os equivalentes em alta resolução dos comandos SET e RESET, que anteriormente examinamos. Eles permitem ligar e desligar pontos na tela. A cor do ponto também pode ser facilmente determinada.

**LINE**
Este comando une dois pontos específicos através de uma linha reta em alta resolução.

**CIRCLE**
Permite ao usuário desenhar círculos de alta resolução a partir de centro e raio determinados. As frações do círculo inteiro também são desenhadas de modo a formar arcos e a forma circular pode produzir elipses.

O micro Dragon 32 encontrado na Inglaterra é um computador de preço relativamente alto, que possui muitos comandos aperfeiçoados para auxílio da programação de gráficos. É mais adequado para apresentação de imagens estáticas do que para animação rápida. É apropriado para crianças que apreciam brincadeiras de aventuras.

O principal inconveniente do Dragon está em não apresentar na tela simultaneamente texto e gráficos de alta resolução. Isso significa que não pode ser usado para apresentar dados estatísticos na forma de gráficos em barras ou círculos.



**Comando de cores**
Imagem típica dos efeitos que podem ser obtidos no Dragon com apenas alguns de seus comandos de alto nível.

**Alta resolução**
Este pequeno programa para o Dragon 32 ilustra algumas de suas capacidades de alta resolução. O programa usa a instrução PMODE 3; este não é o nível mais alto, mas permite algum uso de cores.

```
10 PCLS:PMODE 3,1
20 SCREEN 1,0
30 COLOR 0,1
40 FOR X=0 TO 127 STEP 10
50 LINE(X,85)-(127,85-X/3),
   PSET
60 LINE(X,85)-(127,85+X/3),
   PSET
70 LINE(255-X,85)-(127,85-
   X/3), PSET
80 LINE(255-X,85)-(127,85+
   X/3), PSET
90 NEXT X
100 CIRCLE(127,85),128,4,0,3
110 CIRCLE(127,85),30,4,3
120 PAINT(130,30),3,4
130 PAINT(130,130),3,4
140 GOTO 140
150 END
```

# A ordem da jogada

## A capacidade de classificar dados é fundamental para a maioria dos programas e há muitos meios de consegui-lo.

**Bubble Sort (Classificação Bolha)**
O diagrama ilustra "a classificação bolha" para um leque limitado a nove cartas (D representa a carta 10). A parte ordenada começa da extremidade direita, a cada passagem. Os números 1 e 2 sob os conjuntos indicam as duas cartas que estão sendo comparadas.

```
Início da classificação
2 8 9 3 D 5 R 6 7 Início da passagem 1
1 2
8 2 9 3 D 5 R 6 7
  1 2
8 9 2 3 D 5 R 6 7
    1 2
8 9 3 2 D 5 R 6 7
      1 2
8 9 3 D 2 5 R 6 7
        1 2
8 9 3 D 5 2 R 6 7
          1 2
8 9 3 D 5 R 2 6 7
            1 2
8 9 3 D 5 R 6 2 7
              1 2
8 9 3 D 5 R 6 7 2 Fim pass. 1
9 8 D 5 R 6 7 3 2 Fim pass. 2
9 D 8 R 6 7 5 3 2 Fim pass. 3
D 9 R 8 7 6 5 3 2 Fim pass. 4
D R 9 8 7 6 5 3 2 Fim pass. 5
R D 9 8 7 6 5 3 2 Fim pass. 6
Encerramento da classificação
```

**Classificação por inserção**
Por este método, a parte ordenada avança a partir da extremidade esquerda. As cartas são deslocadas diretamente para sua posição correta na lista à medida que são examinadas.

```
Início da classificação
2 8 9 3 D 5 R 6 7
2 1
8 2 9 3 D 5 R 6 7
2   1
9 8 2 3 D 5 R 6 7
    2 1
9 8 3 2 D 5 R 6 7
2       1
D 9 8 3 2 5 R 6 7
      2   1
D 9 8 5 3 2 R 6 7
2           1
R D 9 8 5 3 2 6 7
          2     1
R D 9 8 6 5 3 2 7
          2         1
R D 9 8 7 6 5 3 2
Encerramento da classificação
```

Embora uma das operações mais corriqueiras do computador, a classificação (sort) é uma tarefa em que ele se mostra — segundo seus próprios padrões — de pouquíssima eficiência. De acordo com pesquisas operacionais, cerca de 30 a 40% de todo o trabalho de computação é empregado em classificação e, se acrescentarmos as tarefas a ela vinculadas, de juntar os dados e buscar itens específicos, esse índice provavelmente ultrapassará os 50%.

Métodos aperfeiçoados de classificação são de entendimento extremamente difícil, mas torna-se bastante fácil compreender os mais simples, quando recorremos ao exemplo da ordenação de um baralho.

Coloque treze cartas do mesmo naipe em uma mesa, dispondo-as em uma linha, sem seguir uma ordem específica, de modo que o ás e o 2 não fiquem na extremidade direita da linha. As cartas devem ser dispostas em ordem decrescente (rei, rainha, valete... ás), a partir da esquerda. Esta é, para nós, uma tarefa banal. Entretanto, se for exigido que apenas uma carta seja deslocada por vez, que nenhuma carta seja colocada sobre outra e que todas ocupem o mínimo espaço possível, não será fácil determinar um método eficiente para sua execução. Nesta analogia, as cartas equivalem às unidades de dados, a superfície máxima ocupada corresponde à memória necessária do computador, e você representa o programa. Como resolver o problema?

1) Coloque uma moeda debaixo da carta situada na extremidade esquerda, para representar um marcador de posição e fazê-lo lembrar-se do ponto em que você se encontra no processo de ordenação ou classificação. Compare a carta marcada com a carta da direita. Estão em ordem decrescente? Se não estiverem, inverta a posição, deixando a moeda no mesmo lugar e seguindo a regra de mover apenas uma carta por vez e não colocar uma carta sobre outra. Observe o que você deve fazer para inverter a ordem.

2) Se as duas cartas estiverem na ordem correta, desloque a moeda um espaço para a direita e repita o procedimento 1. Você agora se encontra em um loop que se encerrará quando a moeda alcançar a extremidade direita da linha. O alcance dessa posição denomina-se fazer uma "passagem" pelas cartas.

3) No final da primeira passagem, examine as cartas. O ás, que é a carta mais baixa do naipe, atravessou a série até atingir a extremidade direita e está portanto na posição correta. Se você fizer outras passagens pela série de cartas, como foi dito nos procedimentos 1 e 2, a carta 2 terá alcançado sua posição correta. Esse procedimento repetido em sucessivas passagens resultará, por fim, na colocação de todo o naipe em ordem decrescente.

Você deve ter percebido vários inconvenientes nesse método. É muito enfadonho e toma tempo, pois a simples troca de posições entre duas cartas exige três operações; e, acima de tudo, muitas das comparações realizadas entre várias cartas são desnecessárias. Por exemplo, após uma passagem, o ás se encontra em seu lugar correto; assim, não adianta deslocar a moeda para a posição 13 (na qual, de qualquer forma, nenhuma comparação é possível). Na segunda passagem, visto que a carta da direita está no lugar correto, não há necessidade de deslocar o indicador para a posição 12. De modo geral, cada passagem terminará em uma posição à esquerda do ponto de encerramento da passagem anterior.

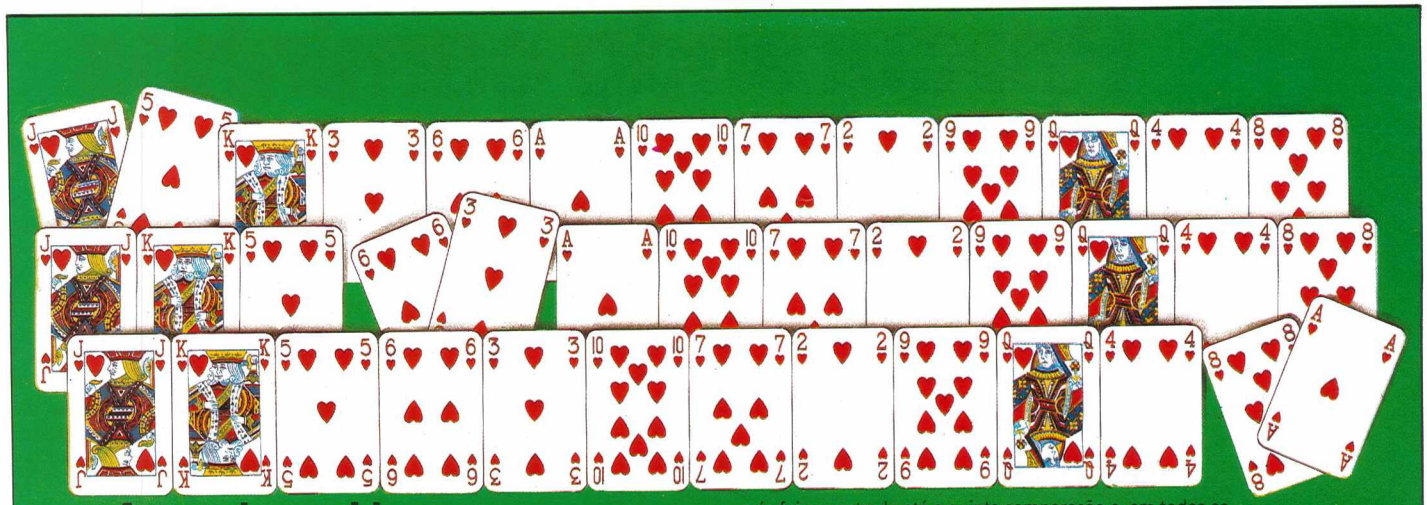
Determinar o ponto onde o procedimento deve ser interrompido é outro problema. Um computador continuará indefinidamente a comparar as cartas, a menos que receba instruções para parar. A única regra segura é a seguinte: pare após uma passagem sem inversões. Em outras palavras, se você fez a passagem pelos dados sem alteração na ordem, segue-se que eles estão na posição correta.

O método de ordenação que examinamos é chamado "Bubble Sort" ("Classificação Bolha"). Entre suas vantagens, estão as técnicas simples de programação, o uso limitado de memória auxiliar e razoável eficácia, com pequena quantidade de dados parcialmente ordenados. E é por esses critérios que os algoritmos de classificação devem ser avaliados. Não obstante, quando os dados a classificar são extensos, a velocidade possivelmente será comprometida em favor da economia de memória, porque a memória do computador talvez não comporte, ao mesmo tempo, os dados não processados e a cópia classificada. Por esse motivo, vamos ignorar os algoritmos que exigem a extração de dados de uma matriz e seu deslocamento para a posição classificada em uma segunda matriz. O segundo método de classificação simples é baseado mais diretamente no modo pelo qual colocamos cartas em uma determinada ordem.

1) Estenda outra vez na mesa as cartas embaralhadas e coloque uma moeda de 10 cruzeiros debaixo da segunda carta, a partir da esquerda. Todas as cartas sob as quais a moeda estiver, no início de cada passagem, serão chamadas "carta 10 cruzeiros"

2) Empurre a carta 10 cruzeiros para fora da linha, deixando uma lacuna, e coloque uma moeda de 50 cruzeiros debaixo da carta imediatamente à esquerda. Esta será chamada carta 50 cruzeiros.

3) Compare as duas cartas. Se estiverem em ordem, recoloque a carta 10 cruzeiros no lugar e passe para o procedimento 4. Caso contrário, coloque a carta 50 cruzeiros na lacuna e desloque a moeda de 50 cruzeiros para a posição à esquerda, a fim de marcar uma nova carta 50 cruzeiros (se esta estiver à extrema esquerda, este procedimento não será pertinente; neste caso, coloque a carta 10 cruzeiros na lacuna e siga para o procedimento 4).



## A grande partida

Um modo de ilustrar a "Classificação Bolha" consiste em ordenar um naipe completo de cartas, de modo que fique o rei na extremidade esquerda e o ás, na direita. Primeiro, comparam-se as duas cartas da extremidade esquerda que, estando fora de ordem, têm sua posição invertida. A seguir, a segunda e a terceira cartas são comparadas e suas posições também são trocadas. Por este procedimento de classificação, o ás foi encontrado até a quinta comparação e, em todas as comparações subseqüentes, ele teve sua posição mudada da esquerda para a direita até que, no final da primeira "passagem", terminou de percorrer a linha de cartas, chegando à extremidade direita. Com a repetição completa desse procedimento, na segunda passagem, a carta 2 chega a sua posição, ao lado do ás. Todavia, podem ser necessárias até doze passagens desse tipo para que todas as cartas fiquem em ordem decrescente conforme a proposição inicial.

Compare esta carta 50 cruzeiros com a carta 10 cruzeiros (a carta deslocada). Agora, repita o procedimento 3 até que a posição correta da carta 10 cruzeiros seja encontrada.

4) Desloque a moeda de 10 cruzeiros para a posição à direita e repita os procedimentos 2 e 3. Quando se tornar impossível deslocar a moeda de 10 cruzeiros para a direita, todas as cartas estarão na ordem correta.

Este procedimento é chamado "Classificação por Inserção" e é muito semelhante ao modo de as pessoas ordenarem um leque de cartas. Embora de programação um pouco mais difícil que a "Classificação Bolha", este método é muito mais eficiente. Depois, neste curso, veremos alguns algoritmos mais complexos para classificação de dados.

```
9  REM************************************
10 REM* ALGORITMOS DE
   CLASSIFICACAO*
11 REM************************************
100 INPUT "FORNECER O NUMERO DE
    ITENS A SER
    CLASSIFICADOS " ; LT
150 IF LT<3 THEN LET LT=3
200 LET LT=INT(LT)
250 DIM R(LT), C(LT)
300 LET Z=0:LET Q=0:LET P=0
350 LET I=1:LET O=0:LET II=2:LET TH=2
400 INPUT "DETERMINAR QUANTOS
    TESTES "; N
450 FOR CT=I TO N
500 GOSUB 4000
550 FOR SR=I TO TH
600 GOSUB 5000
650 PRINT:PRINT:PRINT:PRINT
700 PRINT "TESTE NO."; CT+SR/10
750 INPUT "PRESSIONAR TECLA DE
    RETORNO PARA INICIAR A
    CLASSIFICACAO " ; A$
800 PRINT "A LISTA NAO CLASSIFICADA E"
850 GOSUB 3000
900 ON SR GOSUB 6000, 7000
950 PRINT "A LISTA CLASSIFICADA E"
1000 GOSUB 3000
1050 NEXT SR
1100 NEXT CT
1150 END
2999 REM************************************
3000 REM* IMPRIMIR A LISTA *
3001 REM************************************
3100 FOR K=I TO LT
3200 PRINT R(K) ;
3300 NEXT K
3400 PRINT
3500 RETURN
3999 REM************************************
4000 REM* GERADOR RANDOMICO *
4001 REM************************************
4100 RANDOMIZE
4200 FOR K=I TO LT
4300 LET C(K)=INT(100*RND)
4400 NEXT K
4500 RETURN
4999 REM************************************
5000 REM* GERADOR RANDOMICO *
5001 REM************************************
5100 FOR K=I TO LT
5200 LET R(K)=C(K)
5300 NEXT K
5400 PRINT:PRINT
5500 RETURN
5999 REM************************************
6000 REM* BOLHA *
6001 REM************************************
6050 PRINT "CLASSIFICACAO BOLHA -
     INICIAR ! ! ! ! !
6100 FOR P=LT-I TO I STEP-I
6150 LET F=I
6200 FOR Q=I TO P
6250 LET Z=Q+I
6300 IF R(Q)<R(Z) THEN LET D=R(Q):
     LET R(Q)=R(Z):LET R(Z)=D:LET F=O
6350 NEXT Q
6400 IF F=I THEN LET P=I
6450 NEXT P
6500 PRINT "CLASSIFICACAO BOLHA -
     ENCERRAR ! ! ! ! !
6550 RETURN
6999 REM************************************
7000 REM* INSERCAO *
7001 REM************************************
7050 PRINT "CLASSIFICACAO POR
     INSERCAO - INICIAR ! ! ! ! !
7100 FOR P=II TO LT
7200 LET D=R(P)
7300 FOR Q=P TO II STEP-I
7400 LET R(Q)=R(Q-I)
7500 IF D<R(Q) THEN LET R(Q)=D:LET Q=II
7600 NEXT Q
7700 IF D>R(I) THEN LET R(I)=D
7800 NEXT P
7850 PRINT "CLASSIFICACAO POR
     INSERCAO - ENCERRAR ! ! ! ! !
7900 RETURN
```

**Classificação em alta velocidade**

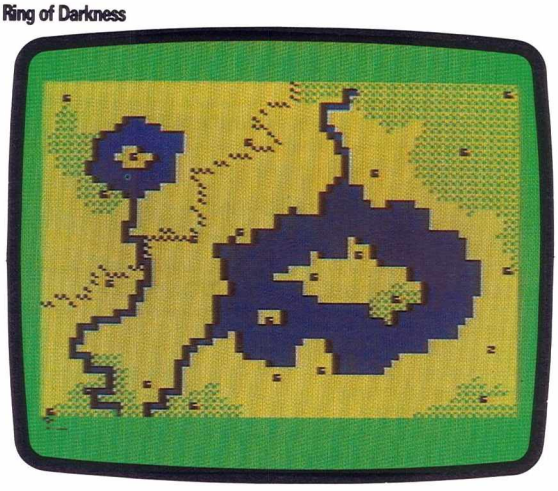
Este programa em BASIC mostra a diferença de eficácia entre a "Classificação Bolha" e o método por inserção. Como o código foi escrito tendo em vista a velocidade, não documentamos a operação das rotinas. A listagem pode ser processada na maioria dos equipamentos, mas veja na página 215 as adaptações no quadro "A propósito...", para a instrução ON...GOSUB, e, na página 175, para as funções RND e RANDOMIZE.

# Procurando caminhos

**Labirintos sempre exerceram fascínio sobre as pessoas — e os jogos de labirinto no microcomputador não fogem à regra.**

Os labirintos, quer sejam grandes a ponto de as pessoas se perderem ou tão pequenos que caibam na palma da mão, sempre constituíram uma fonte de fascinação e divertimento tanto para jovens quanto para adultos. De fato, eles se tornaram a base de uma grande variedade de jogos de computador, que podem apresentar desde um labirinto visto de cima, bidimensional e muito simples, até outros extremamente complexos, em três dimensões. Estes últimos levam o jogador a imaginar que se encontra no interior de um labirinto de verdade. A fim de ajudá-lo a se orientar, ou então confundi-lo mais ainda, alguns desses jogos em três dimensões também mostram, de vez em quando e por alguns instantes, como eles seriam vistos de cima.

**Ring of Darkness**



Como os efeitos sonoros e visuais dos labirintos se tornaram mais aperfeiçoados, a criatividade de seus programadores pôde desenvolver-se.

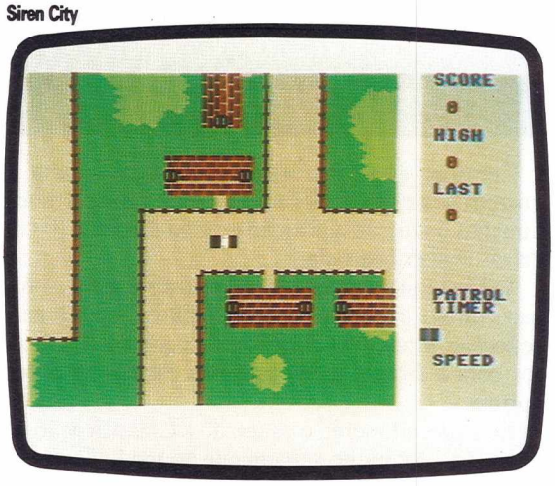
O jogador que quiser dar um passeio pelo labirinto deve evitar aqueles caminhos que escondem monstros devoradores. Um exemplo destes jogos é o 3D Glooper (próprio para o Commodore 64), onde o jogador caminha procurando ladrilhos especiais no chão e pode ser atacado a qualquer momento por monstros que ocupam a tela. No entanto, a chegada iminente dessas criaturas é anunciada pelo ruído de suas mandíbulas.

O Atic Atac (Sinclair-Spectrum) é uma perseguição totalmente animada, em que o jogador pode assumir a identidade de três personagens diferentes. O labirinto consiste em uma série de fossos, escadarias e grandes calabouços, em vários níveis, por entre os quais deve correr contra o tempo. Os calabouços são ocupados por uma variedade de estranhas criaturas e objetos.

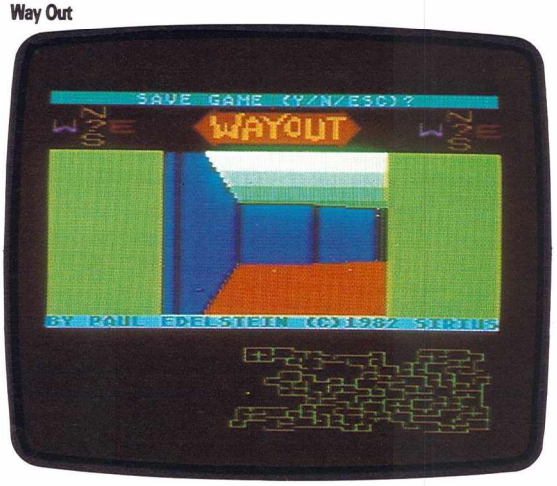
O programa que mais se aproxima da simulação do que realmente se sente ao tentar atravessar um labirinto é o Way Out. As coisas são vistas em verdadeira perspectiva tridimensional e, à medida que você move o joystick aos poucos para a esquerda e para a direita, o cenário vislumbrado muda proporcionalmente.

Vejamos algumas das técnicas básicas de programação usadas na construção de labirintos.

**Siren City**




**Way Out**




## Construção de labirintos

A maneira geral de armazenar dados sobre um labirinto é usar uma matriz bidimensional. Cada elemento da matriz define as características daquele elemento do labirinto. Você poderia, por exemplo, usar uma seqüência de quatro caracteres para representar norte, sul, leste e oeste. Zero poderia indicar ausência de muro, e 1, a presença de um muro. Assim sendo, se M$(5,6) contém a seqüência "1011", isso indica que o elemento na linha cinco, coluna seis, está cercado por muros ao sul e leste.

**Siren City**
Este jogo do Commodore 64 é uma versão desenvolvida do tradicional jogo de "vista aérea". Um carro de polícia patrulha uma cidade.

**Ring of Darkness**
Embora este jogo para o micro inglês Dragon seja do tipo Aventura, apresenta um labirinto em três dimensões como um de seus principais elementos. Você anda para cima e para baixo, em fossos e escadas.

**Way Out**
Uma imagem realista em três dimensões é conseguida no Sinclair-Spectrum com o Way Out. Movimentando gradualmente o joystick, mudará também o cenário.

Para poupar espaço na memória, a matriz pode ser numérica e não alfanumérica e o número de quatro dígitos considerado como um binário. No nosso exemplo, o elemento que contém muros ao norte, a leste e a sul seria representado pelo número 11 (1011).

Todos os elementos começam com quatro muros. Gerando de modo aleatório a entrada por qualquer ponto do perímetro, o próximo elemento será escolhido ao acaso entre qualquer um dos três adjacentes. Quando o elemento é escolhido, a seqüência continua — selecionando ao acaso um elemento de qualquer um dos três adjacentes, desprezando aquele de onde você veio.

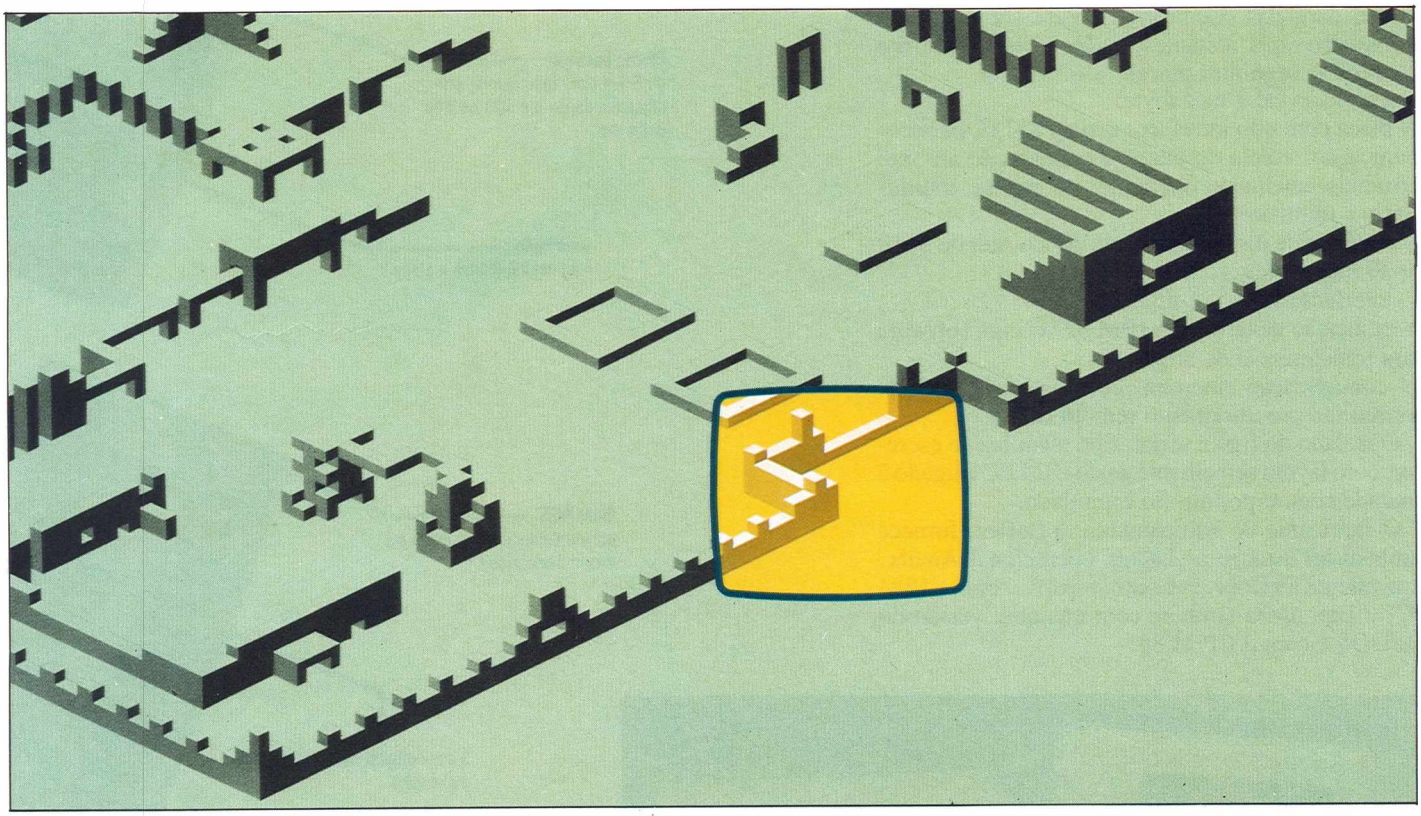
Toda vez que você escolhe uma nova direção, o muro existente entre o elemento em que você está e o elemento em que vai entrar é removido. É preciso prestar atenção para não sair dos limites do labirinto (a não ser que aquele determinado elemento do perímetro seja o ponto de saída), e não criar circuitos fechados (todas as partes de um labirinto devem ter acesso de qualquer outra parte).

Quando um elemento que tem menos de quatro muros (isto é, um elemento que já foi visitado) é encontrado, o programa deve escolher um dos elementos adjacentes que restarem. Se todos os elementos adjacentes já tiverem sido visitados, o programa deve "dar um passo atrás", voltando ao último elemento visitado, e tomar um novo caminho.

Outro método de gravar as características de um elemento consiste no uso mais sofisticado da numeração binária, que é particularmente útil para exibir perspectivas em três dimensões. Há dezesseis maneiras possíveis de se construir um elemento: sem muros, com muros de todos os lados, com um único muro (quatro possibilidades), muros em lados opostos (duas possibilidades), dois muros adjacentes (quatro possibilidades) e três muros adjacentes (quatro possibilidades).

Usando o número binário apropriado (0-15) para cada uma dessas possibilidades, pode-se "girar" o número para a esquerda ou para a direita até chegar à visão que o jogador realmente teria. Por exemplo, um muro face norte pode ser representado por 2 (0010), face sul, por 8 (1000), face leste, por 1 (0001) e face oeste, por 4 (0100). Se o jogador, estando voltado para o norte, num compartimento que tem apenas um muro face oeste (4), se voltar para oeste, sua visão do compartimento será como se ele estivesse diante de um muro face norte. Quando o jogador vira para a sua esquerda (oeste), o padrão do bit muda uma casa à direita, preenchendo a descrição que queríamos, isto é, o binário do muro face oeste 0100 (decimal 4) transforma-se no binário 0010 (decimal 2, um muro face norte!). Quando viramos à direita, os bits viram na direção oposta, e se virarmos duas vezes eles darão meia-volta. É necessário, sem dúvida, incluir um sistema para "embrulhar" os bits que caem das duas pontas do meio-byte durante este processo; caso contrário, as características que identificam um elemento serão mudadas toda vez que o jogador se virar para algum lado. Um elemento originalmente definido como 0011, por exemplo (muros face norte e leste), deve transformar-se em 0110, se o jogador virar à direita, e 1100 se ele der meia-volta.

Em linguagem de máquina há instruções especiais para girar binários para a esquerda e direita. Em BASIC, um binário de 4 bits expresso como um número decimal entre 0 e 15 pode ser virado à esquerda multiplicando-se o número por 2 e então subtraindo-se de 15, se o resultado passar de 15. Para girá-lo à direita, se o número for par, basta dividi-lo por 2 e, se for ímpar, soma-se 15 e divide-se por 2.



**Ant Attack**
Quando este jogo é executado pelo Spectrum, a tela do computador atua como uma janela no amplo campo de jogo em forma de labirinto. À medida que o jogo transcorre, a tela se movimenta mostrando outros aspectos do cenário.

# EGO

## O micro de 16 bits, compatível com o IBM PC, e ainda multiusuário e multitarefa.

O Ego é o primeiro microcomputador nacional de 16 bits compatível com o IBM PC. Destina-se a uso profissional e empresarial, com aplicações nas áreas comercial, científica e educacional.

Composto de três módulos básicos (unidade central, teclado e monitor), dispõe de cinco *slots* para expansão do sistema. Desses, apenas um é ocupado pelo controlador de disco flexível, ficando os outros quatro, portanto, à disposição do usuário.

A placa do controlador de disco tem funções múltiplas, permitindo a sobreposição de até três outras miniplacas — interfaces seriais (RS232 C assíncrona) ou paralelas (Centronics). Para ser acopladas nos quatro *slots* restantes, estão à disposição dos usuários as seguintes placas:

— expansão até 1 megabyte;
— placa com oito interfaces seriais 232 C, para comunicação, coleta de dados e conexão de até sete terminais adicionais operando em regime multiusuário e multitarefa;
— conexão de disco rígido tipo Winchester de 5, 10 ou 20 megabytes;
— interface analógico-digital;
— emulação de terminais IBM 327x, com software para transferência de arquivos;
— comunicação síncrona, com protocolo BSC1, permitindo processamento remoto (RJE).

O teclado do Ego é serial, tipo máquina de escrever, com layout semelhante ao do IBM PC e ligado à unidade central por um fio espiralado.

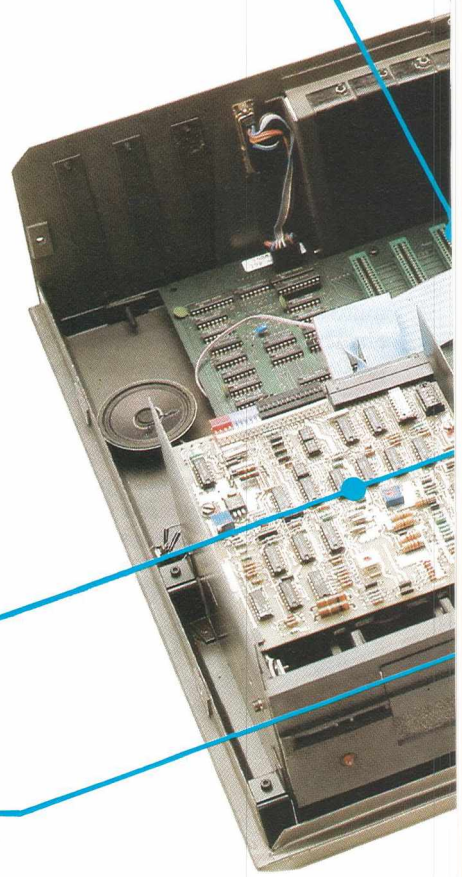
O fabricante do equipamento, a Softec, fornece também ao usuário o sistema operacional Analix, equivalente ao Unix. Por ser compatível com o IBM PC, o Ego opera também com qualquer versão do MSDOS e com o CP/M 86.



4 conectores livres para expansão, que podem receber placas:
— de expansão de memória;
— para emulação de terminais 3278/9;
— para comunicação síncrona;
— para controle de discos rígidos do tipo Winchester ou CMD;
— para controle de até 8 terminais (multiusuário)

**Discos flexíveis**
de 5 1/4 pol., face dupla, com capacidade de até 400 Kbytes cada um

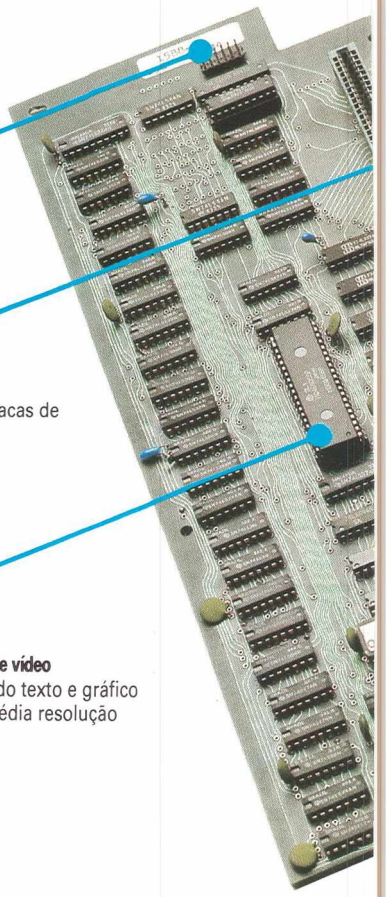
**Ventilação**
forçada na frente e atrás

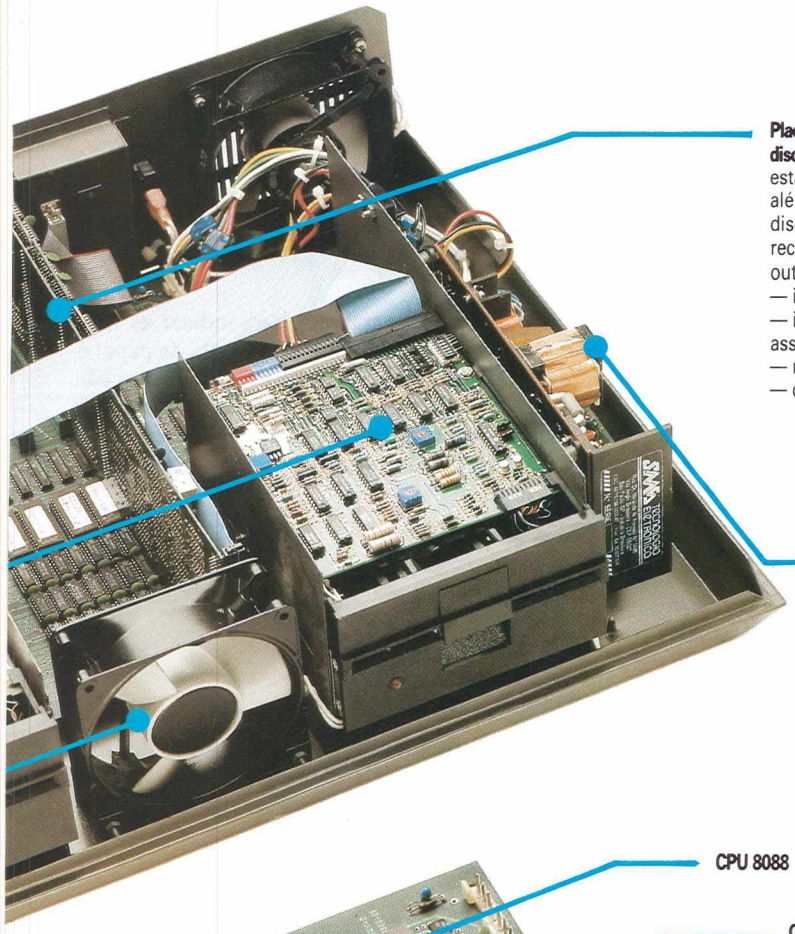


**Saída RGB**
para monitores coloridos ou monocromáticos

5 conectores para placas de expansão

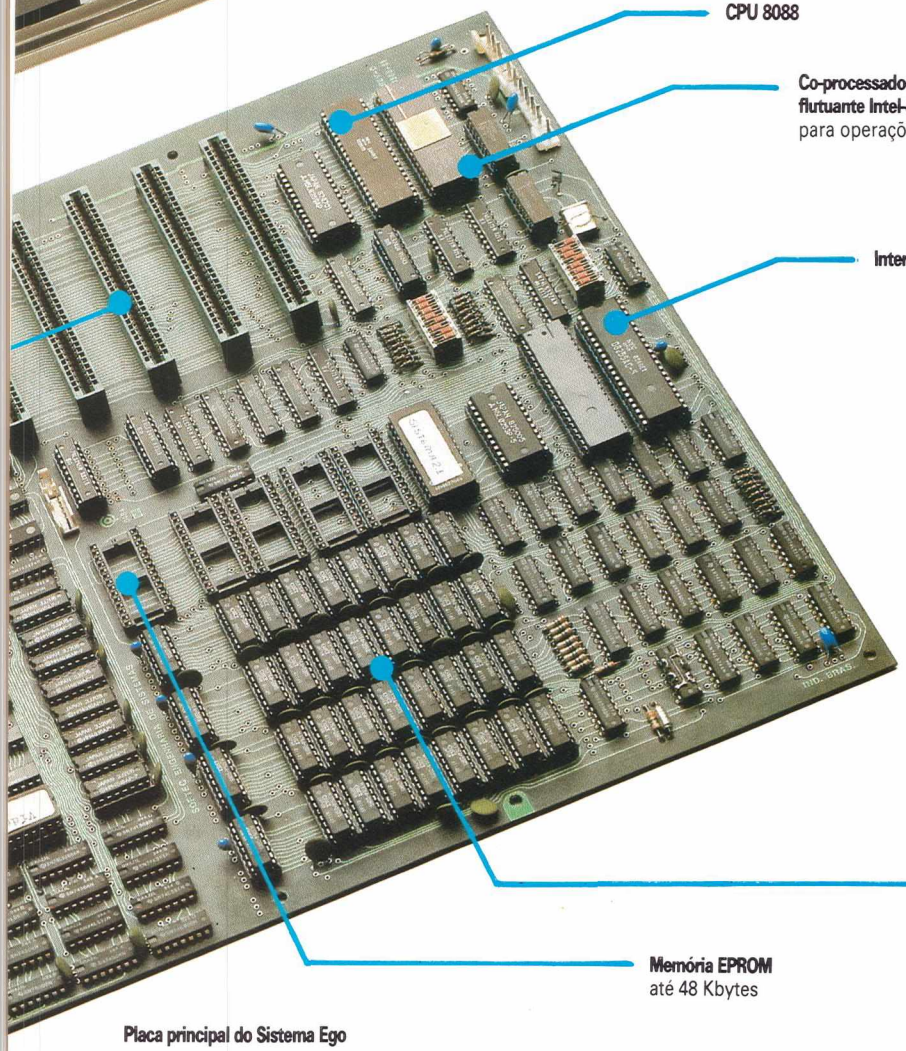
**Processador de vídeo**
controla modo texto e gráfico de alta ou média resolução





**Placa controladora de unidades de disquete**
esta placa foi projetada para, além de controlar até 4 disquetes de 5 1/4 ou 8 pol., receber em espaço reservado 3 outras placas, que podem ser:
— interfaces paralelas;
— interfaces seriais assíncronas;
— relógio não-volátil;
— controlador de Winchester

**Fonte de alimentação chaveada**
com opção para fonte de 100 W ou 250 W, dependendo da configuração do sistema



**CPU 8088**

**Co-processador de ponto flutuante Intel-8087**
para operações científicas

**Interface de teclado**

**Memória RAM**
capacidade de 256 Kbytes na placa com verificação de paridade. Expansão até 1 Mbyte em placa adicional

**Memória EPROM**
até 48 Kbytes

**Placa principal do Sistema Ego**

**MICROPROCESSADOR**

8088 — Processador central
8087 — Co-processador aritmético de ponto flutuante.

**CLOCK**

5 MHz

**MEMÓRIA**

RAM inicial de 128 K, expansível até 1 megabyte.
ROM até 48 K.

**VÍDEO**

Monitor de 14 pol, em cores ou monocromático de alta persistência, em verde, azul ou âmbar. Em modo gráfico, resolução de 200 × 640 pontos e em alfanumérico de 25 × 40 ou 25 × 80 caracteres. Interface RGB e intensidade para formar 16 cores.

**TECLADO**

Tipo máquina de escrever com 88 teclas (13 de funções, sendo 3 com seqüências especiais), teclado numérico separado, repetição automática.

**LINGUAGENS**

BASIC compilado, Compilador C e FORTRAN.

**PERIFÉRICOS**

Unidades de disco 5 1/4 pol., 8 pol., e Winchester de 5 a 20 megabytes; impressoras; interface serial RS232 C; placa e emulação de terminais IBM 3278179; placa de comunicação síncrona (emulação de 3780-RJE).

**DOCUMENTAÇÃO**

Três manuais: um do sistema operacional ANALIX, outro dos programas utilitários e aplicativos e outro das linguagens.

# Elaboração do programa

**À medida que um programa extenso é desenvolvido, sua estrutura adquire a aparência de uma árvore, com maior número de ramificações em cada estágio de refinamento.**

No último fascículo, em nosso curso de programação em BASIC, analisamos alguns dos problemas envolvidos no exame de uma lista para encontrar um determinado item, partindo da premissa de que a lista já estava organizada segundo uma determinada ordem. Trata-se de uma questão que reexaminaremos com maiores detalhes no momento em que iniciarmos o desenvolvimento das rotinas de busca. Nesse meio tempo, porém, vamos desenvolver o tema da programação top-down: a produção de código para as duas segundas partes do programa principal. Esse procedimento inclui quatro chamadas para sub-rotinas ou procedimentos:

```
PROGRAMA PRINCIPAL
INICIAR
  INICIALIZAR (procedimento)
  SAUDAÇÃO (procedimento)
  ESCOLHA (procedimento)
  EXECUÇÃO (procedimento)
ENCERRAMENTO (END)
```

O primeiro procedimento, *INICIALIZAR*, exige inúmeras atividades de razoável complexidade — constituição de matrizes, incorporação de dados a elas, execução de várias verificações, e assim por diante —, de modo que deixaremos os detalhes dessas várias etapas para exame posterior. As duas partes subseqüentes do programa principal abrangem os procedimentos de SAUDAÇÃO e ESCOLHA. Ao desenvolver esses procedimentos, sugerimos uma metodologia para ajudar a evitar que se tornem desorganizadas e confusas as muitas camadas incluídas na programação top-down.

A dificuldade no desenvolvimento do programa com o refinamento top-down está na imprevisibilidade do número de estágios necessários até atingirmos o ponto em que é possível a codificação em linguagem de alto nível. Para procedimentos simples, talvez dois ou três estágios sejam suficientes, mas procedimentos mais complexos poderão exigir muitos estágios, antes de o problema ser suficientemente analisado para permitir a conversão em "código fonte" (conforme é chamado o programa em linguagem de alto nível). Isso significa que escrever um programa por esse método assemelha-se a desenhar uma árvore tombada. Conforme as ramificações aumentam (isto é, à medida que os refinamentos se tornam mais detalhados), elas passam a ocupar mais espaço na página. Por fim, torna-se impossível colocar tudo em uma única página e é este o ponto em que se corre o risco de perder a noção do que está acontecendo.

Um modo muito eficiente de organizar a documentação do programa consiste em numerar sistematicamente os estágios de seu desenvolvimento. Usamos números romanos para indicar o nível de refinamento, e números arábicos para indicar as subseções do programa. Uma folha separada de papel é empregada para cada nível de refinamento, e as páginas para cada bloco de programa ou módulo podem facilmente ser mantidas juntas. Apresentamos abaixo o sistema de numeração para nosso programa:

```
I PROGRAMA PRINCIPAL
INICIAR
  1. INICIALIZAR
  2. SAUDAÇÃO
  3. ESCOLHA
  4. EXECUÇÃO
ENCERRAR (END)
```

Conforme dissemos antes, por ora deixamos de lado o procedimento INICIALIZAR, concentrando-nos no desenvolvimento dos procedimentos SAUDAÇÃO e ESCOLHA.

```
II 2 (SAUDAÇÃO)
INICIAR
  1. Apresentar mensagem de saudação
  2. LOOP (até ser pressionada a barra de espaço)
     ENCERRAR O LOOP (ENDLOOP)
  3. Chamar rotina *ESCOLHA*
ENCERRAR (END)
```

```
III 2 (SAUDAÇÃO) 1 (apresentar mensagem)
INICIAR
  1. Limpar a tela
  2. IMPRIMIR (PRINT) mensagem de saudação
ENCERRAR (END)
```

```
III 2 (SAUDAÇÃO) 2 (LOOP para esperar a barra de espaço)
INICIAR
  1. LOOP (até ser pressionada a barra de espaço)
        SE (IF) a barra de espaço for pressionada
        ENTÃO (THEN)
     ENCERRAR O LOOP (ENDLOOP)
ENCERRAR (END)
```

```
III 2 (SAUDAÇÃO) 3 (chamar *ESCOLHA*)
INICIAR
  1. GOSUB *ESCOLHA*
ENCERRAR (END)
```

Até aqui deve estar claro que os níveis **III-2-1** e **III-2-3** estão prontos para ser codificados diretamente em BASIC, mas o nível **III-2-2** necessita de outro estágio de refinamento:

```
IV 2 (SAUDAÇÃO) 2 (LOOP)
INICIAR
  1. LOOP (até ser pressionada a barra de espaço)
        SE (IF) INKEY$ não for pressionada, ENTÃO
        (THEN) continue
     ENCERRAR O LOOP (ENDLOOP)
ENCERRAR (END)
```

Agora atingimos o ponto em que toda a codificação em BASIC para o procedimento SAUDAÇÃO pode prosseguir com pouco refinamento adicional:

**IV 2 (SAUDAÇÃO) 1 (apresentar mensagem) CÓDIGO BASIC**

```
REM SUB-ROTINA *SAUDAÇÃO*
PRINT
PRINT
PRINT
PRINT
PRINT TAB(14); "*BENVINDO AO*"
PRINT TAB(4); "*MICROCOMPUTADOR: CURSO
      BÁSICO*"
PRINT TAB(1); "*AGENDA DE ENDEREÇOS
      COMPUTADORIZADA*"
PRINT
PRINT "(PRESSIONAR BARRA DE ESPAÇO PARA
      CONTINUAR)"
```

**V 2 (SAUDAÇÃO) 2 (LOOP para esperar a barra de espaço) CÓDIGO BASIC**

```
LET L = 0
FOR L = 1 TO 1
IF INKEY$ < > " " THEN LET L = 0
NEXT L
```

**IV 2 (SAUDAÇÃO) 3 (chamar *ESCOLHA*) CÓDIGO BASIC**

```
GOSUB *ESCOLHA*
RETURN
```

Observe que começamos agora a inicializar as variáveis nas diferentes rotinas que escrevemos, usando instruções da forma LET I = 0. A rigor, esta precaução é desnecessária em algumas das situações em que foi aqui empregada. Contudo, representa um bom hábito a ser adquirido, caso consiga lembrar-se, e se houver suficiente capacidade de RAM. Há três motivos para isso: primeiramente, porque colocar uma lista de instruções LET no início de qualquer rotina serve como um lembrete útil das variáveis específicas que a rotina está usando. Em segundo lugar, porque você não pode ter certeza do valor que permaneceu na variável desde a última vez em que foi utilizada em uma rotina (embora isso nem sempre tenha importância). Por fim, como explicaremos mais adiante no curso, a colocação de instruções do tipo LET I = 0 na ordem correta pode acelerar a execução do programa.

Modificamos o modo de usar o loop FOR-NEXT para simulação da estrutura DO-WHILE ou REPEAT-UNTIL, em relação aos fascículos anteriores do curso. Em vez de empregar os loops FOR I = 0 TO 1 ou FOR I = 0 to 1 STEP 0, agora utilizamos o loop FOR I = 1 to 1. Esta instrução será corretamente processada em todos os microcomputadores que costumamos apresentar, enquanto outros métodos exigem as adaptações para os diversos equipamentos, apresentadas no quadro "A propósito...". A instrução FOR I = 1 TO 1-NEXT I executará o loop uma única vez. No entanto, se em algum ponto do loop a variável I receber o valor 0, o loop será executado novamente, e assim por diante. Podemos inserir uma instrução LET I = 0 como resultado de uma condição de saída que não corresponda ao valor esperado, ou podemos atribuir o valor 0 à variável I imediatamente após a instrução FOR, atribuindo-lhe o valor 1, caso o contrário ocorra. Desse modo, ambos os loops abaixo alcançam o mesmo objetivo:

```
FOR I = 1 TO 1
IF INKEY$ < > " " THEN LET I = 0
NEXT I
```

ou

```
FOR I = 1 TO 1
LET I = 0
IF INKEY$ = " " THEN LET I = 1
NEXT I
```

Para o bloco SAUDAÇÃO completo no programa principal, necessitamos apenas do código em BASIC que acabamos de apresentar. Não colocamos os números de linha porque não podemos realmente fazê-lo antes de todos os módulos do programa ficarem prontos para a codificação final. Por exemplo, não se sabe neste estágio quais são os números de linha apropriados para os comandos GOSUB. Se você quiser testar o módulo neste estágio, será necessário criar entradas e sub-rotinas fictícias. Alguns pontos precisam ser observados quanto a este fragmento de programa: o emprego da função TAB e as instruções de "limpeza de tela". A função TAB move o cursor ao longo da linha, de acordo com o número (o "argumento") especificado entre parênteses. Com os números que fornecemos, a mensagem será impressa claramente, centralizada em uma tela de 40 caracteres de largura. Se a tela de seu equipamento tiver menor capacidade ou maior (computadores maiores têm capacidade para 80 caracteres), será necessário fazer as devidas alterações nesses argumentos TAB. A instrução para limpar a tela em muitas das versões do BASIC corresponde ao comando CLS, mas a versão BASIC da Microsoft usada para desenvolver este programa não admite seu emprego. Por isso, usamos a instrução PRINT CHR$(12), uma vez que nossa máquina usa o código ASCII 12 como caractere de não-impressão para "limpeza de tela". Outros equipamentos empregam o código ASCII 24 com a mesma função.

```
10 REM PROGRAMA PRINCIPAL FICTICIO
20 PRINT CHR$(12)
30 GOSUB 100
40 END
100 REM SUB-ROTINA *SAUDACAO*
110 PRINT
120 PRINT
130 PRINT
140 PRINT
150 PRINT TAB(14);"*BENVINDO AO*"
160 PRINT TAB(4);"*MICROCOMPUTADOR: CURSO
    BASICO*"
170 PRINT TAB(1);"*AGENDA DE ENDERECOS
    COMPUTADORIZADA*"
180 PRINT
190 PRINT "(PRESSIONAR BARRA DE ESPACO PARA
    CONTINUAR)"
195 LET L = 0
200 FOR L = 1 TO 1
210 IF INKEY$ < > " " THEN LET L = 0
220 NEXT L
230 PRINT CHR$(12)
240 GOSUB 1000
250 RETURN
1000 REM SUB-ROTINA FICTICIA
1010 PRINT "SUB-ROTINA FICTICIA"
1020 RETURN
```

Agora utilizaremos exatamente a mesma abordagem para elaborar o procedimento ESCOLHA.

**II 3 (ESCOLHA)**

```
INICIAR
  1. IMPRIMIR (PRINT) o menu
  2. FORNECER (INPUT) ESCOLHA
  3. Chamar a sub-rotina ESCOLHA
ENCERRAR (END)
```

**III 3 (ESCOLHA) 1 (IMPRIMIR (PRINT) o menu)**

```
INICIAR
  1. Limpar a tela
  2. IMPRIMIR (PRINT) o menu com opções
ENCERRAR (END)
```

**III 3 (ESCOLHA) 2 (FORNECER (INPUT) ESCOLHA)**

```
INICIAR
  1. FORNECER (INPUT) ESCOLHA
  2. Verificar se a ESCOLHA está dentro do limite
ENCERRAR (END)
```

**III 3 (ESCOLHA) 3 (chamar ESCOLHA)**

```
INICIAR
  1. OPÇÃO ESCOLHIDA
     ENCERRAR A OPÇÃO
ENCERRAR (END)
```

**III-3-1 (IMPRIMIR (PRINT) o menu)** pode agora ser codificado em BASIC:

**IV 3 (ESCOLHA) 1 (IMPRIMIR (PRINT) o menu) CÓDIGO BASIC**

```
REM LIMPAR A TELA
PRINT CHR$(12): REM OU 'CLS'
PRINT
PRINT
PRINT
PRINT
PRINT "1. ENCONTRAR UM REGISTRO
      (PELO NOME)"
PRINT "2. ENCONTRAR NOMES (POR TRECHO
      DE UM NOME)"
PRINT "3. ENCONTRAR REGISTROS (DE UMA
      CIDADE)"
PRINT "4. ENCONTRAR REGISTROS (DE UMA
      INICIAL)"
PRINT "5. LISTAR TODOS OS REGISTROS"
PRINT "6. ACRESCENTAR UM REGISTRO"
PRINT "7. MODIFICAR UM REGISTRO"
PRINT "8. ELIMINAR UM REGISTRO"
PRINT "9. SAIR E GRAVAR"
```

Todavia os níveis **III-3-2 (FORNECER ESCOLHA)** e **III-3-3 (chamar ESCOLHA)** exigem refinamento complementar. Vejamos primeiro o nível subseqüente de desenvolvimento, o **III-3-2**.

A atribuição de um valor numérico à variável ESCOLHA é extremamente simples: após a indicação, um comando FORNECER ESCOLHA realizará essa tarefa. Entretanto, há apenas nove escolhas possíveis. O que acontecerá se, por engano, fornecermos o valor 0 ou 99? Uma vez que a ESCOLHA feita vai determinar que parte do programa será chamada a seguir, é preciso garantir que não haja erros. Assim, devemos executar um procedimento de "verificação de limite", que consiste em uma pequena rotina que confere se o número fornecido está no limite aceitável, antes de dar prosseguimento ao programa. Eis, a título de exemplo, uma rotina para detectar entradas errôneas:

**ROTINA PARA VERIFICAR O LIMITE**

```
1 REM ROTINA
10 LET L = 0
20 FOR L = 1 TO 1
30 INPUT "ENTRADA 1-9";ESCOLHA
40 IF ESCOLHA < 1 THEN LET L = 0
50 IF ESCOLHA > 9 THEN LET L = 0
60 NEXT L
70 PRINT "A ESCOLHA FOI ";ESCOLHA
80 END
```

Muitas das versões do BASIC podem simplificar esta rotina com a inclusão de um operador booleano no teste, da seguinte forma:

```
10 LET L = 0
20 FOR L = 1 TO 1
30 INPUT "ENTRAR 1-9";ESCOLHA
40 IF ESCOLHA < 1 OR ESCOLHA > 9 THEN LET L = 0
50 NEXT L
60 PRINT "A ESCOLHA FOI ";ESCOLHA
70 END
```

Essas rotinas exemplificam igualmente outro aspecto da instrução INPUT. Esta instrução faz com que o programa seja interrompido e espere a entrada de um item pelo teclado. O BASIC só reconhecerá que o número foi fornecido por inteiro se a tecla RETURN for pressionada; assim, é necessário que você se lembre de pressionar a tecla RETURN após dar entrada ao número.

Um procedimento mais interessante para o usuário consiste em fazer com que o programa continue imediatamente após o fornecimento de um número válido. Para isto, usamos a função INKEY$, e o BASIC lerá um caractere do teclado toda vez que a função INKEY$ for encontrada. O programa, no entanto, não será interrompido, passando para a unidade seguinte, sem realizar um intervalo. Assim sendo, é comum o emprego da função INKEY$ no interior dos loops. O loop para verificar o pressionamento da tecla pode ser IF INKEY$ = "  " THEN..., ou seja, caso a tecla pressionada corresponda a "nada" (isto é, se nenhuma tecla for pressionada), retornar e verificar novamente. Um loop que serviria a nosso objetivo seria o seguinte:

```
LET I = 0
FOR I = 1 TO 1
LET A$ = INKEY$
IF A$ = "  " THEN LET I = 0
NEXT I
```

O único inconveniente no emprego da função INKEY$ é que ela fornece um caractere do teclado, em vez de um caractere numérico. Quando há uma estrutura em que é feita uma entre várias escolhas (um desvio multicondicional), é mais fácil, em BASIC, usar números em lugar de caracteres. É neste ponto que as funções NUM ou VAL do BASIC mostram sua utilidade. Os números apresentados em séries de caracteres são convertidos em números "reais" (isto é, valores numéricos e não códigos ASCII, que representam numerais). São utilizadas da seguinte forma:

```
LET N = VAL(A$) ou LET N = NUM(A$)
```

Mediante as funções NUM ou VAL, podemos fazer com que o programa converta as entradas em variáveis numéricas, usando a função INKEY$. Este procedimento elimina a necessidade de usarmos a tecla RETURN depois de apertarmos a tecla do número. Todavia também é aconselhável a verificação fora do limite.

O fragmento do programa, abaixo, inclui dois loops, um alojado no interior do outro. O loop interno espera o pressionamento da tecla; o externo converte a série em um número e verifica se está dentro do limite:

```
FOR L = 1 TO 1
  PRINT "ENTRAR ESCOLHA (1-9)"
    FOR I = 1 TO 1
      LET A$ = INKEY$
      IF A$ "  " THEN LET I = 0
    NEXT I
  LET ESCOLHA = VAL(A$)
    IF ESCOLHA < 1 THEN LET L = 0
    IF ESCOLHA > 9 THEN LET L = 0
NEXT L
```

Por fim, reproduzimos um programa completo em BASIC para o módulo *ESCOLHA*, inclusive a entrada fictícia e as sub-rotinas para fins de teste. Ressaltamos, ainda uma vez, que os números de linha têm apenas objetivo de teste, e deverão ser remanejados quando o programa definitivo for montado.

```
10 PRINT CHR$(12)
20 PRINT "VOCE DESEJA"
30 PRINT
40 PRINT
50 PRINT
60 PRINT "1. ENCONTRAR UM REGISTRO (PELO
        NOME)"
70 PRINT "2. ENCONTRAR NOMES (POR TRECHO
        DE UM NOME)"
80 PRINT "3. ENCONTRAR REGISTROS (DE UMA
        CIDADE)"
90 PRINT "4. ENCONTRAR REGISTROS (DE UMA
        INICIAL)"
100 PRINT "5. LISTAR TODOS OS REGISTROS"
110 PRINT "6. ACRESCENTAR UM REGISTRO"
120 PRINT "7. MODIFICAR UM REGISTRO"
130 PRINT "8. ELIMINAR UM REGISTRO"
140 PRINT "9. SAIR E GRAVAR"
150 PRINT
160 PRINT
170 LET L = 0
180 LET I = 0
190 FOR L = 1 TO 1
200 PRINT "ESCOLHER DE 1 A 9"
210 FOR I = 1 TO 1
220 LET A$ = INKEY$
230 IF A$ = "  " THEN LET I = 0
240 NEXT I
250 LET ESCOLHA = VAL(A$)
260 IF ESCOLHA < 1 THEN LET L = 0
270 IF ESCOLHA > 9 THEN LET L = 0
280 NEXT L
290 ON ESCOLHA GOSUB
    310,330,350,370,390,410,430,450,470
300 END
310 PRINT "SUB-ROTINA FICTICIA 1"
320 RETURN
330 PRINT "SUB-ROTINA FICTICIA 2"
340 RETURN
350 PRINT "SUB-ROTINA FICTICIA 3"
360 RETURN
370 PRINT "SUB-ROTINA FICTICIA 4"
380 RETURN
390 PRINT "SUB-ROTINA FICTICIA 5"
400 RETURN
410 PRINT "SUB-ROTINA FICTICIA 6"
420 RETURN
430 PRINT "SUB-ROTINA FICTICIA 7"
440 RETURN
450 PRINT "SUB-ROTINA FICTICIA 8"
460 RETURN
470 PRINT "SUB-ROTINA FICTICIA 9"
480 RETURN
```

No próximo fascículo examinaremos estruturas em arquivo e iniciaremos o refinamento do procedimento INICIALIZAR.

## A propósito...

| Comando/Instrução/Função | Ação/Resultado | MS BASIC | Applesoft | TRS-80 | Sinclair |
|---|---|---|---|---|---|
| CONT | Continua a execução do programa | + | + | + | + |
| COS(x) | Dá o co-seno de x (x em radianos) | + | + | + | + |
| CSAVE "arquivo" | Grava um programa em fita | | | + | |
| CSNG(x) | Converte x para precisão simples | + | | + | |
| DATA lista de dados | Cria uma tabela de dados a ser usada pelo READ | + | + | + | |
| DEF FN nome(arg)=expressão | Define uma função numérica ou alfanumérica | + | + | | |
| DEF SEG=endereço | Define um segmento da memória | + | | | |
| DEFDBL lista de variáveis | Define as variáveis com precisão dupla | | | + | |
| DEFINT lista de variáveis | Define as variáveis como sendo inteiras | | | + | |
| DEFSNG lista de variáveis | Define as variáveis com precisão simples | | | + | |
| DEFSTR lista de variáveis | Define as variáveis como sendo alfanuméricas | | | + | |

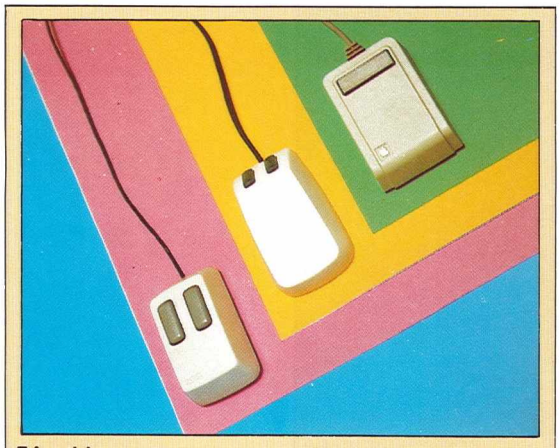
# Rato eletrônico

**Os projetistas de computadores pretendem substituir o teclado por algo de uso mais fácil. Um dos recursos é o "mouse".**

Não faz muito tempo, o acesso aos computadores era feito por meio de grandes máquinas de escrever eletromecânicas — os teletipos. Eram dispositivos barulhentos, incômodos e de pouca confiabilidade, que foram substituídos pela rápida e silenciosa Visual Display Unit (VDU), uma unidade de vídeo equipada com teclado. A VDU eliminou muitos problemas vinculados aos teletipos, destacando-se a grande quantidade de fita perfurada e papel desperdiçado, conforme a informação era fornecida.

Entretanto, o terminal mecânico e a VDU com teclado são limitados pelo seu formato de caractere por caractere e linha por linha. O usuário não pode executar movimentos na tela com rapidez — ao selecionar itens de um "menu", alterar dados ou mudar arquivos e programas —, em vista das limitações do cursor acionado. Só é possível se libertar do teclado quando se utilizam os terminais gráficos ou os jogos no computador com joysticks. Para as demais tarefas não há como escapar ao caminho de caractere por caractere e linha por linha quando se quer deslocar o cursor de um ponto para outro.

A maioria dos microcomputadores hoje existentes é equipada com controles de cursor com quatro direções, que são deslocados na listagem de programa ou em qualquer texto até a posição em que é preciso fazer alguma emenda. Entretanto, o usuário não pode fazer um movimento direto, até seu objetivo. Se o cursor de texto pudesse ser deslocado como um cursor gráfico, que é livremente manipulado sob o controle de um joystick ou track ball, o movimento dos dados seria consideravelmente mais rápido.



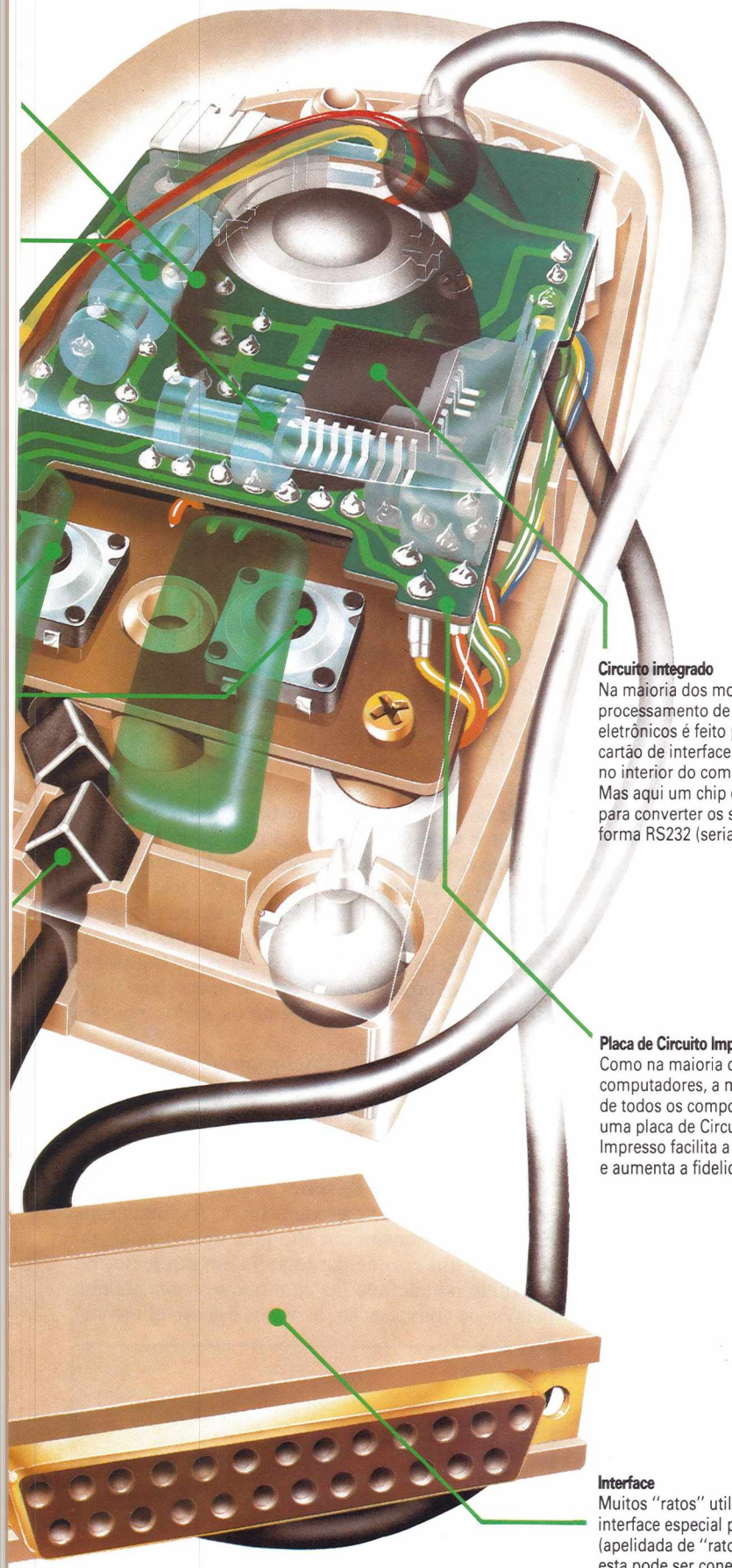
**Três ratinhos cegos**
Muitos microcomputadores mais recentes já utilizam o mouse, e alguns fabricantes oferecem essas unidades como complemento de máquinas já existentes. A maioria funciona por meio de um rolamento esférico na parte inferior e tem um, dois ou três botões SELECT.

Na década de 60, pesquisou-se pela primeira vez uma solução para este problema, no Instituto de Pesquisa de Stanford, na Califórnia; e o primeiro "mouse" — como se chamou o novo tipo de controle — foi patenteado em 1970. O dispositivo recebeu o nome de mouse (camundongo, ratinho) devido a sua aparência: é pequeno o suficiente para caber na palma da mão, possui uma "cauda" (o cabo) e os primeiros destes dispositivos tinham duas "orelhas" (botões de controle). Os track balls e joysticks convencionais não são utilizados porque a precisão que proporcionam para o posicionamento do cursor não é necessária.

O mouse funciona pela detecção de seu movimento sobre qualquer superfície plana, nas direções para diante/para trás e esquerda/direita, como também em combinações das duas. Estes movimentos são convertidos diretamente em deslocamentos do cursor (ou indicador, como também é chamado) na tela. Há dois métodos principais de geração dos sinais elétricos a partir do movimento do mouse. Em ambos, a superfície inferior do mouse tem uma esfera, apoiada na superfície em que o "ratinho" é deslocado.

A rotação do rolamento esférico do ratinho é transferida para rolos cilíndricos internos. Em um sistema, as extremidades desses cilindros são encaixadas em rodas codificadoras que possuem pistas alternadas de material condutor e não-condutor. Os impulsos recebidos são contados pelo software básico do mouse, capacitando-o a fornecer uma leitura





**Circuito integrado**
Na maioria dos mouses o processamento de sinais eletrônicos é feito por um cartão de interface montado no interior do computador. Mas aqui um chip é usado para converter os sinais na forma RS232 (serial).

**Placa de Circuito Impresso**
Como na maioria dos computadores, a montagem de todos os componentes em uma placa de Circuito Impresso facilita a construção e aumenta a fidelidade.

**Interface**
Muitos "ratos" utilizam interface especial própria (apelidada de "ratoeira"), mas esta pode ser conectada em qualquer porta RS232, usando-se o conector padrão de 25 canais.

direta da posição do cursor na tela. No outro sistema, dois discos fendidos estão encaixados nos cilindros. Uma luz é dirigida continuamente para os discos, a qual é detectada na outra face por uma célula fotelétrica. Atravessando as fendas, os impulsos luminosos convertem-se em sinais elétricos, os quais são tratados da mesma forma que os sinais do sistema mecânico.

Há também outros sistemas. Em um deles, por exemplo, o mouse é utilizado junto com um coxim especial, coberto por uma máscara de pontos. Uma luz no interior do corpo do mouse ilumina a área do coxim abrangida pelo mouse, e esta configuração é detectada por um chip especial de processamento óptico. Qualquer movimento do ratinho modifica a configuração detectada pelo chip e este pode calcular de imediato até onde se moveu o dispositivo, em qualquer direção. Tal sistema tem a vantagem de não apresentar nenhuma parte móvel, mas é bem mais caro que os outros.

Assim que o cursor for movido até o lugar desejado na tela, sua posição pode ser fornecida ao computador, pressionando-se uma das "orelhas" (botões) do mouse. O número de botões varia de um fabricante para outro. Alguns sistemas utilizam até três; a Microsoft, que desenvolve o software básico do IBM-PC, usa dois, enquanto o mouse do Lisa, da Apple, tem apenas um. Os botões também servem para seleção de itens de um menu — programas como o MultiTool Word, da Microsoft, usam este recurso — e para dar ao mouse o controle do movimento normal do cursor. Estes dispositivos são utilizados com software altamente sofisticado, como o existente no Lisa. Neste, é necessário pressionar o botão uma vez para selecionar um "ícone" (ver p. 262) de um menu de tela, e duas vezes para dar início à aplicação.

A maior vantagem dos ratinhos e do software produzido para complementá-los é que podem ser usados por quem não tem prática no teclado. Em vez de digitar o título de um programa, ou de pressionar determinadas letras ou números para selecionar funções, o usuário deve apenas mover o mouse, de modo que o cursor da tela indique a aplicação ou procedimento exigido, pressionando, em seguida, um botão para ativá-lo.

Infelizmente, o mouse não elimina de todo a necessidade do teclado — novos textos e números têm ainda de ser introduzidos no computador pelo método convencional. Mas torna o manejo dessa informação introduzida bem mais simples. Testes realizados pela Apple, durante o desenvolvimento do Lisa, demonstraram que o usuário sem prática em computador pode aprender a trabalhar com o software dirigido pelo mouse Lisa em apenas 15 minutos. O processamento de software deste tipo, em um sistema convencional, leva cerca de vinte horas para que o usuário se familiarize com seu manejo, por causa dos problemas envolvidos na aprendizagem do uso do teclado e da necessidade de aprender comandos longos e complexos.

Os ratinhos eletrônicos logo serão parte integrante de todos os microcomputadores. Eles são eficientes, fáceis de usar e não assustam uma pessoa intimidável tanto quanto um teclado QWERTY tradicional.

# Trabalho de detetive

**Quando a informação passa de um computador para outro, há o risco de ficar deturpada. Os códigos de Hamming podem detectar e corrigir o erro.**

Quase todos nós já ouvimos histórias de erros terríveis cometidos por computadores, como o que resultou na remessa por uma empresa de quinhentas cópias de um folheto à mesma pessoa. A verdade é que nem sempre a máquina tem culpa: o engano provém de falha humana, talvez um simples erro de digitação. Ocasionalmente, surge um erro porque o programa aplicativo não foi escrito para prever todas as eventualidades, como no caso de uma emissão pelo computador do último aviso de cobrança de uma conta de luz no valor de Cr$ 0,00.

Entretanto, às vezes os computadores cometem erros que não podem ser atribuídos à intervenção humana e com freqüência se manifestam em forma de "erros de bits". Um erro de bit ocorre quando em uma seção de dados um bit é transposto de 1 para 0 ou vice-versa. Isto pode acontecer quando surge uma falha num componente do hardware — um chip de RAM, por exemplo. É por essa razão que os microcomputadores passam por uma rotina de software que procura diagnosticar erros para serem "diagnosticados", sempre que a força é ligada.

A maioria dos erros de bit são "erros de soft" — os bits ficam "invertidos", apesar de toda a RAM ter passado pelo teste do diagnóstico. Os microcomputadores são projetados para operar em condições ambientais normais, mas, durante uma onda de calor no verão, é possível que a temperatura exceda o limite de tolerância estabelecido para os componentes. É provável que o dano assim causado não seja permanente, mas os erros de bit talvez resultem na súbita mudança de um caractere na tela — um "A" mudar para "B", por exemplo, ou, caso o bit faça parte de um indicador importante, ele pode fazer o programa sofrer um "colapso", havendo necessidade de reprogramar o computador.

Os erros de bit também podem surgir durante períodos de intensa atividade solar, quando partículas subatômicas penetram na atmosfera e interferem no fluxo de elétrons em um circuito miniaturizado. Em aplicações tais como sistemas militares, controle industrial, experiências científicas ou serviços bancários internacionais, os erros podem trazer conseqüências desastrosas, e por isso vários métodos foram adotados para detectá-los.

O método mais simples é a verificação de paridade (ver p. 253). Um método alternativo é o "checksum" (teste de soma), muito usado na gravação de dados em disco ou fita magnética. Os dados são sempre manipulados em blocos de 128 bytes, e o último a ser lido ou escrito será o byte do teste de soma. Este byte répresenta o resto da divisão da soma de todos os outros bytes (cada um com valor de 0 a 255) por 256. Eis um exemplo:

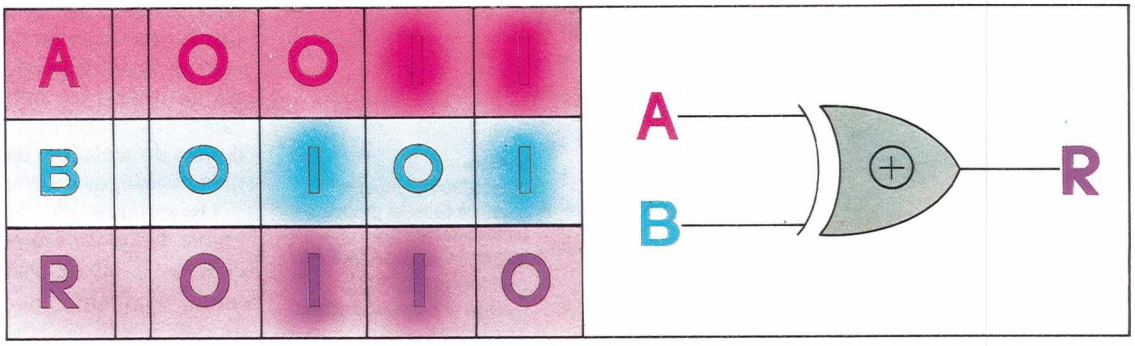
Dados: 114,67,83...(121 outros valores)... 36,154,198
Total dos 127 bytes = 16.673
Total dividido por 256 = 65, resto 33
Portanto, teste de soma = 33

O total dos bytes (16.673) é igual a 65 grupos de 256, mais um resto de 33 — este valor é inscrito no 128.º byte como um teste de soma. Quando o computador lê o bloco novamente, faz seus próprios cálculos de teste da soma com os dados e, se o valor diferir de 33, ele saberá que houve um erro de bit durante o processo de gravação.

Tanto com o método de paridade como no teste de soma, o computador não tem meios de saber qual bit foi alterado. Caso o erro tenha ocorrido na transmissão, o computador receptor pode solicitar que um determinado byte ou bloco de bytes sejam transmitidos de novo; no caso de um erro de gravação, talvez não haja maneira de recuperar os dados não alterados.

Quando os erros são absolutamente inaceitáveis, deve-se usar um sistema para detectá-los e corrigi-los. Os códigos de Hamming, que levam o nome de seu inventor, R. W. Hamming, dos Laboratórios da Bell de Telefone, desempenham esta função.

Todos os sistemas de correção de erros funcionam segundo o princípio da redundância. As linguagens humanas apresentam um alto grau de redundância: se ocorre um erro de datilografia em um texto, ou um ruído no telefone faz com que se perca alguma palavra da conversação, às vezes é possível recons-




**Exclusive-Or (Exclusivo-Ou)** Uma porta Exclusive-Or simples tem dois inputs e um output. Se os dois inputs estiverem em 0 lógico, o output será 0. Se cada input for 1, o output também será 1. Entretanto, se os dois inputs forem 1, o output será 0. Esta última condição diferencia a porta Or da Ex-Or (abreviação). A operação pode ser representada por uma tabela de validação. Onde uma Ex-Or tem mais de dois inputs, o output será 1 se houver um número ímpar de dígitos 1 no input. Estes dispositivos são os meios para criar os bits de paridade e os verificadores de erros.

tituir a palavra considerando-se o contexto da frase. Outras vezes, fazemos uma redundância extra, usada em ambientes "barulhentos": é o recurso de *a* de "amor", *b* de "bola" e *c* de "casa", em radiotelefonia, por exemplo.

Suponha que enviamos pelo nosso computador uma palavra que tem x bits de extensão, consistindo em y bits de dados reais e z bits redundantes (isto é, $x = y + z$). Em nossa explicação de paridade, tínhamos y igual a 7 e z igual a 1. Para os códigos de Hamming, z deverá ser proporcionalmente maior. Agora admitamos que um erro de bit pode ocorrer em qualquer um dos x bits (os z bits redundantes são tão passíveis de erro quanto os y bits de dados). Se a possibilidade de ocorrer um erro de bit numa palavra é de, digamos, uma em um milhão, então a probabilidade de dois erros numa palavra é de uma em um trilhão, e portanto podemos ignorar esta possibilidade.

Quando os dados são recebidos na outra extremidade, haverá x + 1 eventualidades: poderá não haver erros, ou o primeiro bit dos dados estará errado, e assim por diante até o x bit. Agora, com z bits redundantes podemos representar $2^z$ situações, de modo que a palavra seja à prova de um erro de bit:

$$2^z \geq y + z + 1$$

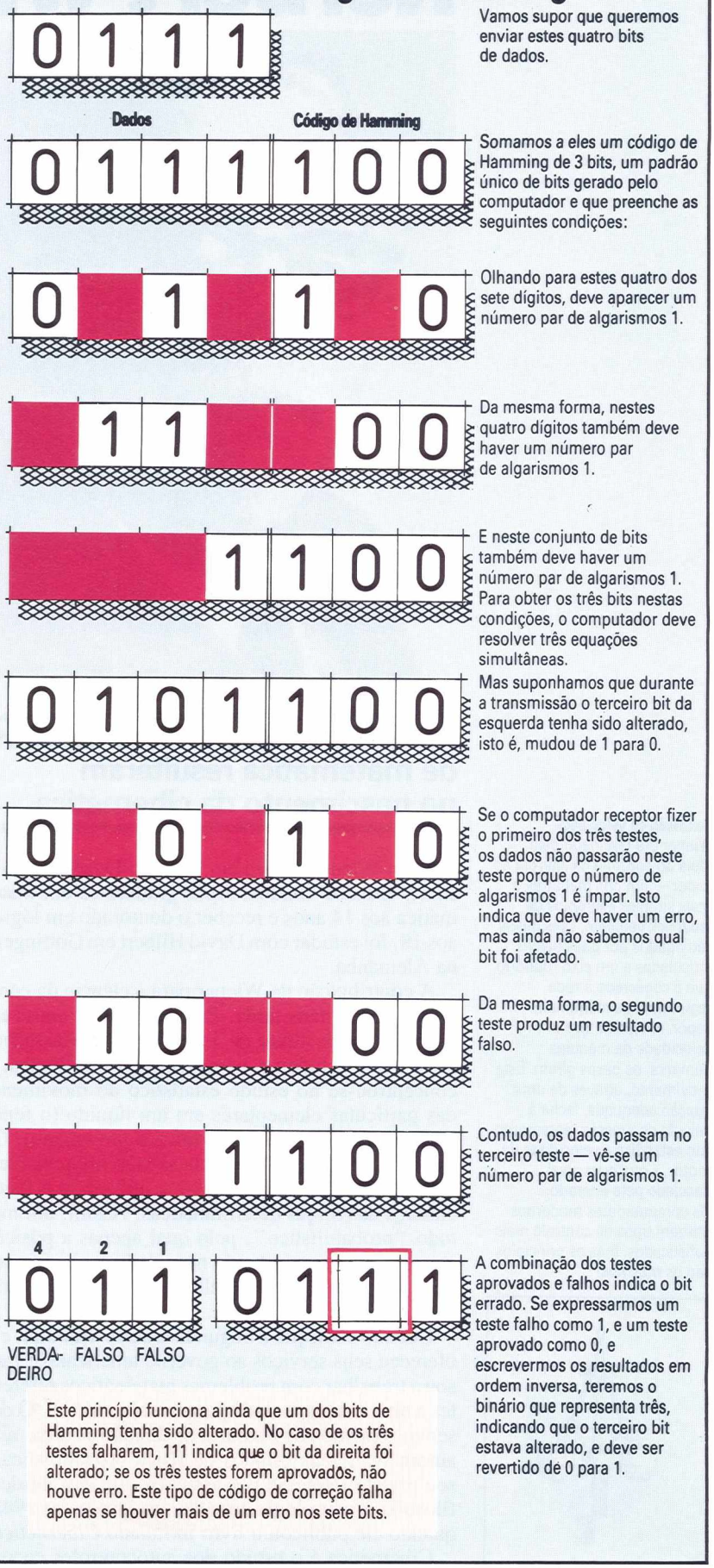
Se y é igual a 7 (pelos códigos ASCII), z deve ser igual a 4. Se y é igual a 4 (como no nosso exemplo do quadro), z deve ser igual a 3. Entretanto, se y é igual a 16, z aumenta apenas para 5. Segue-se que os códigos de Hamming são bem mais eficientes com palavras mais longas do que com palavras curtas.

Em um código de Hamming, cada um dos bits redundantes atua como uma verificação de paridade-par de uma combinação diferente de bits em uma palavra. Se algum bit ficar invertido na transmissão, um ou mais bits de checagem estarão errados e a combinação destes bits mostrará o bit errado na palavra (ver exemplos). O software do computador receptor pode então invertê-lo novamente, fazendo-o voltar à forma certa.

O ponto-chave do funcionamento dos códigos de Hamming são as diferentes combinações de bits sobre as quais cada bit de Hamming atua como verificador de paridade. O número total de bits é dividido em vários conjuntos diferentes, mas superpostos — planejados de modo que dois bits nunca apareçam na mesma combinação de conjuntos. O computador receptor desempenha funções de verificador de paridade nos mesmos conjuntos em que o dispositivo emissor já as desempenhou para criar o código de Hamming. Se algum dos bits, inclusive os de Hamming, foi invertido na transmissão, então um ou mais desses conjuntos não passarão no teste de paridade. A combinação dos testes não aprovados indica um bit único.

Alguns computadores empregam os códigos de Hamming até em suas operações internas de memória. Quando isto acontece, é possível remover um chip inteiro de RAM e constatar que o computador continua funcionando! Alguns computadores de uso militar levam o princípio de redundância ao extremo de duplicar todos os componentes do computador e comparar os resultados das duas metades após cada operação.

## Como funciona o código de Hamming




Este princípio funciona ainda que um dos bits de Hamming tenha sido alterado. No caso de os três testes falharem, 111 indica que o bit da direita foi alterado; se os três testes forem aprovados, não houve erro. Este tipo de código de correção falha apenas se houver mais de um erro nos sete bits.

# Norbert Wiener



## A criança prodígio cujos estudos de matemática resultaram no nascimento da cibernética.

Norbert Wiener nasceu em 1894 no Estado de Missouri, Estados Unidos. Após graduar-se em matemática aos 14 anos e receber o doutorado em lógica aos 18, foi estudar com David Hilbert em Göttingen, na Alemanha.

A contribuição de Wiener para a ciência da computação veio mais tarde. Durante muitos anos, ele trabalhou no Instituto de Tecnologia de Massachusetts, onde estudou a nova física probabilística, e concentrou-se no estudo estatístico do movimento das partículas elementares em um líquido (o fenômeno conhecido como movimento browniano). Os movimentos de uma partícula são tão imprevisíveis que era impossível descrevê-los utilizando a física clássica das forças determinísticas. Assim, um método "probabilístico", pelo qual apenas a posição *provável* de uma partícula num dado momento poderia ser prevista, era a melhor maneira de se resolver a questão.

Quando começou a Segunda Guerra Mundial, ele ofereceu seus serviços ao governo americano e passou a trabalhar com problemas matemáticos referentes a uma arma apontada para um alvo móvel. O desenvolvimento dos sistemas de direção de uma mira automática, seus estudos de física probabilística e seu grande interesse por assuntos que iam desde a filosofia à neurologia apareceram juntos em 1948, quando ele publicou o livro intitulado *Cibernética*.

Cibernética é o estudo dos autocontroles encontrados em sistemas estáveis, sejam eles mecânicos, elétricos ou biológicos. Foi Wiener quem visualizou que a informação como uma quantidade era tão importante quanto a energia ou a matéria. O fio de cobre, por exemplo, pode ser estudado pela energia que ele é capaz de transmitir, ou pela informação que pode comunicar. A revolução trazida pelo computador é em parte baseada nessa idéia: uma transferência da fonte de poder do proprietário de uma terra, indústria ou empresa para o controle de informação. A contribuição de Wiener não foi uma simples peça de hardware, mas a criação de um ambiente intelectual em que computadores e autômatos pudessem ser desenvolvidos.

A palavra cibernética deriva de um termo grego que significa "timoneiro, piloto". Wiener estudou o "piloto" ou peça mestra da máquina a vapor de James Watt, que regulava automaticamente a velocidade do engenho; e ele percebeu que, para os computadores serem desenvolvidos, teriam de assemelhar-se à habilidade dos seres humanos no controle de suas próprias atividades.

O termostato em um ambiente é exemplo de um sistema de controle. Regula o aquecimento, de acordo com as variações de temperatura, em relação a um nível considerado ótimo. O ser humano é necessário somente para estabelecer esse nível. Wiener chamou essa capacidade de auto-regulagem e controle de "feedback negativo"; "feedback" porque o output do sistema (o aquecimento) afeta o seu comportamento futuro, e "negativo" porque as modificações efetuadas pelo termostato restabelecem a temperatura do conjunto.

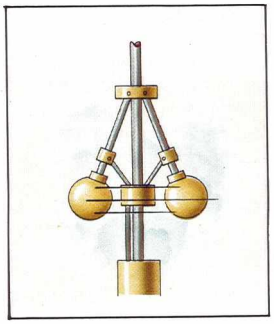
Um sistema que pode agir assim e também escolher sua própria temperatura (além de outros objetivos) é chamado sistema de "feedback positivo". Quando um autômato é capaz de realizar tudo isso e também reproduzir a si mesmo, então ele se aproxima da condição humana.

A teoria da cibernética de Wiener pode ser vista como uma superciência — a ciência das ciências — que estimulou as pesquisas em muitas áreas dos sistemas de controle e sistemas que trabalham com informação. Tudo é informação. Aquilo que sabemos a respeito das mudanças no mundo nos chega pelos olhos, ouvidos e outros receptores sensoriais, que funcionam como instrumentos de seleção de apenas certos dados de uma totalidade, que nos engolfaria em caso contrário.

A informação pode ser estudada, também, de forma estatística, independentemente do significado que possa ter. Por exemplo, pela observação da freqüência com que certos símbolos ocorrem pode-se quebrar vários tipos de códigos. Na língua inglesa, a letra *e* ocorre muito freqüentemente, e o *t* é a outra letra mais utilizada. Com a análise de extensas amostras de um código e comparando-as com exemplos típicos do inglês, é possível identificar letras-chaves e começar a decifrar o código.

Wiener morreu em 1964, antes que a revolução do microcomputador começasse. Mesmo assim, ele previu e escreveu sobre muitos dos problemas que iriam surgir nesta nova tecnologia.

**Restrição de velocidade**
Wiener era fascinado pela idéia do controle da energia do vapor — um dos melhores e mais simples exemplos de feedback negativo. Dois pesos são ligados por duas hastes articuladas a um eixo rotatório, que é conectado à roda reguladora da máquina a vapor. À medida que a velocidade da máquina aumenta, os pesos giram. Este movimento, através de uma ligação adequada, fecha a válvula de pressão lentamente. Isto estabiliza a velocidade do motor, a qualquer nível desejado pelo operador. Os computadores modernos utilizam tipos de controle mais sofisticados, mas os princípios são os mesmos.



# Comunidade "ligada"

## O desenvolvimento da TV via cabo possibilitou o surgimento de um novo meio de comunicação entre usuários de micros.



**Cidade em circuito**
Milton Keynes, uma cidade planejada, teve seu sistema de cabos de televisão instalado desde que foi construída. Vinte e duas mil casas recebem sete canais de TV (seis para transmissão ao vivo, um para apresentação de filmes) e seis canais de rádio em VHF. O próximo passo para uma rede de informações integrada será o fornecimento de energia elétrica e de gás, medido de modo centralizado, em vez de em cada prédio. Logo se acrescentarão redes locais em repartições públicas, bem como o sistema de apresentação de dados ("viewdata") acessível ao público.

Muito foi realizado a partir das vantagens proporcionadas pela chamada "revolução dos cabos", que permitirá o acesso a mais de vinte canais de TV em nossa casa, mas pouco se fala de um efeito colateral que modificará nosso modo de comunicação com a comunidade, pois um desses canais pode ser reservado para ligação entre microcomputadores. Já examinamos dois métodos de emprego de micros como terminais de comunicação: o sistema "viewdata" (ver p. 268), que possibilita o acesso a um grande banco de dados, e as redes locais (ver p. 218).

O emprego de sistemas tipo "viewdata", como o Mailbox, da Prestel, inglesa, ou o Videotexto da Telesp, de São Paulo, torna possível o envio de mensagens entre assinantes; e qualquer assinante pode adquirir mercadorias e serviços (férias, por exemplo) diretamente de um fornecedor central de informações. Entretanto, o emprego desses dois tipos de serviço é limitado. De modo semelhante, se seu computador for uma das estações de trabalho de uma rede local, você poderá se comunicar com qualquer outra estação — mas a maior e mais potente rede de microcomputadores abrangerá apenas 2 ou 3 km —, e seu banco, açougue e corretor de seguros dificilmente estarão incluídos nessa área. Uma pessoa poderá no futuro ser assinante do Videotexto da Telesp e participante de uma rede (de fato, este seria um modo muito sensato de dividir o custo dos serviços da Telesp). Porém, isso ainda está longe do que se entende por membro de uma comunidade cujas partes estão todas ligadas eletronicamente entre si.

Os mais importantes fatores físicos que impedem a criação de redes como essas, que abrangem uma cidade inteira, são a perda de sinal no processo de transmissão, tanto por meio de pares de cabos, quanto por cabos co-axiais (como os de antenas de televisores), e ainda pelas fibras ópticas. Este é o fator que atualmente limita o tamanho das redes locais, e o único modo de superar o problema é a inserção de um "reforçador" ou de estações reamplificadoras em pontos regulares na rede.

Outra questão a considerar é a "densidade do tráfego", ou o volume das informações a serem transportadas. Este aspecto influi na largura da faixa, ou amplitude das freqüências de transmissão necessária. De modo geral, necessitamos de 2 Hz de largura de faixa para cada bit transmitido por segundo. Uma transmissão de 300 bauds (300 bits por segundo) exige 600 Hz; uma de 1.200 bauds necessita de 2,4 KHz. A comunicação oral, feita habitualmente via telefone, requer um mínimo de 3 KHz; esta é a largura da faixa de uma linha telefônica. Entretanto, para transmissão de imagens para televisão em cores, são necessários 8 MHz — três mil vezes o total exigido para a transmissão da comunicação oral. Ou seja, a largura de faixa necessária para

transmissão de imagens de TV tem capacidade para transportar 3.000 conversas telefônicas individuais.

A capacidade de um meio de transmissão talvez seja mais bem apresentada pelo número de conversas telefônicas que pode comportar. O cabo de maior capacidade atualmente usado pela British Telecom pode transmitir 20.000 conversas ao mesmo tempo, mas testes experimentais avaliaram com sucesso que o cabo co-axial tem capacidade para 500 MHz, o que o torna capaz de transmitir 167.000 conversas telefônicas ao mesmo tempo. O cabo em si tem 1 cm de espessura. Compare esta capacidade com a tecnologia de fibra óptica, em que um único fio de vidro, mais fino que um fio de cabelo, possibilita a transmissão de até 10.000 conversas telefônicas.

O equipamento de comunicações já está à venda, faltando apenas a instalação da rede em alguma cidade. Veja como poderíamos estabelecer uma rede comunitária de comunicação por computador. Ao examinar as redes locais, observamos que a maior distância a que uma estação de trabalho deve estar em relação ao controle da rede é de 500 m, o que significa que nossa rede pode alcançar um círculo de 1 km de diâmetro. Também vimos que tem capacidade para admitir 254 participantes, reservando uma estação para serviços de arquivo (controle do disco comum) e outra para controle da impressora. Mas, se tivermos de reservar outra estação como canal de comunicação para mais uma rede, uma unidade bastante simples de programação nos permitirá fazer a união. A ligação exigirá dois equipamentos reservados para transmissão de mensagens de uma rede a outra, com comunicação através de suas respectivas portas paralelas. Quanto maior o número de equipamentos reservados com essa finalidade, tanto maior o número de redes que poderão ser interconectadas.



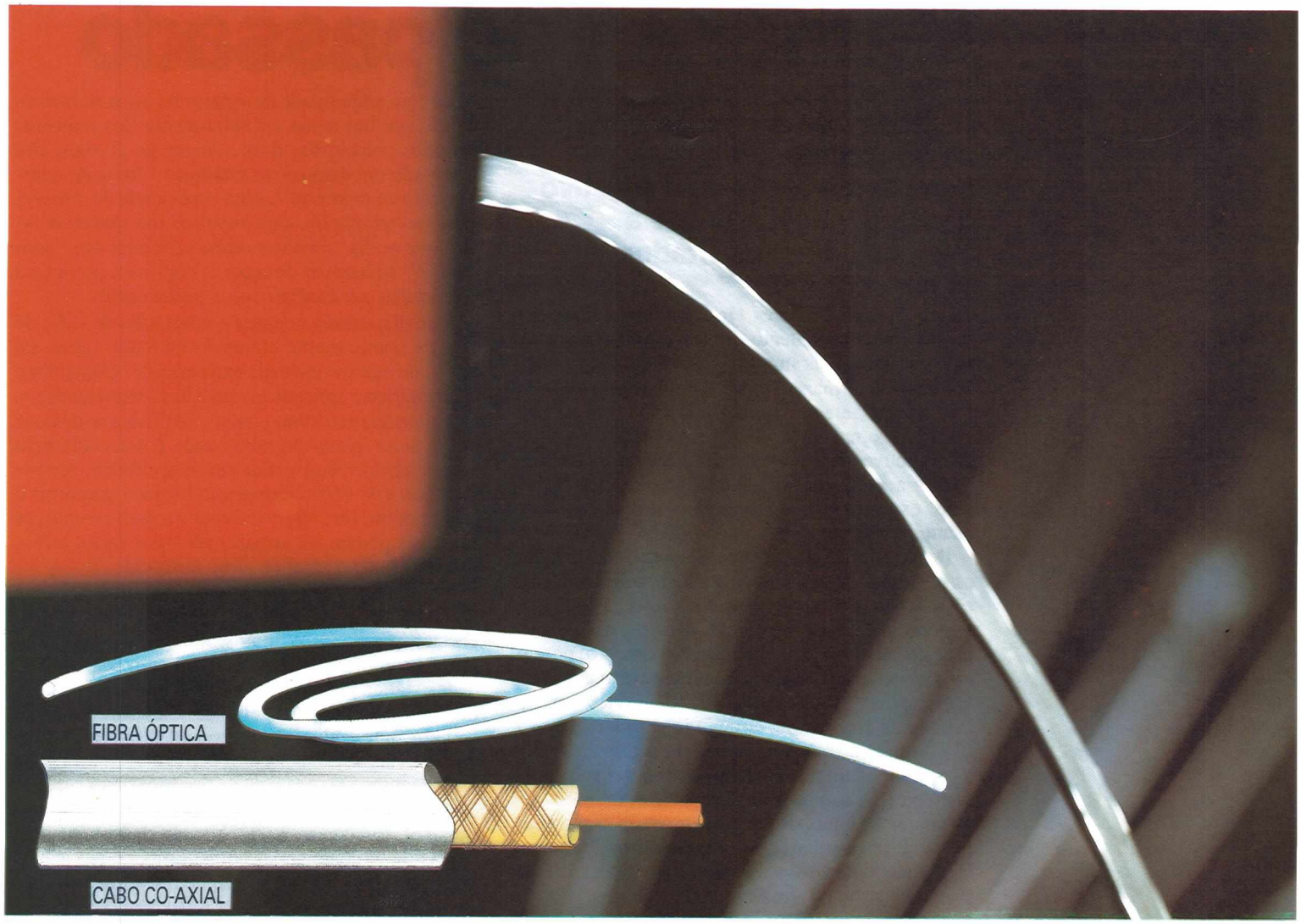

**Estação via satélite**
A MTV utiliza um sistema de distribuição via satélite para transmitir programas para suas 1.650 sucursais, que os retransmitem pelos cabos para as residências de 13,5 milhões de assinantes, nos EUA. A MTV oferece, por semana, 168 horas de programação estereofônica. Embora o serviço seja somente de emissão, a empresa incentiva certo grau de interação com a audiência por meio de números telefônicos para chamadas gratuitas. Com a introdução da TV via cabo, esse tipo de resposta imediata do espectador será muito comum.



Por enquanto, esta solução de interconexão de comunidades é apenas provisória, mas, se for aceita pelo público, poderá ser substituída por uma ligação em rede, projetada especialmente para essa finalidade. Todavia, sua concretização é pouco viável, porque exige que todos os participantes utilizem o mesmo tipo de microcomputador. Para sua real eficácia, esse tipo de rede precisa ser completamente "transparente", o que significa que deve possibilitar que um micro da linha Apple, por exemplo, se comunique com outro da linha TRS-80. Isso requer um sistema central de controle, que opere as conversões entre os vários protocolos usados pelas diversas marcas de computador. O controle também poderá ser o ponto de ligação com a Prestel (ou Videotexto, no caso brasileiro) e outros serviços de "viewdata", como o sistema bancário, as empresas imobiliárias, serviços de saúde etc., bem como todas as outras áreas da sociedade que foram computadorizadas, sem a preocupação de compatibilidade entre os equipamentos.

Quem forneceria, porém, os recursos financeiros para a instalação e manutenção de tal serviço? A experiência inglesa indica que este é o ponto de impasse. O governo inglês, ao fazer pela primeira vez a proposta de instalação de um sistema de TV que opera através de cabos, enfatizou as vantagens que esse sistema poderia trazer à indústria da tecnologia de informação e, como conseqüência, muitas empresas, usuárias em potencial desse sistema, iniciaram projetos de pesquisa. Uma rede de supermercados chegou a montar um esquema-teste de telecompras, com recursos financeiros próprios, para verificar sua praticidade e a reação do público. O esquema não resistiu ao período de experiência e o mesmo resultado — a pouca popularidade dos serviços — foi obtido com os outros projetos.

Ao ser apresentado ao Parlamento inglês o esboço de um projeto de lei que propunha o sistema de televisão via cabos, este ainda continha referências aos serviços de computação, como televendas e transações bancárias via microcomputador, mas era franco em salientar que se esperava da entidade encarregada da instalação dos cabos "que promovesse o fornecimento de serviços de computação de transmissão nos dois sentidos". A ênfase, desse modo, mudou da tecnologia de informação para a de entretenimento. De fato, os operadores de cabos estarão proibidos de fornecer qualquer forma de serviço telefônico, até mesmo os recursos para conferências transmitidas através do vídeo, embora o esboço de um documento de estratégias considere que a rede de cabos um dia assumirá todos os serviços de telecomunicações.




**Impulsos rápidos**
As fibras ópticas, primeiramente desenvolvidas em 1966 pelos laboratórios da Standard Telecommunication, na Inglaterra, baseiam-se no fenômeno de reflexão interna total. A luz introduzida em uma extremidade percorre a fibra de vidro com perda mínima de luminosidade, sendo refletida ao se chocar com uma "parede" externa, do mesmo modo que o faria em um prisma ou corrente de água. Para obter esse resultado, o nível de pureza do vidro empregado deve ser próximo a 100%. A luz utilizada é de laser infravermelho, porque o sinal deve pulsar (ser ligado e desligado) muito rapidamente.

Os cabos co-axiais, conhecidos pelo seu emprego em antenas de TV, são compostos de um único cordão de fio de cobre de alta condutividade, coberto por uma malha de fios extremamente finos, tecida em forma de tubo. Esses dois condutores são isolados um do outro.

A experiência em outros países foi semelhante. Nos Estados Unidos, onde cabos de TV foram há muito instalados, é ainda pequeno o emprego do sistema de cabos na comunicação digital para transmissão e recepção. Mesmo onde o sistema é utilizado, parece que isso se dá com finalidades bastante triviais. Em um incidente muito divulgado, pediu-se aos espectadores de uma apresentação da peça *Hamlet* que sugerissem como deveriam ser desenvolvidos os personagens da peça. A maioria votou para que a apresentação fosse interrompida!

Na Escandinávia e na Holanda foi realizado um movimento para interconexão comunitária de microcomputadores. Correio eletrônico para a comunidade local e redes de serviços de babás por computador estão agora em funcionamento e muitas comunidades chegam a efetuar plebiscitos não-oficiais com relação a questões de seu interesse. Os benefícios sociais foram, infelizmente, em grande parte esquecidos nas propostas dos sistemas de redes de cabos na Inglaterra. As vantagens das redes de comunicação seriam, sem dúvida, sentidas mais intensamente pelos membros da comunidade que têm hoje pouco interesse em adquirir um microcomputador para uso pessoal. Uma vez instalados os cabos interativos, haverá incentivo à indústria de serviços, como bancos ou empresas de telecomunicações, para que distribuam terminais a baixo custo, ou inteiramente gratuitos, como em algumas regiões da França, em que a rede telefônica nacional efetuou experiências com substituição de listas telefônicas.



**Uma imagem vale mil palavras**
É difícil determinar o número de bits digitais necessários para constituir uma imagem de televisão em cores, mas, tendo em conta que os 9 MHz de amplitude de faixa permitem a transmissão a 4,5 milhões de bauds e que cada imagem por inteiro é transmitida 25 vezes por segundo, um cálculo simples nos dá o número de 180.000 bits por imagem. Mil palavras exigem cerca de 60.000 bits.

# Memórias do passado

**O desenvolvimento de meios de armazenamento mais compactos tem sido o objetivo dos engenheiros desde a invenção do computador.**

Todos os computadores têm uma capacidade mínima de armazenamento de dados porque, mesmo nos processos mais simples como a adição, pode ser necessário o armazenamento de um dígito ou o seu "transporte" até que seja requisitado no estágio seguinte de computação. Desde a invenção do computador, houve contínua ampliação das capacidades de memória, e isto não só permitiu resolver problemas mais extensos como também questões anteriormente insolúveis.

Desde as primeiras pesquisas sobre a eletricidade, os cientistas procuraram investigar meios de utilizá-la como recurso para armazenamento de dados. A eletricidade pode ser concebida como um fluxo de elétrons ou como uma onda em movimento; porém, em ambas as descrições, o movimento é a característica implícita. A impossibilidade de reter um elemento que está sempre em movimento levou à adoção de métodos de armazenagem indireta. As baterias, por exemplo, armazenam energia sob forma química e as usinas hidrelétricas liberam a energia potencial da água quando esta atravessa as turbinas. Como teremos oportunidade de verificar, foram encontradas muitas soluções para o problema de lidar com dados sob a forma de sinais elétricos.

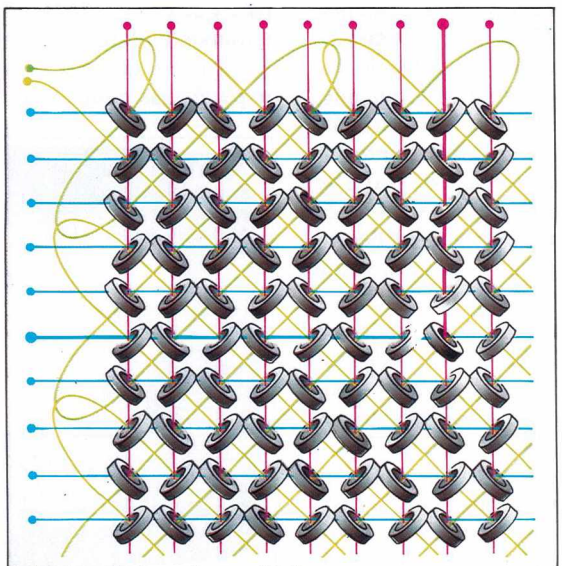
Na Segunda Guerra Mundial, fez-se muita pesquisa com a finalidade de separar no radar o sinal refletido por um avião em movimento dos impulsos emitidos por objetos fixos, tais como árvores. Foi inventado um dispositivo chamado "linha de retardamento de mercúrio", com capacidade de armazenamento temporário dos impulsos das ondas de rádio, que podia comparar ondas refletidas em "sondagens" sucessivas de radar e, com isso, permitia a eliminação das configurações permanentes.

Esse dispositivo consiste em um tubo de vidro de 1 m de comprimento contendo mercúrio, com um cristal de quartzo em cada extremidade. Quando um sinal elétrico é aplicado em uma das extremidades, o cristal de quartzo vibra e uma onda física se desloca até a outra ponta do tubo, onde é detectada pelo outro cristal e reconvertida em sinal elétrico. A onda leva cerca de 660 microssegundos para percorrer o tubo; porém, fazendo-a retornar sucessivamente, os impulsos podem ser armazenados por vários minutos, antes que o sinal se torne distorcido.

Na Inglaterra, outro método de armazenamento de dados foi inventado, na Universidade de Manchester, por F. C. Williams, que pesquisava o emprego, como meio de armazenamento, da eletricidade estática gerada na superfície interna dos tubos de raios catódicos (as telas de televisores). Neste método, um canhão de elétrons determina a estrutura das cargas estáticas na tela e essa estrutura pode ser detectada por uma rede de fios próxima à superfície externa da tela de vidro. Em 1947, tornou-se possível armazenar 2.048 bits de dados em uma única tela, durante várias horas. Embora, nesse processo, a velocidade de acesso de memória seja alta, as cargas estáticas têm de ser refrigeradas a cada 30 microssegundos, caso contrário serão perdidas.

O emprego de fitas magnéticas foi primeiramente experimentado com o equipamento LEO, na Inglaterra (ver p. 320), e, nos Estados Unidos, com o UNIVAC (Universal Automatic Computer, Computador Universal Automático), no final da década de 40. Foi a primeira técnica eficiente em que quantidades consideráveis de dados podiam ser armazenadas a baixo custo. Este método consiste no armazenamento de 8 bits em uma estrutura posicionada transversalmente na fita, com 2.360 estruturas por centímetro. Com um comprimento total de 61 m de fita, ou mais, tornaram-se possíveis memórias permanentes de, no mínimo, 1 megabyte. Todavia, mesmo com os mais rápidos motores impulsionadores, o tempo de acesso aos dados no meio da fita requer alguns segundos. Portanto, este tipo de armazenamento é adequado para arquivos em que os dados são requisitados de modo seqüencial, como cálculo da folha de pagamento de uma empresa. A fita pode ser acelerada fazendo flutuar a "cabeça", que detecta os sinais magnéticos sobre um colchão de ar, e assim é diminuído o desgaste causado pelo atrito.

Memória com alta capacidade, alta velocidade de acesso e baixo custo por bit foi inventada no Instituto Massachusetts de Tecnologia, em 1950. A memória de "núcleo" empregava anéis de ferro ("tori") presos a uma rede de fios que se intersec-



**Memória de núcleo**

Anéis de ferro (tori) são montados em uma rede de fios. Qualquer anel pode ser endereçado individualmente com o envio de uma corrente elétrica pelos fios horizontais e verticais correspondentes. O fio leitor, que passa por todos os anéis, capta as alterações do fluxo magnético no anel, indicando se o dígito armazenado é 1 ou 0.

cionam: o metal é magnetizado pela passagem de uma corrente elétrica através do sistema. O campo magnético do anel pode tomar duas direções, e estas são empregadas para representar os dois estados binários: 0 e 1. A direção do campo magnético do anel é alterada por meio da aplicação de corrente elétrica apropriada. E, do mesmo modo como a corrente induz magnetismo, também o inverso ocorre: quando a direção do magnetismo muda, uma corrente inversa, pequena porém detectável, é induzida. O campo magnético só mudará quando for aplicada uma corrente acima de certo limite. Com o fornecimento de pouco mais da metade da corrente limite, através de um fio vertical, e da outra metade, por um fio horizontal, apenas um torus (anel) na rede receberá corrente suficiente para mudar de direção.

Posteriormente houve a redução do diâmetro de cada anel — de cerca de 4 mm para alguns centésimos de milímetro —, diminuindo, assim, a capacidade energética necessária para o registro de dados em uma célula. Os dados são lidos em uma célula individual da memória, por meio da tentativa de movê-la. Pela observação de uma corrente induzida descobre-se se a célula se moveu ou não e, com isso, seu estado inicial pode ser deduzido. Entretanto, este método coloca todas as células em um estado magnético e, desse modo, após cada leitura, os dados têm de ser registrados novamente.

Com a integração em larga escala de transistores em um único chip, um dos primeiros métodos de conservação de dados — utilizado no ENIAC, o primeiro computador eletrônico — foi retomado: o circuito biestável (''flip-flop''), um dispositivo que se mantém estável em um entre dois estados possíveis. Ele é construído a partir de dois interruptores eletrônicos unidos na parte posterior, onde a saída de cada um é alimentada na entrada do outro. Desse modo, um impulso pode ser ''retido'' no circuito biestável até sua utilização.

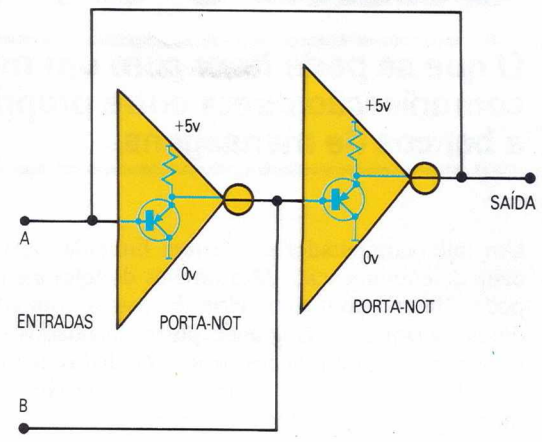
Cada bit exige dois dispositivos interruptores e por esta razão os primeiros circuitos biestáveis montados com válvulas eram caros, de pouca confiabilidade e queimavam com freqüência. Contudo, com o desenvolvimento dos circuitos integrados, nos quais centenas de milhares de interruptores transistorizados são incorporados em um único chip, o circuito biestável tornou-se novamente importante.

Os chips de RAM ''estáticos'', presentes em todos os primeiros microcomputadores, possuem milhares de circuitos biestáveis — um para cada bit. A maioria dos equipamentos mais recentes, todavia, tem memória RAM ''dinâmica''. O usuário não pode perceber nenhuma diferença, mas a memória dinâmica é mais rápida e de produção mais barata. É muito semelhante a um conjunto de capacitores que podem, cada um deles, armazenar uma carga elétrica. O problema está em que cargas armazenadas tendem a vazar; assim, os conteúdos devem ser ''refrigerados'' centenas de vezes por segundo através de um sistema de circuitos incorporado ao chip.

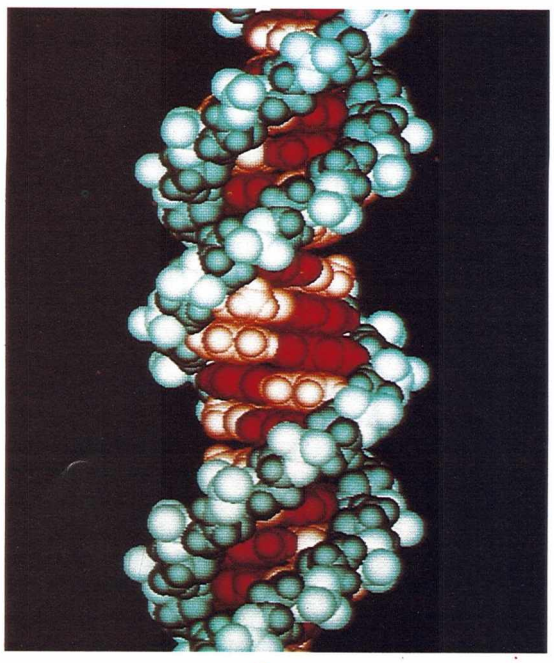
Os tipos de memória que surgiram mais recentemente são os de ''bolha'' e os laser ópticos. Estes últimos são semelhantes a discos utilizados para armazenamento de música ou videofilmes e são lidos por um feixe de laser. A memória de bolha consiste em pequenas áreas magnéticas ou bolhas criadas no interior de um chip de material magnético. Essas bolhas podem ser deslocadas no interior dos circuitos, e a presença ou a ausência de uma bolha representa um dígito binário, 1 ou 0. O tamanho dos circuitos é pequeno, permitindo o acesso em um curto intervalo. Sua densidade é potencialmente tão alta que este tipo de memória poderá, no futuro, substituir os sistemas atuais de discos e de cassetes.

Também poderão vir a ser utilizadas as memórias criogênicas. A temperaturas próximas do zero absoluto, a resistência elétrica em determinado meio quase desaparece, o que possibilita a supercondutividade. Com isso, torna-se teoricamente possível o armazenamento de cargas equivalentes a um único elétron. As cargas com que os atuais chips de memória operam são muito baixas, mas cada uma ainda equivale a alguns milhões de elétrons. Com a supercondutividade obtida pelo resfriamento desses meios de memória, a temperaturas criogênicas em banhos de hélio líquido, sua velocidade e capacidade se elevarão muito além das atuais.

## Circuito biestável ("flip-flop")




Princípio de funcionamento da maioria dos tipos de memórias semicondutoras, o circuito biestável é construído a partir de dois inversores ou portas-NOT (o circuito de transistores e resistores é apresentado na cor azul). A saída de cada inversor é conectada à entrada do outro. Se A receber o valor lógico 1, a saída (output) terá o valor 1. Se B receber o valor 1, a saída será 0. Por ter dois estados estáveis, este circuito é chamado biestável.



**Biotecnologia**
Embora a microeletrônica tenha alcançado um nível jamais imaginado de miniaturização, uma operação em memória semicondutora exige o fluxo de muitos milhares de elétrons. Os cientistas estão tentando criar memórias que operem em nível molecular, ou seja, com elétrons individuais, e isto é mais um ramo da biotecnologia do que propriamente da microeletrônica. Sabe-se, por exemplo, que a estrutura da molécula do ADN armazena o código genético que controla o modo pelo qual nosso corpo se desenvolve. No entanto, ainda serão necessários muitos anos para que se torne possível reproduzir esta propriedade com aproveitamento prático.

# Quadro de avisos

**O que se pode fazer com um modem? Ele permite, além da comunicação direta entre proprietários de micros, o acesso a bancos de mensagens.**

Um microcomputador em si tem limitada capacidade de comunicação. Mas através do telefone ele pode "falar" com uma série de outros computadores, sejam eles de grande porte, instalados em universidades, ou grandes bancos de dados abertos ao público — como o Videotexto — ou simplesmente o microcomputador do vizinho.

## Primeiros passos no Brasil

Também no Brasil os quadros de avisos eletrônicos começam a entrar em funcionamento. A primeira comunidade informatizada do país — o Projeto Ciranda, da Embratel — já dispõe de um serviço desse tipo. Mais de 2.000 funcionários da estatal brasileira de telecomunicações com micros ligados ao sistema trocam recados entre si, não só numa mesma cidade, mas através de todo o território nacional. Basta para tanto que o interessado acione no seu micro o comando que dá acesso à "caixa de mensagens" do Ciranda. E este serviço deverá ficar à disposição de qualquer usuário de computador pessoal. A Embratel está abrindo o sistema para o público em geral. É o Projeto Cirandão, que entrou em operação em agosto de 1984. A Telesp, com seu Videotexto, também deverá dispor de um esquema semelhante. No início de 1984, já havia vários interessados em fornecer o serviço. Quando ele entrar em funcionamento, será possível a uma empresa com subsidiárias em diversas cidades reunir seus executivos sem deslocá-los para a sede. Os assuntos em pauta serão discutidos através dos micros ligados ao telefone, com uma vantagem adicional: as reuniões não terão horário preestabelecido, podendo estender-se por várias horas ou mesmo dias.

Na Inglaterra, com o Bulletin Board Services (Serviço de Quadro de Avisos) ou BBS, é possível deixar mensagens para outros usuários de microcomputadores, anunciar qualquer objeto à venda e até trocar programas. Um BBS equivale aos quadros de avisos forrados de cortiça encontrados em clubes e locais de trabalho. Geralmente são instalados e operados por voluntários que usam micros comuns, acrescidos de um disco com grande capacidade de armazenamento, e qualquer pessoa pode ter acesso a eles. A distância até outro computador não constitui problema: pode-se ter acesso a uma máquina na mesma cidade, no extremo do país, ou além-mar. A única restrição, obviamente, é a conta telefônica.

No entanto, antes que seu micro comece a "falar" no telefone, você precisa de uma peça de hardware para conectar os dois aparelhos. Esse dispositivo é o modem (ver p. 108), ligado na tomada serial (RS232), atrás do computador. Alguns micros não têm essa tomada e nesse caso você deve comprar um cartão de interface serial para substituí-la. O modem é ligado à linha telefônica de duas maneiras: diretamente, usando-se uma tomada extra de telefone, ou acusticamente, mediante um acoplador acústico. Há duas "velocidades" de modem (taxas de baud) de uso generalizado: 300 bauds e 1.200/75 bauds. Na Inglaterra, os operadores comerciais, tais como a Prestel e a British Telecom, usam a velocidade mais alta de 1.200/75, enquanto o BBS (usado por microcomputadores) utiliza 300 bauds. Significa 1.200/75 que a informação é enviada ao usuário a 1.200 bauds, para o computador central a 75 bauds. Por certo, é preferível um modem já adaptado para operar com as duas taxas de baud. O preço desse equipamento vem caindo consideravelmente, e pode-se adquirir um modem de 300 bauds, que funciona com bateria, por 150 dólares. Para o modem do tipo "conexão direta", aluga-se na British Telecom uma tomada extra para ligá-lo.

Na operação de um modem é necessário dispor de software apropriado, para que a máquina funcione como um terminal de computador. Há dois tipos de terminal — "dumb" (burro) e "smart" (inteligente). O "inteligente" pode carregar programas e guardar mensagens de outros computadores. Um "burro" não consegue fazer isso, mas é mais fácil de ser instalado e portanto mais adequado para as primeiras tentativas de comunicação. Este tipo de software é usado em microcomputadores "mais antigos", tais como o Apple II e o Tandy TRS-80, e em alguns modelos mais novos — o BTERM, por exemplo, funciona só com o BBC Model B. Para outros computadores, este software costuma ser distribuído gratuitamente pelos grupos de usuários. Quem não conseguir encontrá-lo, terá de escrever seu próprio software "emulador de terminais". O que se precisa é de um programa que envie todos os caracteres digitados no teclado para a interface RS232 e mostre na tela a informação recebida através dessa interface.

Para ilustrar como se usa um modem, passemos às diversas etapas do processo de comunicação com o BBS. Primeiro, precisa-se saber o número do telefone e os detalhes técnicos (como a taxa de baud) do computador ao qual se quer ter acesso. Após carregar e executar o software de comunicações no microcomputador, disca-se o número do telefone. Se o computador remoto estiver "escutando", ouve-se um sinal agudo no fone — é o "sinal de entrada" Usando-se um acoplador acústico, basta encaixar o fone nos bocais de borracha. No caso de um modem de conexão direta, aciona-se rapidamente o interruptor "LINE" ou "DATA" e recoloca-se o fone no gancho. De uma maneira ou de outra, a ligação será completada.

A primeira coisa que se vê na tela é uma "saudação" gerada pelo computador do outro lado da linha. Então, ele pede a identificação e/ou senha do usuário. Se o serviço chamado for privado e o usuário não for o assinante, provavelmente o usuário não

### TELECOM GOLD

**Correio eletrônico**
O Telecom Gold é um sistema de correio eletrônico da British Telecom. Trata-se de uma versão mais sofisticada do Bulletin Board Services, que tem como primeiro objetivo atender o mercado comercial.
Os usuários inscritos podem alugar uma "caixa do correio" em um minicomputador da British Telecom, onde outros usuários deixam mensagens Estas podem então ser lidas ou copiadas no computador do usuário. Pelo Telecom Gold tem-se acesso a outras redes eletrônicas em diversas partes do mundo.




irá muito longe — a não ser que seja muito bom em adivinhação ou muito paciente! Entretanto, muitos computadores permitem um acesso restrito a "convidados" que não são usuários registrados, e portanto vale a pena tentar usar algumas palavras como NEWUSER, GUEST ou HELP.

O BBS não apresenta este tipo de problema. É aberto a todos e gratuito (exceto a tarifa da chamada telefônica). No instante da comunicação, fornecem-se o nome e a cidade quando solicitados, e, uma vez "inscritos", responde-se a perguntas sobre detalhes do microcomputador, tais como largura da tela ou marca. Com isso, o computador "anfitrião" fica capacitado a identificar o usuário em chamadas futuras e a ajustar o sistema para que funcione adequadamente.

Uma vez feito isso, recebem-se algumas informações sobre o serviço, tais como horários de operação, detalhes técnicos e um tempo limite para a chamada — talvez 30 minutos. Então, pode-se passar ao menu principal, que consiste em uma lista de meia dúzia de comandos que se escolhe, apertando a tecla com a letra inicial correspondente. Por exemplo, para se registrar como um "Novo Usuário" digita-se N; para encerrar digita-se G, de "Goodbye" Outros menus aparecerão e deve-se continuar selecionando as opções até chegar à desejada. O BBS parece uma árvore — começa-se no tronco, que é o menu principal, e escolhem-se diferentes "ramos" com cada submenu. A mesma idéia é usada pelo Videotexto e pela maioria dos outros bancos de dados.

A seção "Novo Usuário" é para aqueles que querem inscrever-se no BBS. O nome e o endereço são solicitados e pode-se escolher a própria senha para usar nas futuras chamadas. O comando "Informação" fornece detalhes técnicos sobre o sistema. A seção "Utilidades" informa a duração da chamada e fornece pormenores de outros serviços do Bulletin Board Services.

A seção de "Avisos" contém mensagens públicas que outros usuários colocaram no sistema. O usuário também pode colocar seus avisos. A seção específica de "Mensagens" é para que ele leia recados particulares que lhe foram endereçados. Da mesma forma, pode-se enviar mensagens para outros usuários ou a um grupo de usuários que têm algo em comum (por exemplo, todos os proprietários de TRS-80). Estas funções talvez sejam os aspectos mais atraentes do serviço e são provavelmente responsáveis pela sua crescente popularidade.

Um serviço diferente, chamado Rewtel, oferece um banco de dados especializado em componentes eletrônicos e informação relacionada. Também atende a pedidos de itens do estoque feitos diretamente através do teclado, porém só de assinantes. Diferente do BBS, neste sistema utilizam-se "palavras-chaves" para especificar a área de interesse do usuário. Por exemplo, quando se quer ajuda para usar o serviço de compras, deve-se digitar HELP REWSHOP. Ele também apresenta um serviço BBS: introduz-se a palavra-chave CHALK e pode-se deixar uma mensagem. As pessoas que não são assinantes também usam o serviço e têm direito a três minutos do sistema. Distel e Maptel são serviços de bancos de dados semelhantes, destinados principalmente a atender encomendas de equipamentos eletrônicos e de computação. A vantagem destes sistemas comerciais é que operam 24 horas por dia e não ficam tão ocupados como o Bulletin Board Services.

Os computadores de grande porte já estão "no telefone" há algum tempo, mas apenas recentemente este tipo de comunicação tornou-se possível para usuários de micros, devido à crescente sofisticação destes aparelhos e à queda nos preços dos modems. Nos próximos anos, é provável que o modem passe a ser um acessório tão comum para o micro como a impressora e o gravador cassete.

**Troca de recados**
Uma das aplicações mais gratificantes de um modem é o acesso aos quadros de avisos eletrônicos, que são versões computadorizadas dos tradicionais quadros de avisos de clubes. As mensagens podem ser "afixadas" para serem lidas por todos ou por determinadas pessoas com as senhas certas. São anunciadas reuniões e vendas de equipamento de computador de segunda mão. Pode-se até carregar jogos ou outras listagens de programas para disco ou cassete.

# Controle editorial

**A maioria dos micros permite editar programas na tela, poupando tempo e trabalho.**

Todos cometem erros quando usam o teclado de um computador, e por essa simples razão é necessário ter recursos de edição. Existem diversas situações em que podemos querer mudar os dados de uma tela: corrigir um erro de digitação, alterar algum texto incorreto ou atualizar informações que mudaram depois de introduzidas.

Muitos programas aplicativos incluem alguma forma de "editor" especializado. As funções de edição de um processador de palavras, por exemplo, são planejadas para fazer as alterações necessárias nas diversas etapas de preparação de cartas e relatórios. Entre elas incluem-se a capacidade de eliminar frases, mudar parágrafos inteiros de lugar no texto e trocar todas as ocorrências de um nome ou frase por outros novos.

Entretanto, quase todos os microcomputadores possuem algum tipo de editor — embutido no sistema operacional em ROM — para preparar listagens de programas. Estes são extremamente vulneráveis a erros. Em qualquer programa ocorrem erros de sintaxe com grande regularidade e em sua operação certamente haverá "bugs" (erros) que deverão ser eliminados, sem falar nos ajustes que podem vir a ser necessários. Os recursos do editor do seu computador podem fazer muita diferença no desenvolvimento de um programa longo. Devemos ressaltar, no entanto, que um bom programador passa um tempo considerável testando a operação de seu programa no papel antes de digitar a listagem no computador. Considera-se um mau procedimento de programação digitar a primeira solução que vem à cabeça e passar 90% do tempo de desenvolvimento do programa corrigindo-o.

Há dois tipos de editor: "editor de tela" e "editor de linha". O primeiro é mais flexível e fácil de usar, mas o segundo é o mais comum nos microcomputadores. O editor de linha data da época em que toda computação era executada através de teletipos ou terminais de um computador remoto. Os teletipos tinham uma memória buffer para apenas uma linha de 80 caracteres (80 bytes). O programador podia obter uma lista impressa do programa inteiro, bastando para isso digitar LIST, mas se fosse preciso fazer uma correção na linha 120, por exemplo, ele teria de digitar toda a linha novamente. Em alguns sistemas, digitando-se EDIT 120, aquela linha específica seria impressa, e então as alterações ou anulações poderiam ser feitas usando-se as teclas "backspace" (retrocesso) e "rubout" (apagar), porque ainda não era possível fazer inserções. Outros comandos, como DELETE (para uma quantidade específica de números de linhas), foram acrescentados, mas a limitação de ter de chamar e modificar uma linha inteira continuava presente.

Os editores de muitos microcomputadores ainda se comportam como se tivessem somente um buffer de uma única linha, quando, na realidade, a tela inteira é mapeada na memória — cada localização de caractere corresponde a 1 byte da memória.

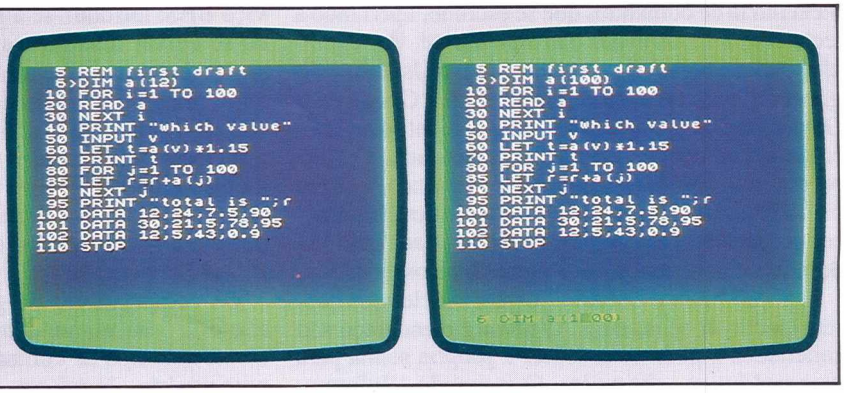
Um editor de tela é bem mais eficiente. Permite que você movimente textos e gráficos por toda a tela com facilidade. Toda vez que o usuário aperta RETURN, o editor lê a linha que está sendo indicada pelo cursor para o interpretador, onde ela é executada (se for um comando) ou introduzida no programa (se começar com um número de linha). Usando as quatro teclas com setas, o usuário pode movimentar o cursor para qualquer ponto do programa na tela e então inserir, retirar ou substituir caracteres à vontade.

Um editor de tela deve ser escrito em linguagem de máquina para conseguir a velocidade necessária, e pode apresentar recursos extremamente úteis. Os melhores editores de tela permitem a movimentação da listagem para cima e para baixo, bem como a inserção ou retirada de linhas inteiras ou caracteres individuais. Alguns até apresentam comandos semelhantes aos dos processadores de palavras que localizam e alteram todas as ocorrências de um determinado conjunto de caracteres.

Os editores estão ficando mais sofisticados e mais fáceis de usar a cada nova geração de computadores. Com a introdução do mouse (ver p. 296) e do software, que imita os processos manuais de cortar e colar pedaços de texto, o tempo despendido para editar um documento ou uma listagem até sua forma final vem diminuindo gradativamente.

## Linha por linha

Os serviços de edição do micro inglês Spectrum são consideravelmente melhores que os dos demais, embora estejam longe de ser tão fáceis de usar como um editor de tela completa. Para trocar determinada linha de um programa, o marcador de posição (>) que aparece entre o número da linha e a linha propriamente dita deve ser movido.



# Epson HX-20

## Este equipamento "de bolso" pode ser levado a qualquer parte, substituindo em viagens as pesadas máquinas de escritório.

O Epson HX-20 foi o primeiro computador realmente portátil. Programável em BASIC, tem uma versatilidade superior à das mais sofisticadas calculadoras. Com todas as suas partes embutidas num único módulo, pesa menos de 2 kg e cabe em qualquer pasta de executivo.

Quando foi lançado no mercado, equipamentos maiores, até então vendidos como portáteis, passaram logo a ser considerados "difíceis de carregar". O HX-20 despertou interesse de toda a gama de usuários de microcomputadores: aficionados, homens de negócio e engenheiros.

Por não ser baseada em um projeto tradicional, essa máquina dispõe ainda de pouco software. Entretanto, é um equipamento ideal para aprender a programar em BASIC. Freqüentemente, a intenção do comprador é escrever um programa que execute uma tarefa específica (talvez incomum): um programa de estimativas para vendedores de seguros, de apoio à navegação para iatistas, ou um para facilitar as anotações de jornalistas. Para qualquer um deles, o Epson é ideal.

A unidade vem acompanhada de um visor de cristal líquido com até quatro linhas de vinte caracteres ou gráficos simples com uma resolução de 120 x 32. Na maioria das aplicações, esse mostrador atua como uma "janela" que pode ser deslocada por meio das teclas do cursor (marcadas com setas) para apresentar qualquer seção de uma área de texto extensa com a qual o computador esteja trabalhando.

A impressora embutida utiliza papel comum em um cilindro de 5 cm de largura, no qual se pode imprimir até 24 colunas de texto.

A unidade de microcassete é um acessório opcional, geralmente apresentado como se fosse embutido, pois a maioria dos compradores assim prefere. O espaço por ele ocupado pode ser usado para cartuchos "solid-state" de software, apesar de, até agora, nenhum ter sido produzido. O microcassete é superior a um gravador comum. O computador pode identificar a posição da fita e movê-la automaticamente em avanço rápido para encontrar o programa ou os dados procurados.

A variedade de interfaces nas partes posterior e lateral do gabinete reflete a diversidade de utilizações visadas; há até uma tomada para uma leitora de código de barras (ver p. 21). Os 16 Kbytes de RAM podem ser expandidos até 32 Kbytes, por meio de um acessório na parte lateral.

Hardware e software a preço acessível possibilitam ao HX-20 a comunicação pelo telefone — tanto com um equipamento semelhante, quanto com um computador de grande porte para obter acesso a informações centrais.

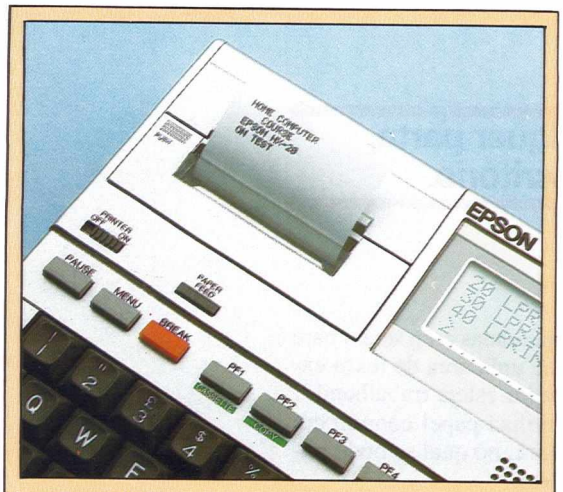



**Teclado do Epson HX-20**
As dimensões do HX-20 são determinadas pelo tamanho do teclado, igual ao das máquinas de escrever. É apreciado por datilógrafas, embora o toque seja um pouco diferente do usual.

Além dos numerais na carreira superior de teclas, pressionando-se a tecla "NUM" pode-se converter U, I, J, K, L, M em um teclado numérico reduzido. Isso torna mais rápida a entrada de grandes quantidades de dados numéricos.

As teclas de deslocamento do cursor e outras de edição estão no alto, à direita, junto a uma tecla SCRN, que desloca a imagem para cima e para baixo, na tela.

As cinco teclas de funções programáveis (PF1 a PF5) possuem um formato diferente das demais e duas delas têm função dupla: para controlar o microcassete e para copiar os conteúdos da tela na impressora.




**Impressora embutida**
Pequenas impressoras matriciais de pontos ou de rolos de tipos foram durante longo tempo parte integrante de máquinas registradoras e até de calculadoras. A Epson, que começou como fabricante de impressoras, rapidamente incorporou a idéia ao HX-20. Esta impressora matricial de pontos tem capacidade gráfica e também para caracteres.




**Entrada para energia elétrica**
Com o transformador conectado, o HX-20 obtém energia elétrica de uma tomada comum e recarrega suas baterias internas

**Botão de Reset**

**Conector para leitora óptica**
Uma leitora óptica pode ser conectada aqui, para leitura dos códigos de barra impressos em vários produtos nos EUA e na Europa, e em breve no Brasil

**Entrada para cassete**
Funciona com um gravador cassete comum e inclui o controle do motor. Seu desempenho, entretanto, não é tão eficiente como o do microcassete




**Memória externa**
O gravador microcassete embutido do HX-20, construído com base nas fitas cassete usadas em ditafones de bolso, é um aperfeiçoamento significativo em relação ao gravador comum, porque o computador controla a função de avanço/retrocesso rápido. O tempo gasto para alcançar determinado ponto na fita é, assim, reduzido drasticamente. Outra vantagem é que esta unidade se liga diretamente em um dispositivo no gabinete — não há complicação de fios.

**Placa de circuito impresso**
Na nossa ilustração, a PCI foi invertida para melhor compreensão do leitor. Na verdade, os chips ficam embaixo do computador e as tampas removíveis na parte inferior do gabinete permitem que sejam trocados sem necessidade de desmontar a unidade interna

**Microprocessadores**
Dois microprocessadores 6301 (fabricados pela Epson) controlam o computador e suas interfaces. Ao contrário do usual, cada um incorpora 4 Kbytes de ROM e 128 bytes de RAM, além dos chips externos

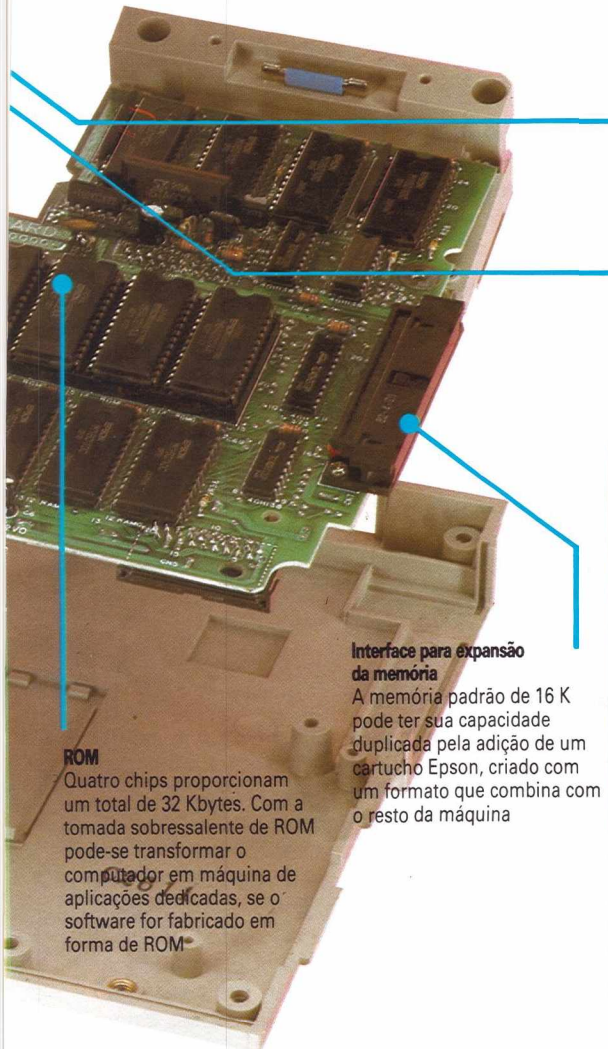
**RAM**
O HX-20 possui um padrão de 16 Kbytes de RAM, dispostos em oito chips de 2 Kbytes




**Ejetor de cartucho**
Permite a remoção completa do microcassete. Teoricamente, este módulo pode ser substituído por um cartucho "solid-state", se alguém quiser usar a máquina para uma só finalidade

**Ajuste do ângulo de visão**
Visores de cristal líquido têm de ser observados de um ângulo correto para serem legíveis. Girando este botão, o usuário pode regular a tela na inclinação mais adequada

**Interruptor de liga/desliga**

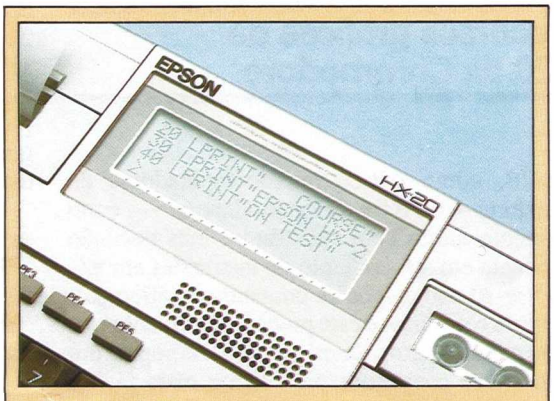


**Interface serial**
Apesar de a porta RS232 ser também uma interface serial, ela foi desenvolvida para controlar vários periféricos, que a Epson pretende criar para o HX-20

**Porta RS232**
Este é o modelo reconhecido como padrão para interfaces seriais e usado para conectar uma impressora ou um modem para comunicação com outro computador por telefone

**Interface para expansão da memória**
A memória padrão de 16 K pode ter sua capacidade duplicada pela adição de um cartucho Epson, criado com um formato que combina com o resto da máquina

**ROM**
Quatro chips proporcionam um total de 32 Kbytes. Com a tomada sobressalente de ROM pode-se transformar o computador em máquina de aplicações dedicadas, se o software for fabricado em forma de ROM




**Visor de cristal líquido**
Muito do espaço físico ocupado em um microcomputador é destinado ao tubo de raios catódicos do monitor. A Epson superou este problema no HX-20 com a substituição da VDU convencional por um visor de cristal líquido.

Visores deste tipo vêm sendo usados em calculadoras, mas esta utilização aqui representa um grande avanço, pois a matriz de 4 por 20 é ajustável para vários ângulos de visão.

# EPSON HX-20

**MICROPROCESSADOR**

Hitachi 6301

**CLOCK**

2,45 MHz

**MEMÓRIA**

16 K de RAM, podendo atingir 32 K por meio de uma placa de expansão; 32 K de ROM, que podem chegar a 64 K, pelo mesmo método.

**VÍDEO**

Visor de cristal líquido (ajustável ao ângulo de visão), com 4 linhas de 20 caracteres, que funcionam como uma "janela" que pode focalizar partes de uma área de até 255 x 255 caracteres. Todos os pixels no visor são endereçáveis, o que proporciona um máximo de resolução gráfica de 120 x 32.

**TECLADO**

60 teclas tipo máquina de escrever, cinco de funções programáveis e quatro botões para funções especiais.

**LINGUAGENS**

BASIC,
Interpretador FORTH

**PERIFÉRICOS**

Entrada para ampliação de memória, interface RS232 (para comunicação externa), serial (para discos ainda não disponíveis), leitora óptica.

**DOCUMENTAÇÃO**

O computador é acompanhado de dois manuais: um de operações e outro de consulta para BASIC. Ambos são bem abrangentes, mas prejudicados pela falta de índice. A Epson utiliza o BASIC da Microsoft, um padrão já reconhecido, e o manual da linguagem pode ser considerado um bom tutor e guia de consultas.

O manual de operações inclui informações especializadas: lista completa de conexões I/O para as interfaces RS232 e serial, níveis aceitáveis de sinais de intensidade, e tabelas de mapeamento de memória compreensíveis.

# Recursos modestos

## A produção de som no Spectrum inglês, da Sinclair.

O Spectrum inglês está se tornando rapidamente o mais popular microcomputador de entretenimento. Dispõe de excelentes recursos gráficos coloridos e alternativas de memória disponíveis a baixo custo. Infelizmente, alguns dos recursos foram descartados em favor das vantagens de preço. Os maiores inconvenientes estão vinculados ao teclado e à linguagem BASIC fora do padrão, mas pode-se dizer que seu aspecto mais frágil são as limitações para produzir som. O Spectrum possibilita recursos mínimos de geração de som e sua produção de "música" a partir de um único oscilador, que opera por "impulsos", é insatisfatória. A duração e a altura das notas podem ser controladas, mas não é possível mudar o tom ou alterar seu envelope (ver p. 276). Outro inconveniente encontra-se no output padrão, feito através de um diminuto alto-falante piezelétrico, interno, que produz um "bip" estridente. Entretanto, a Sinclair oferece outputs alternativos de sinal, através das portas "mic" e "ear" do cassete, adequadas para amplificação externa por meio de sistema hi-fi. Esse, porém, é um benefício questionável, já que resulta em baixa qualidade sonora. Os recursos sonoros do Spectrum têm efetivamente a vantagem da simplicidade em BASIC dos comandos associados BEEP e PAUSE, que possibilitam ao usuário a compreensão mais fácil dos princípios do som produzido por meio do computador. Além disso, o equipamento tem a notável amplitude de freqüência de 10 oitavas.

Para produção de som no Spectrum é necessário ligá-lo a um equipamento hi-fi ou tornar o gerador interno de "bip" audível mediante o seguinte comando direto, antes do processamento dos programas de som:

```
POKE 23609,100
```

Esse procedimento também aumenta o volume do "clique" em feedback, que ocorre quando uma das teclas é pressionada.

### Controle sonoro

Para instruir o computador quanto à saída de uma determinada nota, o comando BEEP deve ser empregado da seguinte forma:

```
BEEP d,n
```

onde "d" representa a duração da nota e "n" a sua altura. O valor da duração pode ser fixado entre 0,00125 e 10 segundos; a altura pode ser definida como o número de semitons contados a partir do dó central — o dó no meio da pauta; no piano, o dó menor central —, que recebe o valor 0, na amplitude de −60 a 69. Por exemplo, a instrução abaixo produz a nota lá, a 440 Hz, que corresponde a nove semitons acima da nota dó central, durante meio segundo:

```
BEEP 5,9
```

Para produção de uma série completa de notas com cronometragem precisa dos espaços entre elas, podemos empregar o comando PAUSE

```
PAUSE ms
```

# Imagens primárias

## Os recursos gráficos do Vic-20 da Commodore.

O Vic-20, como outros microcomputadores da Commodore — o PET e o Commodore 64 —, é um equipamento muito bem construído, mas isso não fica evidente em seu conjunto de instruções em BASIC. O Vic-20 não dispõe de comandos gráficos especiais para o usuário, e este precisa conhecer bem o funcionamento interno da máquina, ou comprar um dos acessórios projetados para tornar mais fácil a programação gráfica. No entanto, a Commodore proporciona um conjunto amplo de caracteres especiais que podem, com um pouco de engenhosidade, ser combinados para a obtenção de resultados interessantes.

Dispõe-se de dezesseis cores no Vic-20 e cada quadrado de caracteres pode apresentar quatro cores. A imagem na tela é constituída de 23 linhas e 22 colunas com células de caracteres de 8 por 8 pixels, que também podem ser apresentados em formato retangular de 16 por 8. O Vic-20 padronizado possibilita gráficos de alta resolução, mas essa programação é um tanto difícil para grande parte dos usuários.

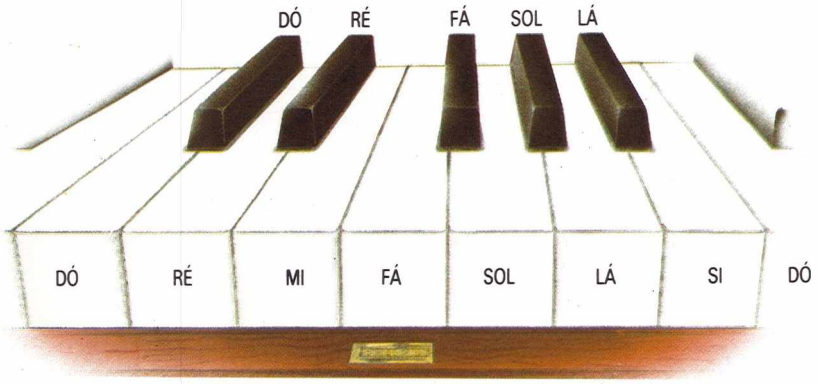
### Baixa resolução

São disponíveis caracteres alfabéticos em maiúsculas ou minúsculas e mais de sessenta caracteres gráficos especiais PET. Muitas das teclas do Vic-20 são marcadas com dois pequenos quadrados, cada qual apresentando um padrão específico. Além dos quadrados de meio e de um quarto de caractere, há símbolos de cartas de baralho, desenhos de tabuleiros de xadrez, círculos e inúmeros outros símbolos que podem ser combinados na tela para desenhar gráficos, fazer tabelas, produzir inscrições em caracteres largos e criar outros efeitos. Cada carac-

onde "ms" representa o tempo em unidades de 0,001 de segundo (isto é, em milissegundos). Para produção de uma oitava na tonalidade de dó maior (dó, ré, mi, fá, sol, lá, si, dó) a partir do dó central, com a duração de meio segundo para cada nota e pausa de um quarto de segundo entre as notas, pode-se empregar o seguinte formato:

```
10 FOR I = 1 TO 8
20 READ N
30 BEEP .5,N
40 PAUSE 250
50 NEXT I
60 DATA 0,2,4,5
70 DATA 7,9,11,12
```

Este programa é um bom exemplo do formato de uma escala. A oitava abrange a nota base (nesta escala, dó central) até a nota seguinte com o mesmo nome (a nota dó seguinte, acima) e inclui 12 semitons. É chamada de uma oitava porque consiste em oito notas em uma escala maior.




**Dó maior**

## Escalas e pianos

Mediante o comando INKEY$, o teclado do Spectrum pode ser usado de maneira a se assemelhar ao de um piano:

```
 10 REM **********************
 20 REM *OITAVA NO PIANO*
 30 REM **********************
 40 IF INKEY$ = "Q" THEN BEEP1,0
 50 IF INKEY$ = "2" THEN BEEP1,1
 60 IF INKEY$ = "W" THEN BEEP1,2
 70 IF INKEY$ = "3" THEN BEEP1,3
 80 IF INKEY$ = "E" THEN BEEP1,4
 90 IF INKEY$ = "R" THEN BEEP1,5
100 IF INKEY$ = "5" THEN BEEP1,6
110 IF INKEY$ = "T" THEN BEEP1,7
120 IF INKEY$ = "6" THEN BEEP1,8
130 IF INKEY$ = "Y" THEN BEEP1,9
140 IF INKEY$ = "7" THEN BEEP1,10
150 IF INKEY$ = "U" THEN BEEP1,11
160 IF INKEY$ = "I" THEN BEEP1,12
170 GOTO 40
```

Podem ser feitos muitos aperfeiçoamentos em programas como o acima, a fim de criar um "instrumento" de teclado mais eficiente. Geralmente, o Spectrum é de pouca utilidade como fonte de som por falta de recursos para esse fim. Entretanto, o equipamento é muito útil como auxiliar no ensino de música. O único modo de obter som satisfatório será a compra de hardware para geração de som compatível com o equipamento.

tere pode ser apresentado de modo invertido (preto sobre branco, ao invés de branco sobre preto), ampliando ainda mais as potencialidades do equipamento. Assim, com paciência e imaginação, obtêm-se imagens de alta qualidade.

Os caracteres aparecem na tela pelo emprego da instrução PRINT ou pelo armazenamento, mediante o comando POKE, dos códigos correspondentes nas memórias de tela e de cores do Vic-20. A versão Commodore da instrução PRINT possui alta capacidade, permitindo ao usuário a definição da cor de cada caractere. O movimento do cursor também é controlado a partir do interior da instrução PRINT, para produzir facilmente imagens com movimento. Armazenar caracteres na tela por meio do comando POKE não é tão rápido quanto sua impressão mediante o comando PRINT, mas em certas circunstâncias este método pode ser mais adequado.

## Alta resolução

Obtêm-se gráficos de alta resolução em equipamentos Vic-20 sem acessórios complementares, mas há memória disponível suficiente para utilização de apenas aproximadamente metade da tela. O processo pelo qual a alta resolução é obtida denomina-se "mapeamento de bit", técnica que possibilita ao programador o controle de cada pixel no interior de uma determinada área da tela. Cada célula de caractere no quadriculado de 23 linhas por 22 colunas consiste em 64 pixels dispostos em oito linhas com oito pixels. Há uma relação matemática entre as coordenadas referentes a um ponto de pixel (x,y) e o bit correspondente na matriz de caracteres. Esta relação pode ser empregada em conjunto com uma combinação de comandos POKE, na produção de imagens constituídas por pixels individuais.

## Cartucho Super Expander

A produção de gráficos de alta resolução por meio do mapeamento de bits pode tornar-se um processo longo e difícil. Uma abordagem alternativa consiste na aquisição do cartucho Super Expander, da Commodore, que permite ao usuário comandos de alta resolução, de música e de cores, em BASIC. Entre os comandos de alta resolução, estão o GRAPHIC, para determinar o modo da imagem, o POINT, para traçar pontos de pixel na tela, e o comando PAINT.

Há dois inconvenientes principais no emprego do cartucho: não existe comando UNPOINT (eliminação do ponto) para apagar os pontos de pixel e não é possível pintar ambos os lados de uma linha diagonal sem interferir na própria linha.

Os gráficos de baixa resolução do Vic-20 são bem elaborados e muito flexíveis, enquanto os de alta resolução implicam a aquisição de um cartucho para conexão, o que os encarece.

```
100 REM ** VIC 20 RABBIT **
110 POKE36879,110
120 PRINT"
130 PRINT"
140 PRINT"
150 PRINT"
160 PRINT"
170 PRINT"
180 PRINT"
190 PRINT"
200 PRINT"
210 PRINT"
220 PRINT"
230 PRINT"
240 PRINT"
250 PRINT"
260 PRINT"
270 PRINT"
280 PRINT"
290 PRINT"
300 PRINT"
310 PRINT"
320 GOTO320
```

**O coelho que corre**
Esta listagem de programa para o Vic-20 exemplifica a versatilidade do conjunto especial de caracteres da Commodore. A figura de um coelho é inteiramente construída por esse conjunto de caracteres predefinidos.

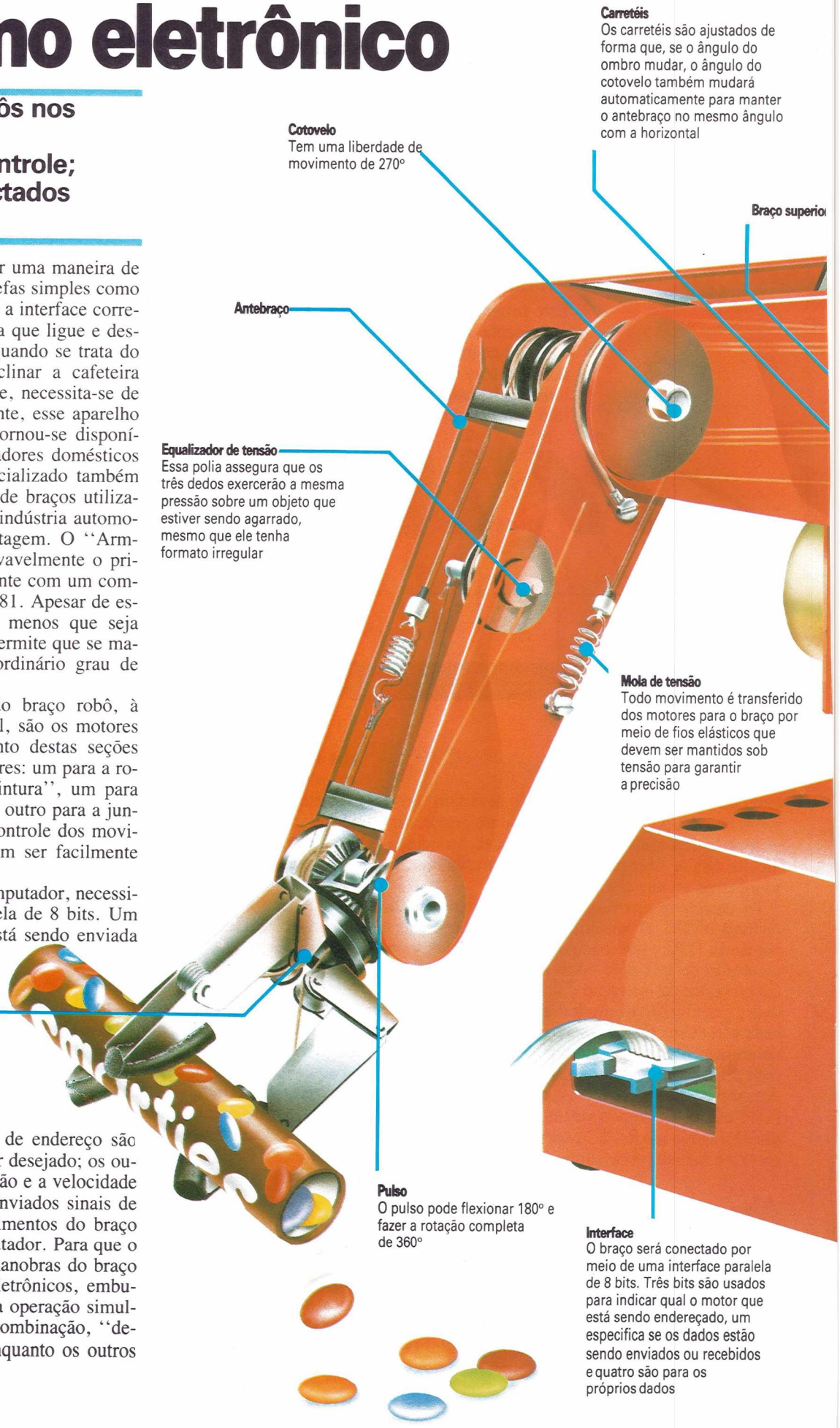
# Mordomo eletrônico

## Pequenos braços robôs nos ajudam a entender a programação de controle; eles podem ser conectados a micros domésticos.

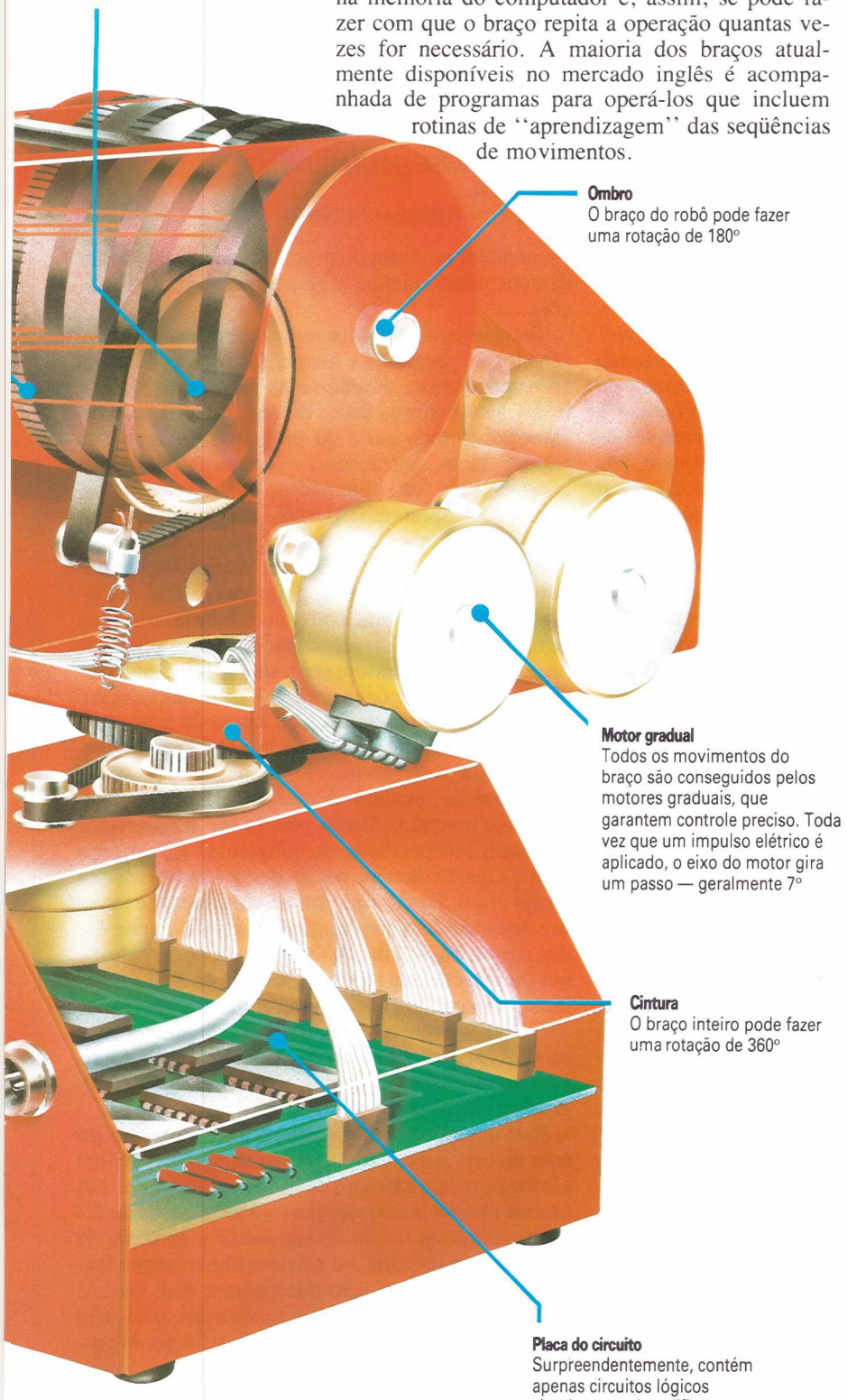
Você nunca pensou em encontrar uma maneira de fazer seu micro desempenhar tarefas simples como preparar um cafezinho? Havendo a interface correta, não é difícil programá-lo para que ligue e desligue a cafeteira elétrica. Mas, quando se trata do manuseio de objetos, como inclinar a cafeteira para entornar café quente no bule, necessita-se de um braço mecânico. Recentemente, esse aparelho — denominado braço robô — tornou-se disponível para os usuários de computadores domésticos na Inglaterra e logo será comercializado também no Brasil. São versões menores de braços utilizados em soldagem e pintura pela indústria automobilística em suas linhas de montagem. O ''Armdroid'', da Colne Robotics, provavelmente o primeiro braço robô de uso condizente com um computador doméstico, surgiu em 1981. Apesar de esse braço não ter locomoção (a menos que seja montado em um robô de solo), permite que se manipulem objetos com um extraordinário grau de precisão.

Os principais componentes do braço robô, à parte as próprias seções de metal, são os motores graduais que geram o movimento destas seções em etapas precisas. Há seis motores: um para a rotação do braço na altura da ''cintura'', um para controle da junção do ''ombro'', outro para a junção do ''cotovelo'' e três para controle dos movimentos da ''mão''. Todos podem ser facilmente dirigidos por um computador.

Para conectar o braço a um computador, necessita-se apenas de uma porta paralela de 8 bits. Um bit determina se a informação está sendo enviada ou recebida pelo robô, três bits de endereço são utilizados para selecionar o motor desejado; os outros quatro bits controlam a direção e a velocidade dos movimentos. Também são enviados sinais de relógio para sincronizar os movimentos do braço robô com as instruções do computador. Para que o processo seja mais rápido e as manobras do braço mais complexas, há ''fechos'' eletrônicos, embutidos no circuito, que permitem a operação simultânea dos motores em qualquer combinação, ''detendo'' informações para um, enquanto os outros estão sendo informados.



Para fazer com que o braço se posicione e apanhe um objeto, é necessário, primeiro, dividir o movimento completo em uma série de etapas simples. Cada motor precisa receber, separadamente, informações sobre um movimento preciso que, somado aos outros, realiza o movimento total do braço robô. Depois, esta informação é armazenada na memória do computador e, assim, se pode fazer com que o braço repita a operação quantas vezes for necessário. A maioria dos braços atualmente disponíveis no mercado inglês é acompanhada de programas para operá-los que incluem rotinas de "aprendizagem" das seqüências de movimentos.



**Mecanismo de transmissão**
A correia motriz de borracha dentada e grandes rodas dentadas permitem uma transmissão reduzida para que o braço seja repetidamente posicionado com precisão de 2 mm

**Ombro**
O braço do robô pode fazer uma rotação de 180°

**Motor gradual**
Todos os movimentos do braço são conseguidos pelos motores graduais, que garantem controle preciso. Toda vez que um impulso elétrico é aplicado, o eixo do motor gira um passo — geralmente 7°

**Cintura**
O braço inteiro pode fazer uma rotação de 360°

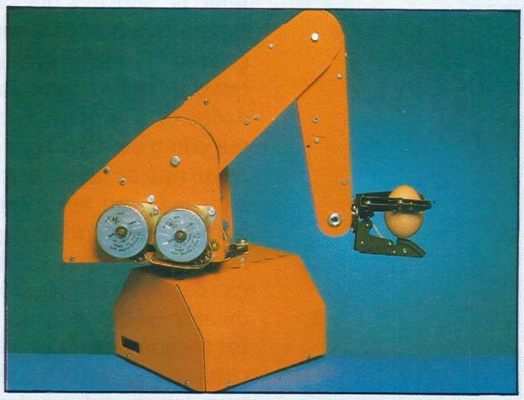
**Placa do circuito**
Surpreendentemente, contém apenas circuitos lógicos simples para decodificar os sinais do computador. Não há um microprocessador ROM ou RAM

Se o braço estiver manuseando objetos delicados — a peça de prova, geralmente, é um ovo —, o computador deve monitorar a pressão da garra. Se o aperto for muito suave, o ovo cai, e, se for muito forte, a casca quebra. Diversos métodos são utilizados para transmitir informação do braço para o computador, mas o mais comum envolve miniinterruptores simples. Eles podem ser ajustados para estabelecer os limites dos movimentos do braço (a maioria dos braços de baixo custo não possui sensores) ou embutidos na garra para detectar um limite de pressão.

O principal sistema alternativo para o de miniinterruptores, que é geralmente utilizado nos braços maiores, baseia-se no sensoriamento de pressão. Certos materiais alteram suas resistências elétricas quando sujeitos a mudanças de pressão, e estas flutuações podem ser medidas. Embora dispendioso, o método oferece resultados precisos.

Quando o programa não permite um feedback de informação do braço para o computador, é conhecido como "circuito aberto" ou determinístico. Em nosso exemplo acima, o programa, sem dúvida, resultaria em um ovo quebrado. Entretanto, se houver alguma forma de feedback que ajuste as ações realizadas, o programa será de "circuito fechado" ou estocástico. Nele, usam-se os microinterruptores ou sensores de pressão para limitar o fechamento da garra até um ponto em que o ovo é seguro mas não quebrado.

Muitos dos sistemas mais sofisticados de robô incluem múltiplos sensores para medir luz, calor e outras variáveis. Eles podem ser usados para rastrear o que está acontecendo enquanto o braço desempenha sua tarefa e prestam contas do que está saindo errado: por exemplo, um robô soldador fazendo furos em si mesmo!



**Compreensão da linguagem**
É relativamente simples fazer um programa para controlar um braço robô. No BASIC, a principal tarefa é possibilitar que seu computador aceite os comandos de controle do teclado e os transmita para o braço, através da porta, usando POKE. Da mesma forma, os dados fornecidos pelo braço podem ser lidos através da porta associada, utilizando-se a função PEEK. Se, acima de tudo, for exigida velocidade, é essencial a programação em código de máquina. FORTH é uma linguagem que oferece a facilidade de programação do BASIC e muito da velocidade associada a códigos de máquina. Seções de um programa, como sub-rotinas ou procedimentos, de preferência, são nomes dados que podem ser incorporados ao conjunto de comandos da linguagem. Isto a torna eficiente para aplicações especializadas como os programas de controle de braços robôs.

# Ampliação de arquivos

## Estabelecida uma estrutura geral, continuamos nosso projeto de programação examinando a manipulação de arquivos.

A agenda de endereços computadorizada que desenvolvemos nos últimos fascículos do curso de programação em BASIC é, na verdade, um tipo de banco de dados simples e, como tal, envolve o conceito de ''arquivo''. Este termo é empregado de vários modos relacionados, mas ligeiramente diferentes. Começaremos por analisá-lo com mais detalhes, a fim de que possamos depois empregá-lo com maior precisão.

Em linguagem de computação, um ''arquivo'' pode ser visto como muito semelhante a uma estante de arquivamento. Consiste em uma coletânea de unidades de dados armazenados em conjunto. Os computadores armazenam arquivos provenientes de fitas magnéticas ou de discos. Cada ''arquivo'' de dados recebe um nome individual, de modo que o computador possa acessá-lo, sempre que necessário. Os dados armazenados em uma fita cassete ou em disco flexível podem ser programas ou ''dados'' utilizados em programas. Tomando como exemplo a agenda de endereços computadorizada, os dados necessários constituem duas partes separadas: o programa em si e os dados com que o trabalha. O programa é o conjunto de instruções que possibilitam ao computador (e ao usuário) lidar e operar com os dados.

Os dados empregados pelo programa são o conjunto de registros que contém as informações que se espera encontrar em uma agenda — nomes, endereços etc. Também incluem certos tipos de dados aos quais o usuário normalmente não tem acesso. São os dados de ''preparo inicial'' utilizados pelo programa como auxiliar para o processamento. Exemplos desse tipo de dados são: ''flags''; as informações relacionadas ao tamanho efetivo do banco de dados (isto é, o número de registros nele contidos); se foi feita ou não alguma ordenação desde a última inserção de um novo registro; ou, possivelmente, uma indicação sobre quantas vezes determinado registro foi acessado ou impresso. O motivo por que dados como esses — e os que incluem os registros — devem ser operados independentemente do programa se tornará evidente tão logo comecemos a elaborar o programa.

Nos primeiros estágios do curso de programação em BASIC, empregamos as instruções READ e DATA como métodos para introduzir os dados no interior do programa. Esse procedimento só é conveniente se os dados não estiverem sujeitos a alterações, como o número de dias em um mês. Se os dados estiverem sujeitos a mudanças, o programa poderá indicá-lo na tela e empregam-se as instruções INPUT, INKEY$ ou outros métodos para fornecer dados ao programa. Um exemplo do uso adequado deste método de fornecimento de dados é o jogo de adivinhação de números, no qual parte do programa pode assumir a seguinte forma:

```
PRINT "PENSE EM UM NUMERO"
INPUT N
IF N < > NUMCOMP THEN...
```

No entanto, os dados do programa da agenda de endereços estão sujeitos a grande número de alterações.

Teoricamente, todos os registros podem ser armazenados no programa e fornecidos para as matrizes correspondentes, usando-se as instruções READ e DATA. Mas todos os dados do registro teriam de ser fornecidos como parte do programa. Sempre que fossem feitas alterações — nomes e endereços acrescentados ou eliminados, por exemplo — seriam necessárias modificações consideráveis. Isso exigiria, no mínimo, imprimir o programa, verificar onde as alterações deveriam ser realizadas, desenvolver novos segmentos de programa e então digitá-los. O maior problema, entretanto, estaria em que os novos segmentos não seriam módulos completos de programa que pudessem ser testados independentemente — as alterações estariam espalhadas ao acaso por todo o programa. O único modo de saber se o programa alterado funcionaria de forma conveniente seria processá-lo.

Felizmente, nenhum desses procedimentos é necessário, porque os dados podem ser armazenados de modo independente do programa. Isso é feito com a criação de arquivos de dados em fita cassete ou em disco. Esses arquivos são coletâneas de registros tratados de modo muito semelhante ao dos dados nas instruções DATA. O programa pode ''abrir'' um ou mais desses arquivos, extrair os dados neles incluídos (geralmente de uma matriz) e, a seguir, ''fechar'' o arquivo. Sendo necessária uma alteração dos dados, o programa abre um arquivo apropriado, extrai os dados, modifica-os e, em seguida, registra-os outra vez assim alterados no arquivo.

Para os equipamentos de computação que operam com discos, a localização de um arquivo específico e o registro ou a extração de dados nele contidos são procedimentos muito rápidos — a localização do arquivo leva uma fração de segundo e a extração de dados ou seu registro, no máximo alguns segundos. Sistemas de computação que operam com cassete podem ser muito mais lentos e exigir que o usuário retroceda a fita e espere que avance até o ponto correto do arquivo. Outra vantagem do emprego de discos está na possibilidade de se obter mais de um arquivo ''aberto'' ao mesmo tempo, ao passo que não são práticos os sistemas que operam com cassete.

Os arquivos consistem, portanto, em conjuntos de dados guardados em um meio de armazenamento

de grande capacidade disponível para emprego em um ou mais programas. Por exemplo, um programa de processamento de palavras pode necessitar de acesso ao mesmo conjunto de nomes e endereços para escrever cartas automáticas "personalizadas".

A manipulação dos arquivos ocorre de diferentes modos, de acordo com a versão de BASIC empregada. Para saber como seu computador lida com arquivos, verifique no manual que o acompanha as indicações sobre as instruções OPEN e CLOSE e experimente alguns exemplos. A descrição que apresentamos aqui é bastante genérica e preparada para dar uma visão geral do uso de arquivos.

Os arquivos podem ser seqüenciais ou aleatórios. Em um arquivo seqüencial, a informação é armazenada com a colocação da primeira unidade de dados em primeiro lugar, seguida pela próxima unidade e depois pela terceira, e assim sucessivamente. O arquivo aleatório é organizado de modo que o computador possa se dirigir diretamente à unidade de dados que se busca, sem necessidade de partir do início e passar por todos os dados até localizar a unidade procurada. O funcionamento do arquivo seqüencial é muito semelhante ao desenrolar de um filme no cinema, onde se começa do início e se acompanha a passagem dos dados ou eventos até o fim. O arquivo aleatório é mais parecido, digamos, com fazer passar uma fita de videocassete em casa, em que se pode avançar e retroceder a mesma e ver a parte desejada. Examinaremos apenas os arquivos seqüenciais, pois são mais adequados para sistemas que operam com cassete.

Suponhamos que você queira guardar um registro da temperatura média nos dias da semana:

| | |
|---|---|
| SEGUNDA-FEIRA | 13,6 |
| TERÇA-FEIRA | 9,6 |
| QUARTA-FEIRA | 11,4 |
| QUINTA-FEIRA | 10,6 |
| SEXTA-FEIRA | 11,5 |
| SÁBADO | 11,1 |
| DOMINGO | 10,9 |

Para simplificar, todos os dados serão tratados como dados numéricos, segunda-feira sendo o dia 1 e domingo, o dia 7. Os dados podem então ser representados da seguinte forma:

1.13,6,2.9,6,3.11,4,4.10,6,5.11,5,6.11,1,7.10,9

Serão necessários os seguintes estágios no programa para armazenamento dos dados em arquivo seqüencial:

ABRIR (OPEN) o arquivo
Fornecer os dados para o arquivo
FECHAR (CLOSE) o arquivo

Sempre que se usa a instrução OPEN, é preciso indicar se estamos fornecendo dados provenientes do computador para o arquivo (uma saída) ou dados provenientes do arquivo para o computador (uma entrada). Na linguagem BASIC da Microsoft isso é feito com o emprego das instruções OPEN "O" e OPEN "I". Um pequeno fragmento de programa para registrar os dados acima em um arquivo (em BASIC da Microsoft) seria:

```
100 OPEN "O", #1, "DAD.TEMP"
110 PRINT# 1.1.13,6,2.9,6,3.11,4,4.10,6,5.11,5,
    6.11,1,7.10,9
120 CLOSE #1
```

A palavra OPEN na linha 100 dá ao programa acesso ao arquivo. Essa instrução é seguida pelo sinal "O" para indicar que os dados provirão do programa para ser armazenado no arquivo. Esse procedimento vem seguido do sinal "#1", que informa ao computador que esse arquivo será referenciado pelo número 1 em nosso programa. Cada arquivo recebe um número arbitrário que posteriormente será empregado com as instruções INPUT# ou PRINT#, quando quisermos extrair ou fornecer dados nesse arquivo. Por fim, temos o nome do arquivo entre aspas. Nós o chamamos de DAD.TEMP para indicar que contém registros de temperaturas e que é um arquivo de dados e não um programa.

Damos abaixo um programa completo em BASIC da Microsoft para registro dos dados em um arquivo e a sua leitura e impressão subseqüentes:

```
100 OPEN "O", #1, "DAD.TEMP"
110 PRINT# 1.1.13,6,2.9,6,3.11,4,4.10,6,5.11,5,
    6.11,1,7.10,9
120 CLOSE #1
130 REM AS LINHAS 130 E 140 SAO LINHAS
    "FICTICIAS"
140 REM PARA REPRESENTAR O PROGRAMA
    INSERIDO
150 OPEN "I", #1 , "DAD.TEMP"
160 FOR X = 1 TO 7
170 INPUT #1, DIA,TEMP
180 PRINT "DIA # ";DIA,TEMP
190 NEXT X
200 CLOSE #1
210 END
```

Este programa abre um arquivo, com o número #1, e é denominado DAD.TEMP; fornece dados usando a instrução PRINT# e, a seguir, o ENCERRA (CLOSE). Mais adiante no programa, o mesmo arquivo é aberto com o emprego do número e também do nome do arquivo (o número não precisa ser o mesmo usado quando da criação do arquivo, mas o número utilizado nas instruções PRINT# ou INPUT# deve ser o mesmo dado ao nome do arquivo, quando este foi aberto). A instrução INPUT #1 na linha 170 indica que a entrada será proveniente de um arquivo com o número #1 (isto é, do arquivo DAD.TEMP) e não do teclado.

Vamos deixar, por enquanto, o exame da manipulação de arquivo e voltemos ao programa da agenda de endereços e de alguns dos elementos incluídos na subseção INICIALIZAR do programa. Primeiro, vamos ver a capacidade de memória necessária para um único registro no arquivo da agenda de endereços (o termo "arquivo" é aqui empregado em sentido específico ao banco de dados, como o conjunto de todos os registros relacionados, e não no sentido do sistema operacional, como um grupo denominado de dados, armazenado em fita ou disco).

O emprego de campos de determinada extensão consome uma parte da capacidade de memória, mas torna o procedimento de programação bem mais simples. Se reservarmos uma linha inteira para cada campo, com quarenta caracteres por linha, todos estes serão retidos em uma matriz, mesmo que a maior

parte da linha consista em espaços vazios. Em algumas versões de BASIC, todavia, quando as matrizes alfanuméricas são DIMensionadas, cada elemento pode totalizar 256 caracteres de comprimento. O dimensionamento apenas determina o número de elementos na matriz e não o tamanho deles.

Se você tem um equipamento em BASIC que pode lidar com matrizes multidimensionais, é possível empregar uma dimensão específica para cada um dos campos; porém, muitas versões de BASIC não o podem fazer. Assim, apresentaremos procedimentos alternativos. O método mais simples consiste em usar uma matriz alfanumérica separada para cada um dos campos. Mas aqui está um "truque", caso você queira matrizes multidimensionais e o BASIC de seu computador não possa manipulá-las.

O "truque" consiste em lidar com todos os elementos da matriz multidimensional como se fossem elementos de uma unidimensional. Por exemplo, uma matriz bidimensional com três linhas e cinco colunas poderá ser dimensionada da seguinte forma: DIM A(3,5) e conterá um total de quinze elementos: A(1,1) até A(3,5). Os mesmos dados poderão ser manipulados por uma matriz indexada comum, como esta: DIM A(15).

Se empregarmos uma matriz alfanumérica para cada campo, teremos de decidir sobre o modo como DIMensionar as matrizes. O mais simples é utilizar um tamanho fixo de matriz, mas isto limita a quantidade de registros que poderemos armazenar no banco de dados. Um procedimento melhor será a detecção do tamanho da matriz em função do número de registros que estão sendo utilizados. Todavia, nem todos os dialetos de BASIC admitem que o tamanho das matrizes alfanuméricas seja o que desejamos. Mesmo que admitam a presença de muitos registros no banco de dados, logo se esgotará a capacidade de memória disponível no computador. A seguir, apresentamos um programa que lhe permitirá encontrar o número máximo de elementos que seu computador poderá admitir. Muitas versões de BASIC, entretanto, admitirão tantos elementos na matriz quantos você desejar, até o ponto em que toda a capacidade de memória disponível tiver sido consumida. Toda vez que o programa apresentar a pergunta "QUAL O TAMANHO DA MATRIZ?", você deverá fornecer um valor maior, até finalmente obter uma mensagem de erro. A instrução CLEAR, na linha 100, tem como resultado a eliminação de uma matriz no final de cada passagem. Sem essa instrução, você receberá uma mensagem de erro na linha 30, ao tentar redimensionar a matriz.

```
10 READ D$
20 INPUT "QUAL O TAMANHO DA MATRIZ";A
30 DIM N$(A)
40 FOR L = 1 TO A
50 LET N$(L) = D$
60 NEXT L
70 FOR L = 1 TO A
80 PRINT L, N$(L)
90 NEXT L
100 CLEAR
110 GOTO 10
120 DATA "MICROCOMPUTADOR — CURSO
    BASICO"
130 END
```

Mesmo que cada elemento admita apenas quarenta caracteres, com cinco campos por registro, havendo 256 elementos reservados para cada matriz, o total de capacidade de memória para manter todos os dados dentro da memória principal se torna enorme. Se 1 byte é necessário para o armazenamento de cada caractere, precisamos de 51.200 bytes (5 x 40 x 256 bytes) só para armazenar os dados. Evidentemente, não é prático empregar tanta capacidade de memória principal para os dados, e é por isso que se usam arquivos separados de dados.

Infelizmente, como já mencionamos, a manipulação de rotinas de arquivo pode ser de execução um pouco difícil. Se quisermos evitar o uso de arquivos externos, a única alternativa é pôr os dados na instrução DATA, de modo que estejam sempre presentes no programa. Sem considerar o desgaste que impõe à capacidade de memória do computador, esse procedimento torna a modificação dos dados extremamente cansativa e aumenta a probabilidade de erro. Assim, é preferível usar arquivos de dados externos. Entretanto, se você experimentar alguns programas curtos para registrar dados para arquivos externos ou deles provenientes, o processo será mais claro e fácil de compreender.

Por exemplo, em equipamentos compatíveis com Apple, usando-se discos, podemos recorrer ao comando OPEN NOME ARQUIVO, S6, D2 para abrir um arquivo seqüencial chamado NOMEARQUIVO no disco (Slot 6) do drive 2 (D2). Num programa, o comando OPEN deve estar dentro de uma instrução PRINT e ser precedido pelo caractere CTRL-D. Para fechar o arquivo, o comando correspondente é o CLOSE NOMEARQUIVO. Para gravar os dados no arquivo seqüencial, usa-se o comando WRITE NOMEARQUIVO, e desta forma todos os demais dados das instruções PRINT seguintes serão gravados no disco, em vez de dar saída na tela. O comando WRITE deve estar igualmente dentro de um PRINT e precedido por CTRL-D. Assim, um pequeno exemplo de criação de um arquivo seqüencial em disco e gravação de dados num equipamento compatível com Apple seria:

```
100 D$ = " "
110 REM CTRL- D
120 PRINT D$; "OPEN ARQUIVO1, S6, D1"
130 PRINT D$; "WRITE ARQUIVO1"
140 FOR I = 1 TO 50
150 PRINT I
160 NEXT I
170 PRINT D$; "CLOSE"
180 END
```

Neste exemplo, serão gravados em disco no arquivo seqüencial ARQUIVO1 os números 1, 2, 3... 50. Para se recuperar do disco este arquivo, usa-se o comando READ NOMEARQUIVO, que deve ser utilizado da mesma forma que o WRITE, ou seja, dentro de uma declaração PRINT e precedido de um CTRL-D.

Nos equipamentos compatíveis com o Sinclair, particularmente no TK85, pode-se utilizar rotinas embutidas em linguagem de máquina para fazer gravação e recuperação de dados. Para tanto, cria-se um buffer, que servirá de área intermediária entre o programa e a fita cassete. Depois, utilizam-se as funções USR 8288 para gravar em fita uma matriz alfanumérica e USR 8305 para recuperar a informação

da fita. Um pequeno exemplo de gravação e leitura de um arquivo seqüencial em fita cassete no TK85 segue abaixo:

```
100 DIM C$(200)
110 C$ = "DADOS QUE SERAO GRAVADOS"
120 LET Z = 26
130 LET Z$ = "C"
140 LET CONTROLE = USR 8288
150 DIM T$(100)
160 LET Z$ = "T"
170 LET CONTROLE = USR 8305
180 PRINT T$
```

No exemplo acima, C$ é a variável alfanumérica de entrada e T$ é a variável de recuperação dos dados da fita. A variável Z$ indica o endereçamento no buffer ("C" para gravação e "T" para recuperação no nosso caso).

Finalmente, para equipamentos da linha TRS-80 (CP 500, Digitus etc.), a linguagem para gravação e recuperação de dados é muito próxima do exemplo inicial dado no BASIC da Microsoft. Para detalhes de programação e de operação com fita e disco, deve-se consultar o manual do fabricante do equipamento.

Até aqui, vimos como se pode transferir dados, através de matrizes, para arquivo em fita (ou disco) e vice-versa. No próximo passo veremos com detalhes o processo de INICIALIZAÇÃO, observando exatamente quantas matrizes serão necessárias, quantos elementos cada uma vai precisar e em que pontos do programa deve ser feita a transferência de dados, para elas e a partir delas.

## Exercício

Escrever um programa com os seguintes elementos:

```
INICIAR
  INICIALIZAR
  FORNECER OS DADOS
  ESCOLHER
    Gravar os dados
    Carregar os dados
    Sair do programa
ENCERRAR
```

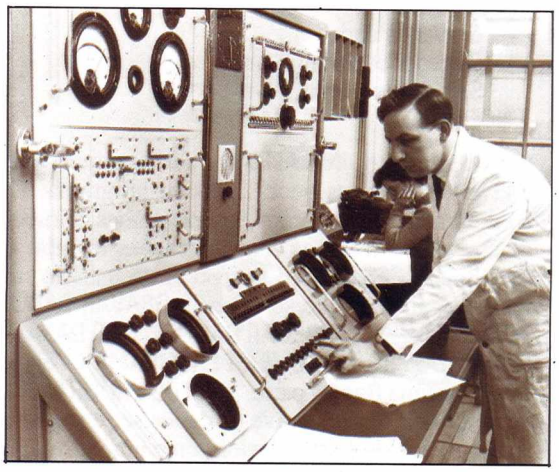
O procedimento INICIALIZAR introduz quaisquer variáveis e matrizes necessárias ao programa. Os dados abrangerão, digamos, quinze nomes, fornecidos através do teclado, com indicações na tela. O procedimento ESCOLHER fornecerá ao usuário um menu perguntando-lhe se "deseja GRAVAR DADOS (SAVE DATA)?", "CARREGAR DADOS (LOAD DATA)?" ou "SAIR DO PROGRAMA (EXIT PROGRAM)?" Veja se pode criar uma flag na parte "SAIR DO PROGRAMA", que automaticamente grave os dados se, e somente se, não tiver sido feita anteriormente uma instrução SAVE.

## A propósito...

| Comando/Instrução/Função | Ação/Resultado | MS BASIC | Applesoft | TRS-80 | Sinclair |
|---|---|---|---|---|---|
| DEL linha1, linha2 | Elimina as linhas entre linha1 e linha2 inclusive | | + | | |
| DELETE "arquivo" | Elimina um arquivo em disco | | + | | |
| DELETE linha1-linha2 | Elimina linhas específicas do programa | + | | + | |
| DIM lista de var(index) | Define dimensão das matrizes e reserva espaço de memória | + | + | + | + |
| DRAW x | Desenha uma figura de comando equivalente a x | | + | | |
| EDIT linha | Exibe uma linha do programa para edição | + | | + | |
| END | Pára o programa e fecha todos os arquivos | + | + | + | |
| EOF(a) | Indica uma condição de fim de arquivo em a | + | | | |
| ERASE matrizes | Elimina matrizes do programa | + | | | |
| ERL | Retorna a linha onde ocorreu o último erro | + | | + | |
| ERR | Retorna o código do último erro | + | | + | |
| ERROR n | Simula um erro de código n | + | | + | |
| EXEC "arquivo" | Executa um arquivo de texto em disco como se digitado | | + | | |
| EXP(x) | Eleva e à potência x | + | + | + | + |
| FILES "arquivo" | Lista os arquivos do disquete c/ arquivo | + | | | |
| FIX(x) | Trunca x para um inteiro | + | | | |
| FLASH | Provoca um piscar na tela a qualquer mensagem | + | + | | |
| FN variável (expressão) | Chama uma função definida anteriormente | | + | | |
| FOR variável=x TO y STEP z | Repete linhas de programa. O NEXT encerra o loop | + | + | | |
| FRE(x$) | Dá o espaço livre de memória para o usuário | + | | | |

# Uma casa de chá

## A computação comercial na Inglaterra começou em um lugar inusitado.



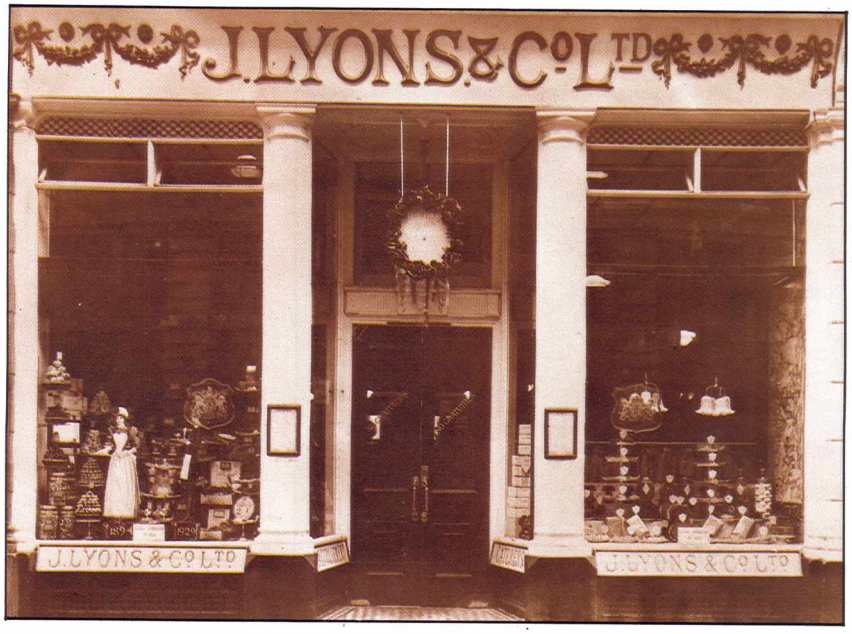
**Escritório eletrônico**
Ao contrário de todos os computadores anteriores, que se destinavam a aplicações científicas ou militares, o LEO 1 foi projetado para executar apenas operações aritméticas simples, mas com milhares de itens ou transações por dia.

Em 1947, foi tomada uma decisão pioneira: tentar construir um computador que pudesse automatizar o trabalho de escritório. Seria o primeiro computador de uso comercial do mundo. Essa decisão criativa partiu de uma fonte surpreendente: a J. Lyons, empresa proprietária de uma rede de casas de chá. As operações da Lyons envolviam grande número de pequenas transações e, para que o negócio fosse rentável, era necessário manter a contabilidade sob rígido controle. Mesmo depois da devastação causada pela Segunda Guerra Mundial, a empresa empregava mais de mil funcionários para controlar a contabilidade das casas de chá.

Na verdade, a Lyons já tinha uma longa tradição de inovações nos métodos administrativos: introduziu o uso de máquinas de calcular em suas lojas em 1896 e, por volta de 1930, fazia experiências registrando transações em microfilmes. Nessa época, criou também o primeiro centro de pesquisas de administração para introduzir novos métodos operacionais. A Lyons costumava enviar periodicamente representantes ao exterior para investigar novos desenvolvimentos que lhe pudessem ser úteis e, em 1947, dois funcionários foram aos Estados Unidos conhecer o novo "cérebro eletrônico". A descoberta mais proveitosa dos dois foi saber que um computador estava sendo construído bem mais perto de casa, em Cambridge, na própria Inglaterra.

A diretoria da Lyons determinou que fosse estudada a possibilidade de a empresa desenvolver seu próprio computador. A estimativa feita indicou que o computador poderia ser construído com um investimento de 135.000 dólares e que ele possibilitaria uma redução de 67.000 dólares anuais nos custos. Conseqüentemente, em outubro de 1947, a Lyons começou a trabalhar no projeto. O empreendimento era muito arrojado, pois na época o computador de Cambridge também estava em fase de projeto. A Lyons cedeu uma verba de 5.000 dólares à Universidade de Cambridge para ajudar a construir o aparelho que se tornou conhecido como EDSAC (Electronic Delay Storage Automatic Computer). A verba foi usada para comprar válvulas excedentes do governo. Em 1949, o EDSAC completou com sucesso seu primeiro trabalho — calculou uma tabela de números primos.

**Aplicação pioneira**
A tradicional casa de chá Lyons não parece o lugar mais apropriado para se fazer a primeira aplicação comercial importante de computadores, mas foi exatamente esse ramo de negócio, com seu número considerável de pequenas transações, que se prestou ao emprego de métodos computadorizados.




A Lyons analisou os problemas que seu computador teria de resolver, fazendo um esboço das rotinas que seriam necessárias. Estes estudos transformaram-se nos projetos para os primeiros programas e ajudaram a determinar o design do hardware. Logo, porém, ficou evidente que um computador de uso comercial era bem diferente de uma máquina destinada a pesquisas na universidade. O EDSAC fora projetado para executar operações matemáticas longas e complexas com um input de poucos números, e um computador de uso comercial tinha de resolver problemas que eram exatamente o oposto. As operações matemáticas requeridas eram mínimas — apenas somas e multiplicações —, mas a quantidade de informação processada, enorme.

O LEO (Lyons Electronic Office) foi, portanto, projetado de acordo com essas necessidades, mas só se tornou operacional em 9 de fevereiro de 1954, quando calculou a folha de pagamento dos 1.700 membros da equipe. Ele realizava em 1,5 segundo o trabalho que anteriormente um funcionário levava 8 minutos para fazer.

O equipamento foi um grande sucesso para a Lyons, cuja direção logo percebeu que uma só máquina seria insuficiente. O projeto despertou grande interesse no mercado e a Lyons acabou fundando uma empresa para aproveitar o *know-how* adquirido na fabricação e comercialização de computadores. A Leo Computers foi muito bem-sucedida e passou a produzir uma série de versões aperfeiçoadas do LEO. A empresa foi absorvida em 1963 pela English Electric Company.

# Conforto no trabalho

**Ergonomia é a ciência que torna mais agradável o trabalho com máquinas. No tocante aos computadores, as pesquisas estão voltadas para o vídeo e o teclado.**

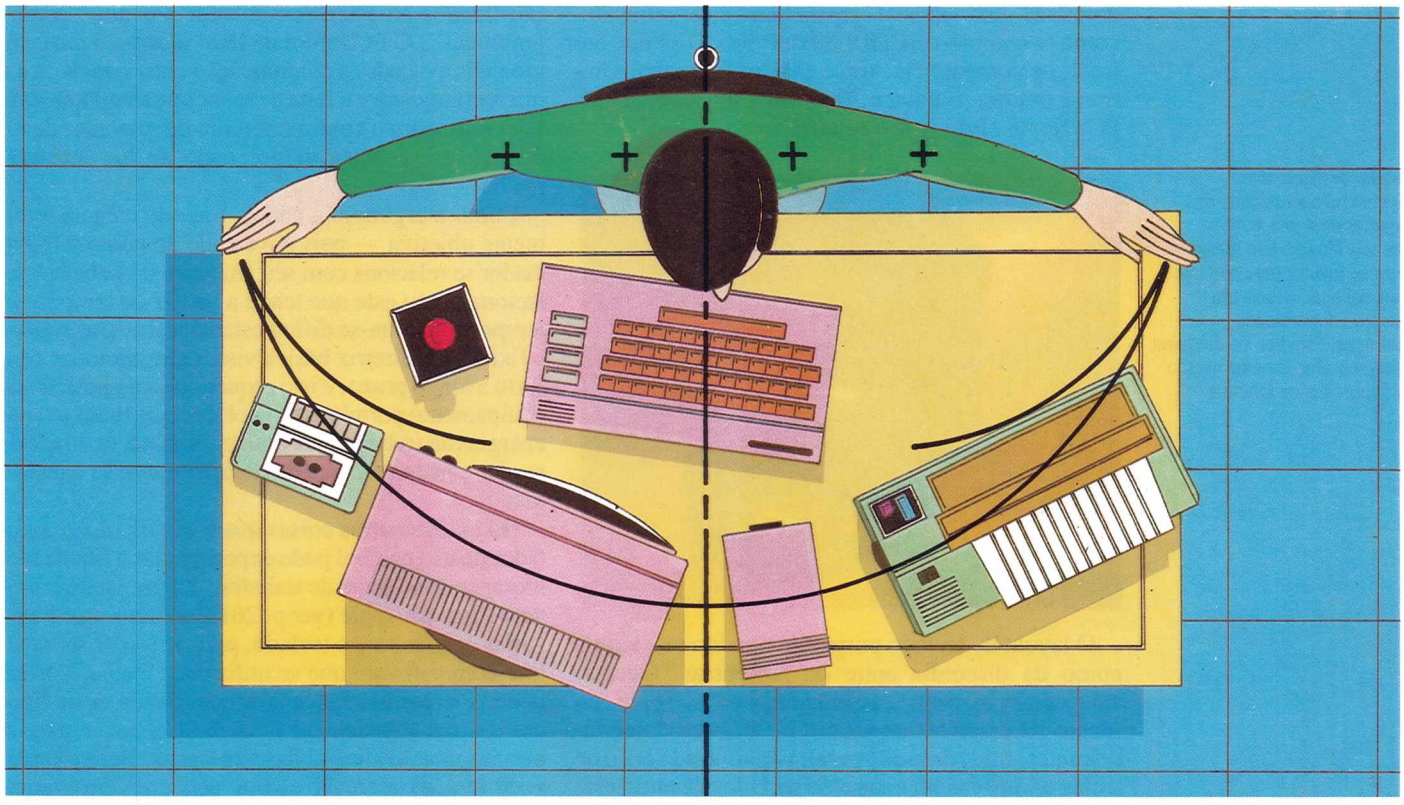
O desenho de uma peça deve ser analisado, levando-se em conta dois aspectos: o estético, ou seja, beleza na forma e na aparência, e o ergonômico, que trata do relacionamento do trabalhador com seu meio ambiente. Por melhor que alguma coisa funcione, não nos sentimos felizes ao usá-la se ela não tiver boa aparência; pela mesma razão, o ambiente de trabalho não deve ser confuso ou desconfortável.

Na compra de um microcomputador, o fator qualidade ergonômica provavelmente influirá menos que o preço e o desempenho da máquina. No entanto, convém analisar o ambiente físico em que o computador será utilizado. Antes de mais nada, o lugar em que você trabalha assemelha-se a um escritório, com espaço adequado na superfície da mesa e altura adequada? Ou, simplesmente, você liga seu micro ao televisor e trabalha com ele no colo ou, pior ainda, deitado no chão, diante da TV?

Programação de computadores é um processo complexo, e torna-se mais difícil ainda quando se trabalha num ambiente completamente inadequado. Há muitas maneiras de tornar o local de trabalho mais confortável. Primeiro, vejamos o que pode ser feito para facilitar a leitura da tela. Se você estiver usando um aparelho de televisão doméstico, não poderá tirar proveito das mais recentes inovações que ajudam a reduzir ou eliminar o brilho da tela nos monitores, entre as quais se incluem filtros para minimizar os reflexos e vídeos de fósforo, em geral verdes. Contudo, você pode melhorar a qualidade da imagem no aparelho de televisão, colocando um filtro diretamente sobre a tela. Além dos filtros coloridos comuns, que são fáceis de se obter, pode-se também usar um filtro polarizador, que elimina reflexos. Com estes meios, consegue-se alto contraste com baixo nível de brilho, evitando um esforço visual desnecessário.

O nível de iluminação externa também é muito importante. Trabalhando-se durante a noite, é aconselhável usar uma luminária baixa, que ilumine o teclado do micro e as anotações com que estiver trabalhando, mas que deixe a tela em relativa penumbra. A distância do corpo e dos olhos em relação ao vídeo também é importante — deve ser equivalente ao comprimento do braço. O monitor deve ser móvel, de modo que o plano da tela faça um ângulo de 90 graus com a linha que vai dos olhos ao centro da tela. Usando-se um aparelho de televisão, basta colocar um ou mais livros sob a parte dianteira, para obter o mesmo resultado. Entretanto, neste ponto pode surgir seu próprio reflexo na tela, que se elimina com o uso de um filtro de superfície fosca.

**Expressão corporal**
O corpo humano pode variar em forma e tamanho, mas suas proporções permanecem constantes, como é de conhecimento de qualquer estudante de desenho artístico. O estudo da ergonomia usa essa coerência para estabelecer regras gerais para o planejamento de ambientes de trabalho. No que diz respeito ao computador de uso pessoal estas regras sugerem que a tela deve ficar a uma distância equivalente a um braço (para minimizar as mudanças que ocorrem no foco ocular quando se olha alternadamente para perto e para longe, da tela para o papel e vice-versa). A posição do teclado também é estabelecida pelas mesmas regras.



## Problemas no teclado

Estando o monitor bem ajustado para ser usado, passemos à análise do layout e dos atributos físicos do teclado. Os fatores mais importantes são: a altura do teclado em relação à mesa de trabalho e o ângulo que as fileiras de teclas formam entre si. Em circunstâncias ideais, o teclado deve ficar a uma altura tal que permita que os pulsos e os antebraços do operador fiquem apoiados na mesa, à frente do teclado, cuja inclinação deve ser regulável. Infelizmente, poucos microcomputadores de uso pessoal são projetados com o perfil baixo exigido. Boa parte deles apresenta problemas nos teclados, pois usam teclas de membrana flexível ou de lâmina de borracha moldada, em lugar de teclas com molas. As teclas de membrana flexível não proporcionam "sensibilidade" alguma e, como no caso dos TK82 e TK83, estão espaçadas de tal forma que chega a ser um desafio para qualquer pessoa conseguir teclar. A combinação desses fatores transforma a entrada de programas longos em trabalho exaustivo. Alguns micros tentam contornar este problema emitindo um sinal audível sempre que uma tecla é pressionada o necessário para fazer contato, o que não chega a ser uma compensação suficiente. Há empresas que vendem teclados alternativos para tais computadores — no tamanho profissional e apresentando teclas com molas —, mas os exemplares bem projetados são caros. Eles também mantêm uma única chave de entrada, criada pela Sinclair para acelerar operações em BASIC, o que constitui motivo de constante irritação até para digitadores semi-especializados.

O layout ideal de um teclado requer que as fileiras de teclas, quando vistas de lado, estejam dispostas de modo a parecer que fazem parte da circunferência de um tambor. Isto deve minimizar o movimento direcional dos dedos do operador. Alguns computadores de uso pessoal preenchem este requisito, como os compatíveis TRS-80 (CP 500, Digitus e outros), os compatíveis Apple (Micro Engenho, Unitron e outros), o Itautec 7000, os compatíveis IBM-PC (Nexus, Ego, Link e outros).



**Máquina de taquigrafia**
Quando há necessidade de registrar uma fala, e o estenógrafo não tem meios de pedir ao locutor para que fale mais devagar, usa-se um aparelho denominado Palantype. Este tipo de máquina de taquigrafia usa uma versão estenográfica da fonética.

O layout do teclado propriamente dito tem sido o pomo da discórdia entre os projetistas. Quando apareceram as primeiras máquinas de escrever, no século XIX, havia tantos tipos de teclados como fabricantes, mas, de modo geral, o mais usado era o de teclas agrupadas no centro do teclado. Quando este tipo de máquina foi desenvolvido, na década de 1870, os fabricantes perceberam que os tipos colidiam muito uns com os outros, mesmo quando as datilógrafas eram bastante lentas. O problema ocorria com maior freqüência com palavras como, por exemplo, "ten", pois as letras mais usadas em inglês eram convenientemente colocadas próximas umas às outras e, quando batidas em rápida sucessão, entrechocavam-se. A solução adotada foi separar as letras que apareciam juntas com maior freqüência nas palavras — daí o teclado padrão QWERTY, projetado por Scholes e Gliden, nos Estados Unidos. Não há razão para que um teclado eletrônico tenha forçosamente esse layout, a não ser para continuar mantendo-se dentro dos padrões convencionais — um exemplo interessante de como um padrão global "de fato" pode tornar-se indesejável, e ainda assim continuar imutável.

## "Cordão umbilical"

Contudo, esforços vêm sendo realizados no sentido de desenvolver teclados alternativos. Em 1977, a sra. Lillian G. Malt, aproveitando a flexibilidade inerente ao equipamento eletrônico, produziu um teclado com formato ajustável à mão, bem menos consativo do que o modelo convencional; ele também é mais rápido para operar — em geral, são digitadas trezentas ou mais palavras por minuto. Infelizmente, esse teclado não conseguiu quebrar o tabu do QWERTY no layout dos teclados.

Uma característica útil que este teclado (chamado Maltron) tem em comum com muitos microcomputadores é a de ser destacável. A maioria dos computadores domésticos não têm monitores incorporados e são portáteis, mas o mesmo não acontece com muitos microcomputadores de uso comercial. Ultimamente, os teclados tornaram-se bem mais finos e são ligados ao microcomputador por um "cordão umbilical". O PC Junior da IBM já atingiu um estágio mais avaçado: a comunicação entre o teclado e o microcomputador é feita de maneira semelhante à da televisão, com o controle remoto do gravador de videocassete, e funciona por meio de luz infravermelha.

Como a ergonomia não se trata de ciência totalmente objetiva — pois é o estudo de como o trabalhador se relaciona com seu ambiente de trabalho, relacionamento este que tende a mudar de tempos em tempos —, torna-se difícil estabelecer regras rígidas e fixas. Sua diretriz básica visa a proporcionar conforto a longo prazo, e isto requer que a disposição do equipamento permita que você dedique todas as suas energias ao trabalho que realiza, sem necessidade de mudar constantemente de posição e sem se cansar à toa.

Há muitas outras coisas que o usuário do computador de uso pessoal pode experimentar a fim de melhorar seu ambiente de trabalho. Examinando o modelo Lisa, da Apple (ver p. 261), notamos que havia várias opções para o teclado, para o caso de se trabalhar com software que se utiliza de "menu". Você poderia experimentar uma versão barata deste tipo, usando um joystick ou track ball, e fazer para si mesmo uma avaliação das vantagens. Sem dúvida, você terá de escrever alguns pequenos programas




**Batendo nas teclas**
Antes da invenção do teclado eletrônico, cada tecla da máquina de escrever tinha de ser fisicamente conectada ao minúsculo tipo fundido em forma de caractere que, por sua vez, fazia a impressão sobre o papel. Isso impunha certas restrições ao formato do teclado, uma vez que era essencial manter separadas as teclas usadas com maior freqüência, para que as barras que portavam os caracteres não se entrechocassem. Embora isso não seja mais necessário, continuamos empregando o QWERTY.

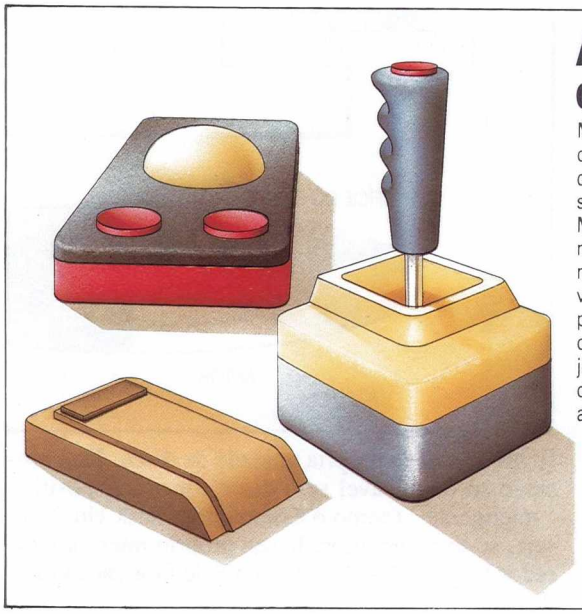
## Reduzindo o cansaço

Finalmente, se você tem jeito para marcenaria, poderia tentar construir um gabinete para a máquina, com o teclado embutido no tampo e o aparelho de televisão, ou monitor de vídeo, posto no ângulo mais conveniente.

Com freqüência as versões comerciais deste tipo de móvel, quando disponíveis no mercado, têm espaço para armazenar unidades de disco ou cassetes em quantidade, em prateleiras colocadas sob o tampo, permitindo que todo o material fique guardado. Basicamente, ergonomia não passa de bom senso aplicado, mas o fato de refletir um pouco sobre o assunto pode resultar numa redução significativa de dor nas costas e de esforço visual.

para serem usados durante a experiência, mas o emprego do PEEK e do POKE, nos confins da memória da tela, facilitará a tarefa.

Alternativamente, se o seu computador permite a reespecificação do valor de cada tecla, você pode modificar o arranjo do teclado, colando etiquetas sobre as teclas para indicar seus novos valores. Neste caso, talvez seja mais fácil, utilizando o comando PEEK, pegar o valor dos 8 bytes que formam um caractere num arranjo de oito variáveis, mudar os valores no arranjo e então colocá-los de volta, utilizando o comando POKE. Mediante o POKE, você pode colocar os 8 bytes que formam o caractere diretamente no espaço alocado para o caractere que você quer substituir, mas ao usar este método lembre-se de guardar o primeiro conjunto de valores em um arranjo temporário e só então mudar cada caractere para uma nova posição, na ordem. Guarde em fita cassete o programa desta operação, pois quando você desligar a máquina, ou após um "reset", o valor de cada caractere voltará ao original.



## Alternativas do futuro

Muitos designers de computador dispensariam completamente o teclado, se isso fosse possível. Modelos mais novos de micros, com memória de maior capacidade e maior velocidade de processamento, permitem o uso alternativo de outros dispositivos, tais como joystick, track ball e mouse, desde que o software seja apropriado.

# Registro de trilhas

**A função do Sistema Operacional de Discos (DOS, em inglês) é controlar o lugar em que as informações estão registradas nos discos. Sem o DOS, a programação seria uma tarefa árdua.**

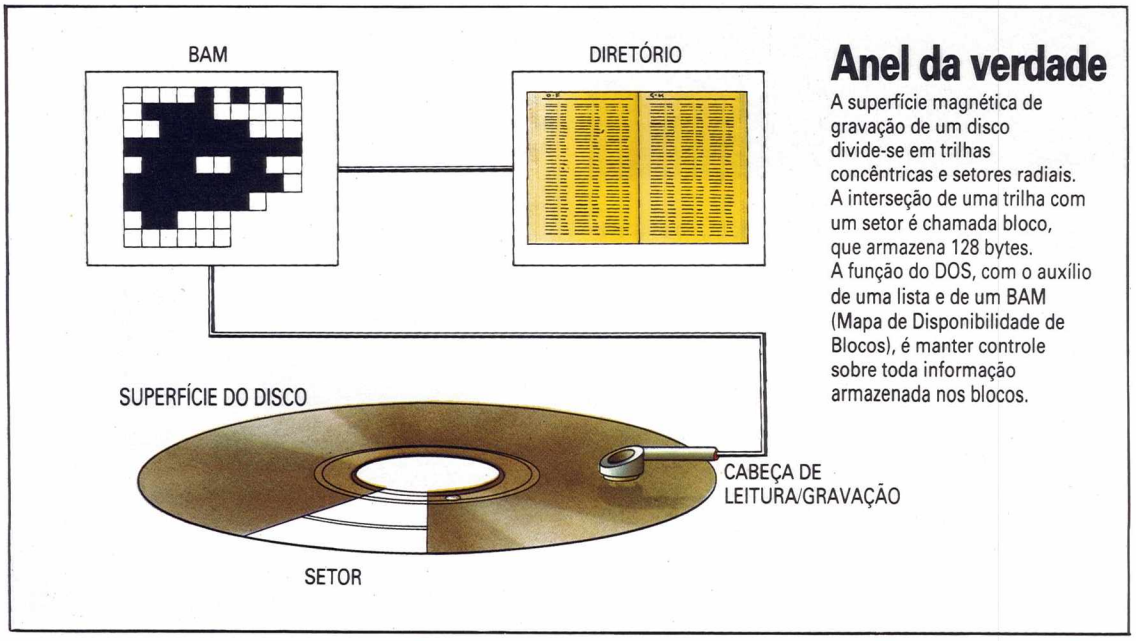
Antes que um computador possa executar qualquer tipo de programa aplicativo, precisa ter seu próprio conjunto de programas para dirigir as diversas partes do sistema e entender as instruções contidas no programa do usuário. Esse conjunto de programas internos é chamado Sistema Operacional, e na maioria dos microcomputadores fica permanentemente dentro do computador, em forma de memória ROM. Em geral não se percebe que o Sistema Operacional está funcionando, e por esta razão dizemos que ele é "transparente quando em operação".

Se o seu micro inclui uma unidade de disco, então uma grande parte do Sistema Operacional estará envolvida com as diversas operações relacionadas com os discos. Este conjunto de operações rotineiras é chamado Sistema Operacional de Discos (Disk Operating System, DOS). Estas três letras são encontradas nas marcas de muitos produtos do ramo — o sistema operacional da Microsoft, por exemplo, chama-se MSDOS. Um DOS pode vir de três formas diferentes. Primeiramente, ele pode fazer parte da ROM dentro do computador, como no micro inglês Sinclair Spectrum, que tem comandos para operar o microdrive embutido (que não consiste de fato num disco, mas é semelhante em operação).

Um outro tipo de sistema armazena o DOS em ROM dentro da própria unidade de disco. Este processo só é aplicável se o disco for um dispositivo "inteligente" (como o Commodore Disk Unit), ou seja, se tiver incorporado seu próprio microprocessador ROM e RAM. Este tipo é de fabricação mais dispendiosa, mas oferece vantagens consideráveis sobre unidades de disco "não inteligentes". Por exemplo, ele não ocupa a memória principal do computador e pode executar uma operação de disco complexa, enquanto o computador propriamente dito continua com o programa aplicativo.

Na terceira forma, o DOS fica dentro do computador em RAM. Esta técnica vem tendo grande aceitação nos sistemas de aplicação comercial, nos quais as unidades de disco vêm embutidas no computador, e há bastante RAM disponível (digamos, mais de 128 Kbytes, como medida padrão). Para o fabricante, este sistema oferece a vantagem de eliminar a necessidade de criar um conjunto totalmente novo de ROMs sempre que houver a mínima modificação no DOS, e para o usuário há a possibilidade de escolha entre uma variedade de marcas de sistemas operacionais que podem ser usados no mesmo hardware.

Ao ser ligado o sistema, imediatamente surge uma pergunta: como o DOS entra em primeiro lugar na RAM? O DOS tem de ser transferido do disco para a RAM, mas se não há DOS no computador para dizer-lhe como controlar o disco, como é que ele pode colocar algo na RAM? Um programa não pode colocar-se na RAM de forma espontânea; por isso, o computador tem de ter um programa minúsculo em ROM, executado sempre que a máquina é ligada. Este programa chamado "bootstrap" é uma forma muito simples de DOS. A tarefa do "bootstrap" é simplesmente achar o DOS principal no disco, transferindo-o, byte por byte, para a RAM, depois do que o DOS pode assumir a direção e de-




## Anel da verdade

A superfície magnética de gravação de um disco divide-se em trilhas concêntricas e setores radiais. A interseção de uma trilha com um setor é chamada bloco, que armazena 128 bytes. A função do DOS, com o auxílio de uma lista e de um BAM (Mapa de Disponibilidade de Blocos), é manter controle sobre toda informação armazenada nos blocos.

sempenhar funções bem mais sofisticadas. Este processo de ligar o computador e esperar o DOS assumir a direção chama-se "booting-up". Quando ele termina, aparece um aviso na tela que indica que o computador está pronto para receber ordens do usuário.

Qualquer que seja a forma tomada pelo DOS no sistema, sua função principal é controlar o lugar em que as informações estão contidas no disco. Lembre-se de que um disco (ver p. 114) é dividido em anéis concêntricos, chamados trilhas, que, por sua vez, são divididas em setores; a interseção de uma trilha com um setor denomina-se bloco. Um bloco contém 128 bytes de informação, e é a menor unidade que o disco pode ler ou gravar por vez. Uma das principais razões para a existência do DOS é que ele faz o computador lembrar o local exato de tudo o que está contido no disco, e esta tarefa é mais impressionante do que parece. Suponhamos que uma unidade de disco tenha uma capacidade de 320 Kbytes — suficiente para armazenar vinte programas de 16 Kbytes cada. Uma vez que cada bloco contém 128 bytes, para carregar um destes programas sem se utilizar do DOS seria necessário que você especificasse 128 blocos diferentes, cada um com sua própria trilha e seu próprio número de setor!

Para poder desempenhar esta função, o DOS possui um diretório (uma espécie de lista de endereços), normalmente localizado na trilha central do disco,

## Diretório local

| Arquivo | Tipo | Local (trilha-setor) |
|---|---|---|
| Invaders | Prog | 20-1,20-7,20-2... |
| Temperat | Prog | 25-11,26-5,26-12... |
| Budget | Prog | 23-12,24-3,24-9... |
| Budgetdat | Dados | 27-1,27-7,27-2... |

O diretório sempre ocupa a trilha central de um disco. Contém os nomes e os tipos de arquivos (programa, dados e talvez outras categorias), e números da trilha e do setor em que o arquivo está armazenado.

porque tendo de ser consultado com freqüência isto minimiza a distância que a cabeça de leitura/gravação precisa percorrer. A velocidade de operação de um disco depende mais do tempo despendido para mover a cabeça, de uma trilha para outra, do que da velocidade de rotação do disco.

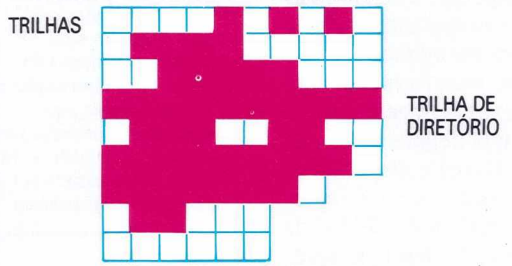
O diretório é uma lista que contém tudo o que está arquivado no disco (tanto programas quanto arquivos de dados), com detalhes como nome e tipo de arquivo e uma lista dos blocos (todos com especificação de trilha e setor) onde ele está armazenado. Pode haver outras entradas, tais como os dados de uma cópia back-up (de reserva) que acabou de ser feita, ou uma lista dos usuários que têm acesso a determinado arquivo.

Quando um novo arquivo é armazenado, o DOS deve consultar uma espécie de lista de endereços, chamada Block Availability Map (BAM) ou Free Sector List, que tem um bit correspondente a cada bloco do disco e, quando o bloco é usado, o valor de seu bit muda de 0 para 1.

Alguns computadores de uso pessoal com unidades de disco possuem um programa utilitário que apresenta o BAM na tela, e pode-se ver as entradas sendo feitas enquanto você grava um programa. Quando um arquivo é apagado, o DOS não se ocupa em limpar todos os blocos usados — ele muda as entradas no BAM, para indicar que o conteúdo daqueles blocos passou a ser desnecessário.

Outro aspecto deste sistema é que os arquivos não

## Espaço de sobra




Antes que o DOS possa armazenar um arquivo novo e fazer uma entrada no diretório, ele deve consultar o Block Availability Map (BAM) ou Free Sector List. Esta é uma parte da memória onde cada bit corresponde a um bloco do disco. O binário 1 indica que o bloco está ocupado e o 0 indica que está livre (representados por quadrados cheios e vazios). Note-se que as trilhas que ficam mais próximas do centro do disco por serem mais curtas, têm menos setores.

são armazenados em blocos vizinhos consecutivos, como se poderia supor. Admitamos, por exemplo, que uma trilha consista em doze setores, numerados de 1 a 12, no sentido horário. O primeiro conjunto de 128 bytes de um programa encontra-se no setor 1; o segundo, no setor 7; o terceiro, no setor 2, e assim por diante. Isso acontece porque, enquanto o conteúdo de um bloco é transferido para o buffer (memória intermediária), usado para gravar o bloco, transcorre um pequeno lapso de tempo. Se o DOS precisasse gravar setores consecutivos, teria de esperar uma rotação completa do disco entre cada gravação, o que retardaria o sistema. Além do mais, um




**Teste de QI**
Algumas unidades de disco contêm seu próprio microprocessador e sua RAM. Estas são chamadas unidades "inteligentes", onde o DOS está incorporado em forma de ROM. Quando são usadas unidades "não inteligentes", o DOS fica armazenado dentro do computador.

disco que tem certo tempo de uso, com arquivos que mudam de tamanho todo dia, acaba com um BAM parecendo um pedaço de queijo suíço, e assim os novos arquivos têm de ser encaixados nos buracos.

Um Sistema Operacional de Discos possui muitas outras funções, inclusive a de formatar novos discos (marcar trilhas e setores em discos virgens e criar diretórios vazios), fazendo cópias back-up e "arrumando" discos cheios. Versões mais sofisticadas incluem uma variedade de estruturas de manipulação de dados (ver p. 204).

# Expansão dos limites

## O micro inglês Sinclair ZX81 ainda é o mais barato, porém, com acessórios adequados, pode tornar-se uma máquina sofisticada.

Entre todos os microcomputadores existentes no mercado, o inglês ZX81, da Sinclair, é o que mais vantagens oferece em troca de seu dinheiro, mesmo no modelo standard; mas, com os inúmeros acessórios que existem à venda, se transforma em um sistema de microcomputador extremamente sofisticado. Entre eles, incluem-se: tela de alta resolução gráfica em cores, sintetizador de voz e dispositivos para comunicação. No Brasil, este modelo é representado pela família TK da Microdigital, CP 200 da Prológica e outros. Alguns dos opcionais aqui apresentados já podem ser encontrados no mercado brasileiro, ou estarão disponíveis muito em breve.




**Cartucho RAM**
O modelo standard do ZX81 tem apenas 1 Kbyte de RAM, do qual 123 bytes são utilizados por variáveis do sistema. A memória adicional da própria Sinclair, aqui apresentada, é comercializada em apenas um modelo — 16 Kbytes —, mas existem outras opções, tais como a versão Cheetah, que oferece até 64 Kbytes

**Outras novidades**
Além das unidades aqui mostradas, existem outros dispositivos para melhorar o desempenho do ZX81. Um cartão colorido, por exemplo, fornece até 16 cores para a imagem de TV, e um gerador de som permite a programação de três "vozes". Portas bidirecionais admitem a ligação de até 16 dispositivos de input/output ao mesmo tempo. Longe de ser um pequeno computador, o ZX81 é um equipamento que tem possibilidade de ser expandido para usar o pleno potencial de seu microprocessador Z80.

**Acopladores acústicos**
O modulador/demodulador é apresentado em duas formas: modems de conexão direta, que requerem um conector de plugue adicional para serem ligados ao sistema telefônico, e os acopladores acústicos, como o Micro-Myte 60, visto aqui, que utiliza o próprio fone.
Modems de conexão direta, que geralmente são mais caros, geram e reconhecem sinais eletrônicos que representam os dígitos 0 e 1 na informação que está sendo recebida ou transmitida. Acopladores acústicos, que podem funcionar com baterias, traduzem os dígitos 0 e 1 em sons audíveis, para transmissão pela rede telefônica

**ROM com linguagem FORTH**
Os microcomputadores ZX da Sinclair usam uma versão particular do BASIC e, já que não é possível instalar um dialeto diferente, pode-se mudar completamente a linguagem, para FORTH, por exemplo. Há duas maneiras de fazer isso: na primeira carrega-se o computador com a nova linguagem em RAM, a partir de uma fita cassete; isso significa que o computador voltará à linguagem BASIC toda vez que for ligado, ou após um reset, ou quando a ROM do BASIC for trocada por outra. Esta linguagem FORTH embutida em ROM, de David Husband, supera a maioria das outras, permitindo que dez ou mais programas sejam executados ao mesmo tempo pelo computador




**Sintetizador de voz**
Outro acessório interessante da Cheetah é o sintetizador de voz Sweet Talker: utiliza um sistema alofônico, bem mais fácil de programar do que as unidades que trabalham com fonemas (alófonos são grupos de fonemas de som semelhante). Há outras unidades similares no mercado para o ZX81 e para muitos outros microcomputadores de uso pessoal

**Hebot**
A tartaruga Hebot, encontrável já montada ou em forma de kit, é um dos mais sofisticados robôs que andam no chão. Ela vem equipada com o software necessário para fazê-la funcionar, mas, além disso, há uma variedade de acessórios, como os fotossensores, que podem ser usados em conjunto com uma fita refletora fixa no chão, para fazer com que o robô siga um caminho predeterminado

**Teclados**
O teclado tipo membrana flexível talvez seja a característica menos satisfatória do ZX81; não é de admirar que outra empresas ofereçam teclados alternativos em tamanho normal, com teclas convencionais de molas. O teclado Mapsoft ZX81, visto aqui, é fabricado pela Maplin Electronics.

Além do conjunto normal de caracteres, o Mapsoft apresenta três teclas a mais. É um acessório muito útil para um ZX81, embora haja produtos similares que custem mais que o dobro. Outro tipo de teclado é o que se encaixa sobre as teclas do ZX81, usado juntamente com o original — ele facilita bastante a localização de uma determinada tecla

**Joysticks**
Como muitos dos aparelhos ZX81 são usados com videojogos, chega a ser estranho que a Sinclair não tenha produzido seus próprios joysticks e controles. Entretanto, existe uma grande variedade deles, que tanto podem ser não-programáveis, e neste caso especificam que movimentos deverão ser feitos com o joystick, como programáveis, quando o usuário decide quais serão os movimentos. Para programar o modelo da AGF Hardware visto aqui, deve-se girar os cabos de conexão. Outros são programados pelo computador e aceitam qualquer joystick e um track ball

**Impressora ZX**
A impressora ZX da Sinclair utiliza papel aluminizado, que é sensível à eletricidade. Ao invés de imprimir da maneira convencional, a cabeça de impressão remove a camada aluminizada, revelando uma superfície mais escura. Ainda que seja razoavelmente rápida na operação, o tipo e a largura especiais do papel constituem um problema. Contudo, pode-se usar uma impressora normal, com um cartão de interface. Os cartões servem para as interfaces RS232 e Centronics

# A toda velocidade

**Dando atenção cuidadosa às variáveis e à estrutura, você pode acelerar a execução de quase todos os programas em BASIC.**

O BASIC é, apesar do que dizem os críticos, uma linguagem versátil e um recurso pedagógico muito eficiente. Você pode desenvolver qualquer programa em BASIC, desde que seu equipamento tenha memória suficiente e o tempo de operação não seja fundamental. Entretanto, uma vez que o BASIC é interpretado e não compilado (ver p. 184), pode ser penosamente lento na execução dos programas, em especial aqueles que exigem uma mesma instrução traduzida e executada repetidamente.

A ordenação, por exemplo, é um processo altamente repetitivo: o procedimento se realiza em um loop, com loops menores alojados no interior do loop principal (ver p. 286). Se tivermos 100 itens a serem ordenados, o programa pode ter de fazer de 2.500 a 5.000 interações do loop. Uma ordenação em BASIC sempre será lenta, mas o modo como o programa for escrito poderá resultar em diferença significativa na velocidade de execução. Se uma instrução deve ser repetida 5.000 vezes e se durante a codificação pudermos economizar um centésimo de segundo do tempo de execução em cada repetição, haverá economia total de 50 segundos — uma melhora considerável para o usuário.

Para perceber a diferença entre a boa e a má codificação, você necessitará de um cronômetro e de um programa "banco de teste". Se possuir um Commodore, você poderá utilizar o próprio relógio do equipamento, com as variáveis correspondentes, TI$ e TI, como parte do programa banco de teste. Se seu computador não tiver relógio, você terá de utilizar um cronômetro para medir o tempo do código em execução. Também é uma idéia interessante fazer seu programa indicar o início e o encerramento da execução, através de um bip, de modo que você possa saber quando está em operação.

O programa banco de teste pode ser o seguinte:

```
1000 L=500
2000 PRINT "**INICIO**": REM "BIP"
       instruções aqui
2100 TI$="000000"
2200 FOR K=1 TO L
.....
2900 NEXT K:T9=TI
2950 REM "BIP" instruções aqui
3000 PRINT "******STOP******"
3100 PRINT "Levou ";(T9/60)," segundos"
```

As linhas 2100 e 3100 foram elaboradas para os usuários do Commodore. Para outros equipamentos, deve-se eliminá-las ou substituí-las, de acordo com o código correspondente. No espaço entre as linhas 2200 e 2900 colocaremos o código a ser cronometrado. Observe que as cronometragens serão referidas às repetições L, onde L representa o limite do loop. O teste da execução de apenas uma unidade de código será de pouca precisão, pois o relógio do equipamento mede unicamente em sexagésimos de segundo e há uma cronometragem superior também imposta pelo código do programa de banco de teste.

Aqui estão algumas regras gerais para desenvolvimento de uma linguagem BASIC eficiente, apresentadas em ordem aproximada de importância:

### 1. Evitar todos os procedimentos aritméticos no interior dos loops.

A potenciação ($x^3$, significando "x elevado à terceira potência", por exemplo) e as funções matemáticas (cos (y), significa "o co-seno do ângulo y", por exemplo) são operações particularmente lentas. A multiplicação e a divisão são procedimentos mais morosos que a adição e a subtração; porém, mesmo a mais rápida dessas operações (a adição) é relativamente lenta.

Introduza as seguintes linhas no programa banco de teste:

```
900 Z=1.1
2300 X=Z↑3
```

e efetue o processamento. Em nosso equipamento de teste, 500 repetições levaram 27,95 segundos. Agora substitua a linha 2300 por:

```
2300 X=Z*Z*Z
```

e realize o processamento. Levou 3,55 segundos — uma diferença sensível!

O exame complementar revelará o nível da potenciação em que se torna conveniente substituir a multiplicação repetida pela função de potenciação. Em nosso computador, isso se deu na 18.ª potência (quando X=Z↑18). Todavia, lembre-se de que para calcular $Z^{2.3}$, por exemplo, a multiplicação repetida será inútil, enquanto a função de potenciação (↑) operar com qualquer número real, inclusive os negativos.

Utilize o programa banco de teste para verificar a duração dos outros processos aritméticos e compare as alternativas. Por exemplo, o que é mais rápido: dividir um número por 2 ou multiplicá-lo por 0,5?

### 2. Utilize variáveis em vez de constantes numéricas.

Toda vez que uma constante numérica (7.280, por exemplo) aparece em uma instrução em BASIC, gasta-se tempo com a tradução do número para forma utilizável. Experimente esta linha:

```
2300 X=X+7280
```

Em nossa máquina levou 4,63 segundos para executar 500 repetições, enquanto:

```
900 C=7280
2300 X=X+C
```

levou 2,75 segundos para realizar o mesmo número de repetições.

3. Se você tiver de utilizar a instrução GOTO, procure fazê-lo saltando para a frente e não para trás no programa. Entretanto, se tiver de retroceder, salte para o início do programa, ao invés de algumas linhas para trás.

O mesmo se dá com as instrução GOSUB. Ao encontrar uma instrução GOTO ou GOSUB, o interpretador BASIC compara o número da linha meta com o número da linha em operação no momento. Se a meta for maior que a linha em operação, o interpretador simplesmente avança buscando-a, linha por linha, até ser encontrada. Porém, se a meta for menor que a linha em operação, a busca sempre se iniciará a partir da primeira linha do programa. Isso significa que poderá ser um procedimento mais eficiente colocar sub-rotinas e unidades freqüentemente utilizadas em uma das extremidades do programa. Acrescente 56 linhas REM no início do programa para dar-lhe a extensão comumente usada e experimente:

```
2300 GOTO 2400
2400 GOTO 2500
2500 GOTO 2900
```

Levou 2,33 segundos para 500 repetições, enquanto:

```
2300 GOTO 2500
2400 GOTO 2900
2500 GOTO 2400
```

levou 4,85 segundos.

4. Inicialize todas as variáveis na ordem da freqüência de acesso.

Os nomes de variáveis são armazenados pelo interpretador, em uma tabela de símbolos, na ordem em que aparecem no programa pela primeira vez. Quanto mais tarde uma variável aparecer na tabela, mais tempo será gasto para encontrá-la e para acessar seus conteúdos. Pela mesma razão, você deve evitar o emprego de uma nova variável se puder recorrer a uma já anteriormente utilizada pelo programa, porém não em uso no momento.

Se uma variável for utilizada no interior de loops alojados — como acontece com freqüência em programas de ordenação —, essa variável será acessada constantemente; assim, faça sua inicialização no começo do programa, antes de qualquer outra variável, com um valor fictício, se necessário:

```
1000 L=500:C=7280:X=0:Z=1.1
2300 A=0
```

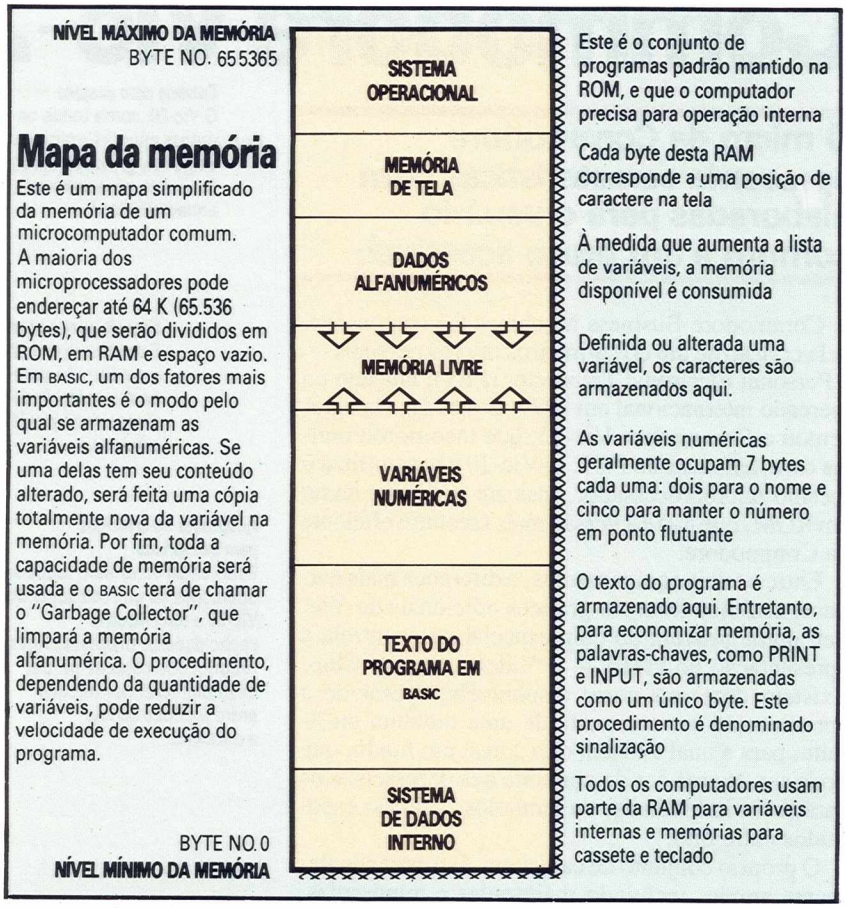
Levou 2,2 segundos para 500 repetições, enquanto:

```
1000 A=0:L=500:C=7280:X=0:Z=1.1
2300 A=0
```

levou 2,06 segundos.

5. Evite o emprego de variáveis alfanuméricas.

Operações com variáveis alfanuméricas utilizam a memória de modo aritmético diferente e um sistema de programação chamado "Garbage Collector" (coletor de lixo) pode ter de ser chamado ocasionalmente pelo interpretador para organizar os conteúdos de memória alfanumérica. Esse procedimento pode exigir muito tempo.




Uma demonstração genérica desse processo é difícil de ser escrita porque os computadores variam muito em seu sistema de gerenciamento de memória: você terá de preencher a maior parte da memória disponível para o usuário com dados — uma grande matriz numérica fará isto — e então executar manipulações com variáveis alfanuméricas que façam com que o Garbage Collector seja requisitado. Fornecemos a nossa máquina os seguintes dados:

```
40 POKE 52,32:POKE 56,32:CLR
```

para reduzir bastante o total de memória disponível para os programas em BASIC e então fornecemos:

```
1000 L=500:DIMT$(L)
1100 FOR K=1 TO L
1200 T$(K)="A"+"B"
1300 PRINT K
1400 NEXT K
```

que utiliza grande quantidade de memória alfanumérica e fornece uma matriz alfanumérica para uso posterior. As instruções PRINT são executadas em cada interação, apresentando o valor do contador do loop. Quando processamos essa versão do banco de teste, a impressão interrompeu-se repetidamente para chamada do Garbage Collector a fim de reordenar a memória. Algumas vezes a interrupção demorou mais de três segundos. O programa continua:

```
2300 A$=LEFT$(T$(L),1):B$=A$+RIGHT$(T$(L),1)
```

Isso exigiu um total de 30,03 segundos em 500 repetições. Quando processamos o mesmo programa com muito mais memória disponível, a garbage collection não foi percebida em operação e o loop cronometrado levou 8,66 segundos.

# Commodore Vic-20

## O micro da Commodore apresenta características bem elaboradas para o usuário comum e um preço acessível.

A Commodore Business Machines foi responsável pela criação de um dos primeiros micros pessoais — o Personal Electronic Transactor (PET), lançado no mercado internacional em 1977. Em 1981, ela apresentou o Commodore Vic-20, que incorporou muitas características do PET. O Vic-20 não só utiliza o mesmo microprocessador, mas até o mesmo BASIC em ROM, que não é a versão mais recente e eficiente da Commodore.

Entre os dois equipamentos, a diferença mais evidente está nos recursos gráficos adicionais do Vic. Seu nome provém do chip especial que controla a apresentação no vídeo — o Video Interface Chip. Existem dezesseis cores disponíveis, apesar de a apresentação ser composta de uma moldura ou limite, para a qual existem oito cores; um fundo, que pode ser de uma cor do conjunto das dezesseis; e os caracteres individuais, ou símbolos, que são escolhidos entre oito.

O próprio conjunto de caracteres é surpreendentemente amplo, incluindo maiúsculas e minúsculas, bem como dois conjuntos de caracteres gráficos de 62 teclas, além de quatro teclas especiais que podem ser utilizadas — com a tecla de mudança de operações acionada ou não — para proporcionar oito funções programáveis. A estrutura do teclado é bem adequada, tanto ergonômica quanto tecnicamente.

O principal inconveniente do Vic-20 é a pequena capacidade de memória — apenas 5 Kbytes, reduzidos a 3,5 Kbytes, após serem carregados na RAM o sistema operacional, o controle de vídeo e outras necessidades internas. Entretanto, pode-se endereçar até 32 Kbytes de memória e há disponibilidade de uma memória adicional, comercializada por diferentes fornecedores.

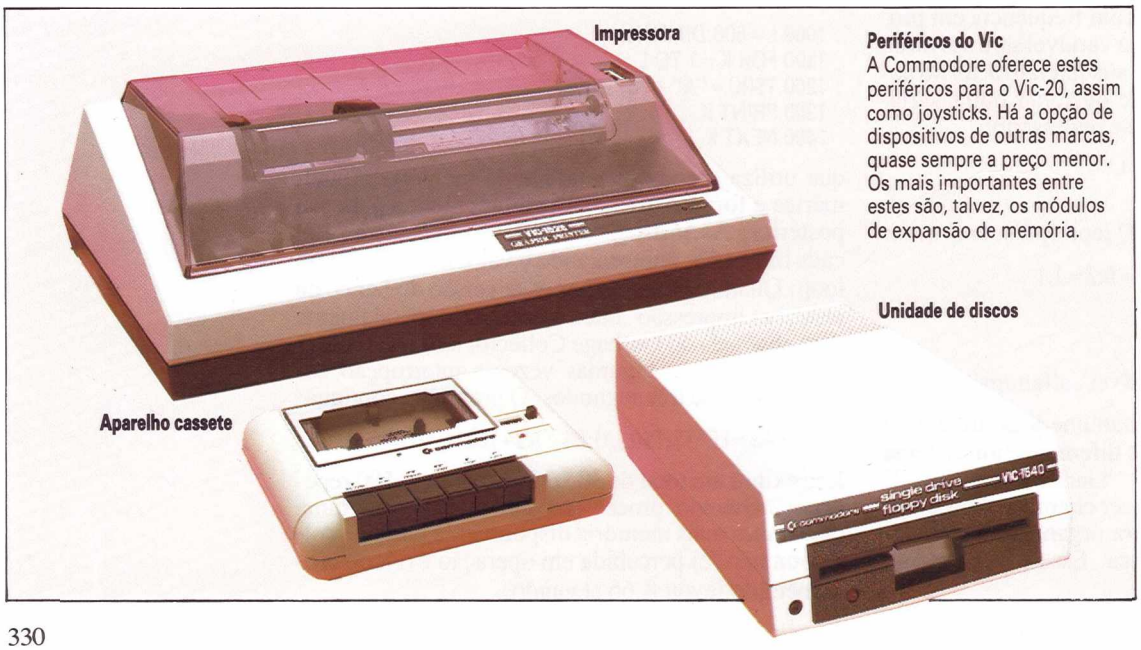
Possui entradas de interface para paddles/joysticks, canetas ópticas, cartuchos de jogos/expansão de memória, periféricos de impressão/discos, cassete, televisão, e há uma que se enquadra no padrão serial RS232 e pode ser usada com um modem ou impressora, não da Commodore. Além disso, existe grande variedade de acessórios para hardware, do mesmo modo que para o Commodore 64, lançado no mercado internacional mais recentemente (ver p. 189).

**Entrada para cassete**
O Vic-20, como todos os outros micros Commodore, exige um gravador cassete fabricado especialmente, que é conectado aqui

**Entrada para o usuário**
Este conector de 24 pinos é uma entrada serial, utilizada para o controle de vários dispositivos periféricos adicionais

**Adaptador de interface para periféricos**
Estes chips controlam todas as operações de entrada/saída do Vic-20 e têm alguma capacidade de processamento próprio. São capazes de, por exemplo, fazer conversões entre padrões seriais e paralelos

**Conector do teclado**
Conecta-se aqui o teclado ao adaptador de interface para periféricos



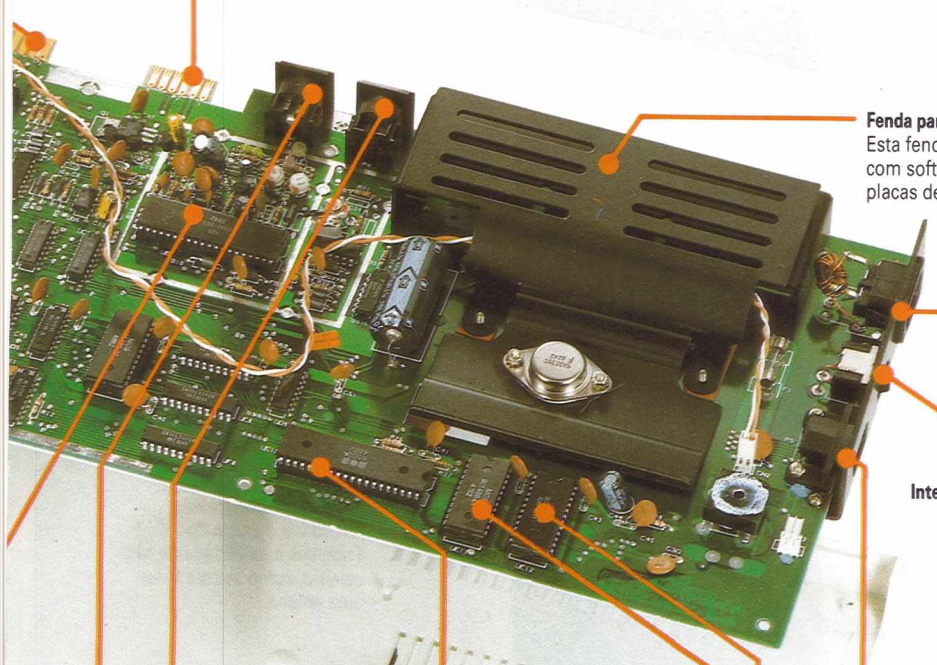

**Periféricos do Vic**
A Commodore oferece estes periféricos para o Vic-20, assim como joysticks. Há a opção de dispositivos de outras marcas, quase sempre a preço menor. Os mais importantes entre estes são, talvez, os módulos de expansão de memória.

**Chip de interface de vídeo**
Usa-se este dispositivo projetado para controle da apresentação no vídeo e do gerador de som a três vozes do Vic-20

**Memória de acesso aleatório (RAM)**
O Vic-20 é provido de 5 K de RAM, apesar de ser expansível externamente para 32 K



**Teclado**
O teclado de 62 teclas proporciona quatro conjuntos completos de caracteres e permite o uso de oito funções programadas pelas teclas especiais no lado direito



**Fenda para cartuchos**
Esta fenda comporta cartuchos com software em ROM, ou placas de memória

**Tomada de alimentação de energia**
Liga-se a alimentação de energia externa nesta tomada

**Interruptor de ligar/desligar**

**Entrada para jogos**
Pode-se conectar canetas ópticas, joysticks ou paddles neste lugar

**Tomada de áudio/vídeo**
Liga-se um modulador RF externo neste ponto

**Porta serial**
A maioria dos periféricos do Vic é controlada por esta entrada especialmente projetada que, ao contrário de muitas interfaces seriais, pode ser ligada a mais de um periférico por vez

**Microprocessador**
O Vic-20 emprega o microprocessador popular e muito testado 6502, de tecnologia MOS

**ROM (Read Only Memory)**
Estes chips incluem o interpretador BASIC, o conjunto de caracteres e outras funções necessárias para interpretar e executar programas

## COMMODORE VIC-20

**CLOCK**

1 MHz

**MEMÓRIA**

A máquina possui um padrão de memória de 5 K, podendo atingir 32 K por meio de uma expansão de 3 K, uma de 8 K e uma de 16 K.

**VÍDEO**

São 23 linhas de 22 caracteres. Gráficos de alta resolução, proporcionando 184 × 176 pixels. Dispõe de um máximo de 16 cores.

**TECLADO**

Teclado tipo máquina de escrever de tamanho normal.

**LINGUAGEM DISPONÍVEL**

BASIC

**OUTRAS LINGUAGENS DISPONÍVEIS**

Assembler e comandos adicionais de BASIC.

**PERIFÉRICOS**

Duas entradas seriais, conector para áudio/vídeo, entrada para cassete, fenda para cartuchos, entrada para jogos.

**DOCUMENTAÇÃO**

Uma das áreas em que a CBM falha é na de documentação de seus micros domésticos. Sistemas contábeis, como o 8032, são acompanhados do excelente "Guide to CBM Computing" (Guia para Computação CBM), de Adam Osborne, mas os proprietários dos Vic são relegados a um plano mais rudimentar. Mesmo escrito de modo simples e fácil de compreender, o manual é superficial no que se refere aos recursos da máquina. Felizmente, há a opção de outras publicações no mercado.

# Bastões ligados

**Dois novos tipos de joystick parecem não ter partes móveis. Um deles utiliza interruptores operados por meio de mercúrio, o outro capta sinais do corpo.**

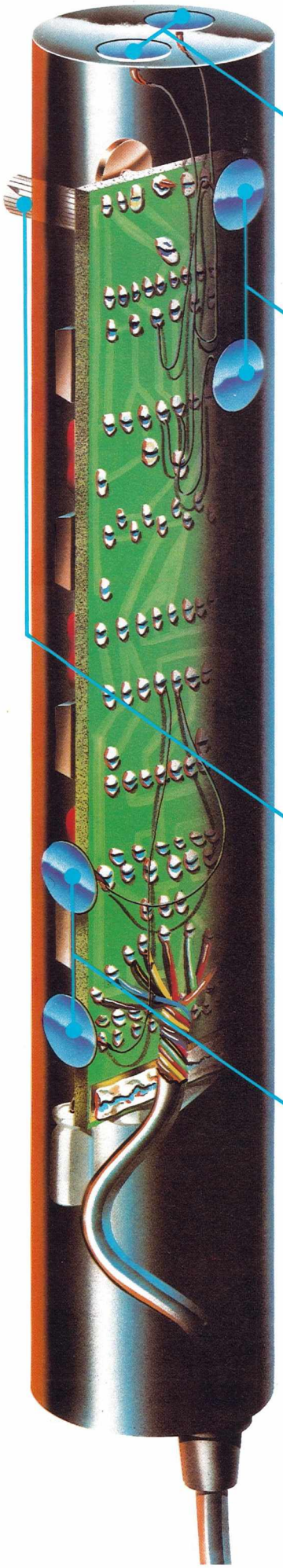
A indústria de microcomputadores se caracteriza por progressos tecnológicos rápidos e essas mudanças não estão limitadas aos computadores — os equipamentos acessórios e os periféricos têm tido também acelerados aperfeiçoamentos. Por exemplo, no curto espaço de tempo desde que tratamos pela primeira vez do funcionamento do joystick (ver p. 56), dois tipos completamente novos entraram no mercado inglês. Os joysticks agora desenvolvidos romperam quase que de forma absoluta com o sistema mecânico convencional descrito naquele artigo.

Um dispositivo chamado Le Stik foi o primeiro joystick analógico que superou os mecanismos habituais de sinalização. O Le Stik consiste em uma alavanca coberta, equipada com um botão acionador na extremidade e um controle de pausa montado lateralmente. Ao contrário de muitos outros dispositivos montados sobre unidades de base, o joystick é simplesmente mantido no ar e desviado da posição vertical para a posição requerida, movendo de modo apropriado a imagem em questão.



**Controle manual**
O Trickstick depende da "vibração de força", que é a radiação eletromagnética emitida pela força anelar, dispersa por todo o recinto. O corpo humano funciona como antena para as "vibrações de força" e os sensores do bastão captam os diferentes níveis de vibração, conforme a pressão feita pelos dedos.

## Trickstick



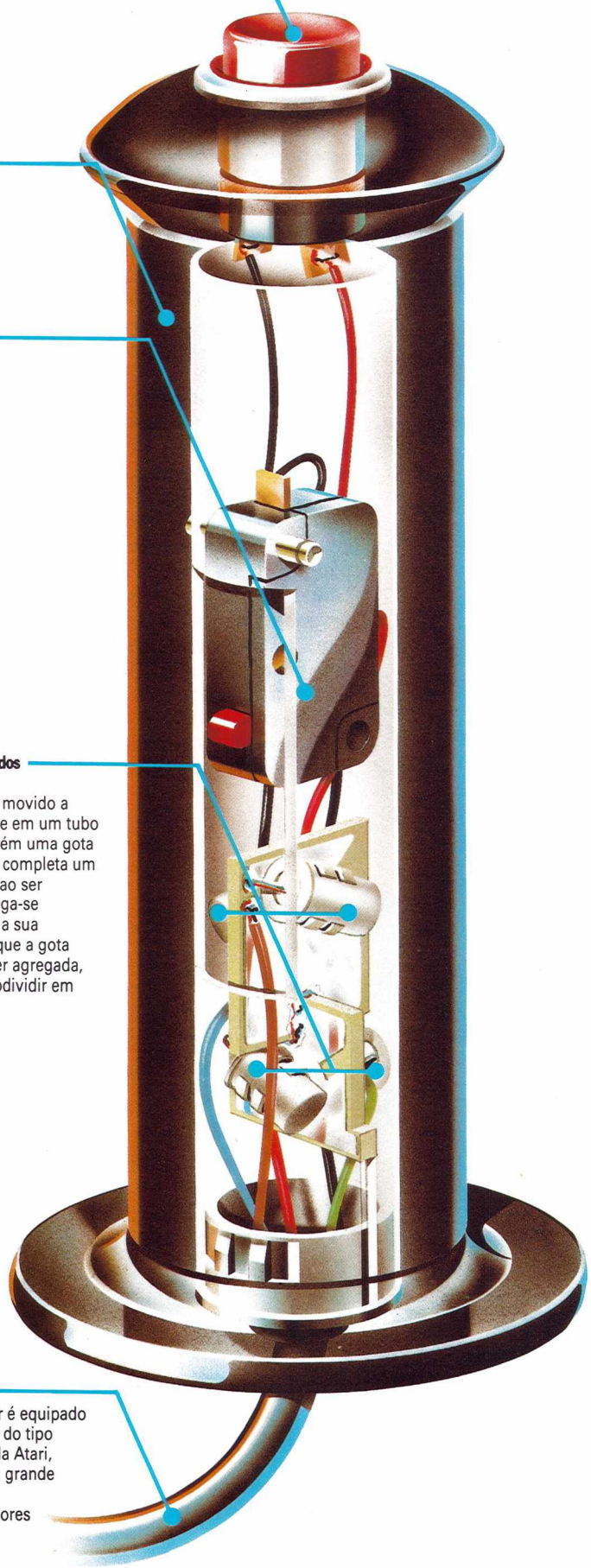
**Controle de movimento horizontal**
Pelo deslocamento do polegar entre essas duas teclas, pode-se controlar o movimento para diante e para trás

**Controles de movimento vertical**
A tecla superior controla o movimento para cima e a inferior para baixo

**Controle por sensibilidade**
Possibilita o ajuste do Trickstick à habilidade de cada usuário, como se este fosse uma antena

**Botões de detonação**
Cada um deles gera um sinal independente: pode-se lançar bombas através do inferior e detonar o canhão de laser com o outro, ao mesmo tempo

# Le Stick




O mecanismo no interior do Le Stik consiste em quatro tubos selados, preenchidos com mercúrio. À medida que o joystick se afasta da posição vertical, o mercúrio flui na direção escolhida e faz um ou mais contatos elétricos, como se um interruptor se fechasse. O retorno da alavanca para a posição vertical permite ao mercúrio escorrer de volta aos tubos, interrompendo o contato. A resposta do sistema é muito melhor que a dos joysticks anteriores. Na verdade, às vezes ele é excessivamente sensível, em especial se o jogo em questão for desenvolvido para ser usado com tipos convencionais de joystick.

O mais recente recurso de conversão de movimentos manuais em sinais que o computador pode identificar é utilizado pelo Trickstick da East London Robotics. Este joystick é único no efeito elétrico que emprega: utiliza o corpo humano como antena para captar a "vibração de força" (a radiação eletromagnética inofensiva emitida pela força anelar em qualquer recinto). O Trickstick consiste em um tubo vedado dentro de uma proteção plástica, que é mantido verticalmente em ambas as mãos. Há três pares de teclas sensíveis ao toque na superfície do tubo: um par controla o movimento para a frente e para trás; outro, o movimento para cima e para baixo; e o último corresponde aos botões detonadores.

A "vibração de força" que o corpo humano capta é transmitida através dessas teclas ao toque e a um sistema de circuitos no qual os impulsos são convertidos em sinais que fornecem ao computador dados direcionais. O sinal também pode indicar a distância que o movimento deve atingir. Quanto mais fortemente a tecla for pressionada, mais forte será o sinal e com maior rapidez será realizado o fornecimento de dados ao computador. Desse modo, o Trickstick combina o controle proporcional do joystick analógico com o controle digital rápido e direto das unidades que operam por interruptor. Uma vez que as pessoas causarão efeitos diferentes nos circuitos, o Trickstick deverá ser adaptado à sensibilidade de cada um. Isto é realizado por meio de um botão montado em uma das extremidades do dispositivo.

A idéia é certamente intrigante e os fabricantes se empenharam em obter uma patente da técnica. Entretanto, a confiabilidade e o desempenho do dispositivo ainda não foram comprovados.

# O som ideal

## O BASIC do Commodore 64 não combina com suas notáveis facilidades de som.

Entre os modelos existentes de microcomputadores, o Commodore 64 apresenta as mais sofisticadas facilidades para produção de som. Elas são atribuídas a um chip especial chamado Sound Interface Device — ou SID, como é mais conhecido.

A capacidade do SID é semelhante à do sintetizador monofônico comercial. Ele possui três osciladores com um alcance de oito oitavas (0-3.900 Hz em 65.536 etapas); um controle de volume principal que vai de 0 a 15; quatro formas de onda para cada oscilador (triângulo, serra, pulso variável e ruído); sincronização do oscilador; e geradores de envelope permitindo um controle ADSR para cada oscilador. Outras características são: modulação em anel; filtro programável com baixa passagem, passagem de sintonia, alta passagem, saída de entalhe (que bloqueia uma faixa estreita de freqüência) e ressonância variável; filtragem de envelope; duas interfaces para potenciômetro analógico-digital que podem ser usadas para controlar as facilidades SID; e uma entrada de áudio externo, permitindo que chips SID adicionais sejam ligados entre si. Outros sinais de áudio podem ser introduzidos, filtrados e misturados com a saída comum do SID.

Seria impossível detalhar aqui a operação de cada uma dessas funções (existem vários bons livros disponíveis), mas podemos explicar o significado de todas essas expressões. Antes de tudo, a sincronização do oscilador faz com que dois sinais separados (neste caso, duas vozes específicas) sejam reunidos harmonicamente, formando um único tom, mais complexo.

Modulação é a modificação de um sinal através de outro, afetando a freqüência ou a amplitude (volume) do som. Modulação em anel é a modulação da amplitude de uma voz por meio de uma outra. O resultado é um tom claro mas que tem um efeito desafinado, dissonante e pode ser usado para produzir um som de campainha, semelhante ao de um tambor de aço. Estes sons são considerados sobretons discordantes.

Através dos filtros, é possível eliminar tipos específicos de freqüência de um sinal. Os diferentes tipos de filtragem possíveis de se fazer no Commodore 64 têm os efeitos sugeridos pelos seus nomes: filtros de baixa freqüência cortam freqüências acima de uma freqüência específica; filtros de passagem de faixa eliminam freqüências acima e abaixo de uma determinada "faixa" de freqüência; filtros de entalhe são o inverso dos filtros de passagem de faixa — eles cortam uma determinada faixa; os filtros de alta freqüência cortam freqüências mais baixas do que uma freqüência específica; e a ressonância variável pode ser aplicada a todos os filtros acima citados

# Luz-guia

## Gráficos tipo Player-Missile constituem um dos pontos fortes dos Atari.

### Gráficos Player-Missile

Gráficos do tipo Player-Missile ou "PM" fazem parte importante da capacidade gráfica do Atari. Eles são semelhantes em natureza aos gráficos sprite do Commodore 64, permitindo que o programador desenhe e controle até oito formas diferentes de alta resolução. Essas formas móveis operam à parte do cenário de fundo e podem ser programadas para se movimentar na frente ou atrás de quaisquer outras formas desenhadas na tela. Isto permite ao programador acrescentar uma terceira dimensão aos efeitos da tela. Os gráficos PM podem ser movimentados de modo uniforme na tela, em alta velocidade, e portanto são ideais para jogos rápidos do tipo fliperama. Eles também são usados para criar displays estáticos mais coloridos do que os elaborados com modos gráficos normais, uma vez que os objetos de PM podem ser coloridos independentemente uns dos outros e do cenário de fundo.

Assim como acontece com todos os gráficos sprite, o segredo das facilidades de gráficos PM está no hardware dedicado. Registros especiais são planejados para controlar movimento, cor e display de tela dos objetos de PM. Tudo que o programador deve fazer é colocar certos valores nesses registros para manipular os objetos. Em BASIC isso é feito usando-se o comando POKE. Assim que o número é colocado no respectivo registro através do POKE, o hardware do Atari assume o resto do trabalho. Isso ocorre em velocidade de código de máquina e, portanto, é muito mais rápido do que se o processo fosse controlado pelo BASIC.

Examinemos agora a criação de objetos de PM e os registros qué os controlam. Os jogadores são desenhados a partir de uma faixa vertical, com uma largura de oito pixels e altura de 128 ou 256 pixels. Cada linha dessa faixa é representada por um único byte na memória do computador. Inserindo-se, por intermédio do comando POKE, os códigos binários adequados, é possível definir a forma de um jogador, empregando-se um método semelhante àquele usado para criar caracteres estabelecidos pelo usuário (ver p. 246). Até quatro jogadores podem ser de-

```
10 SID=54272
20 POKESID+23,0
30 POKESID+24,15
40 POKESID+5,40
50 POKESID+6,201
60 FOR N=1TO 5
70 READ FH,FL,D
80 POKESID+1,FH:
   POKESID,FL:
   REM * NOTA DE
   JOGO *
90 POKESID+4,33
100 FOR I=1 TO 300 *
   D: NEXT I
110 POKESID+4,32
120 FOR I = 1 TO 100:
   NEXT I
130 NEXT N
140 FOR I=1 TO 2000:
   NEXT I
150 POKESID+24,0
160 REM ** FH FL D **
170 DATA 57, 172, 1
180 DATA 64, 188, 1
190 DATA 51, 97, 1
200 DATA 25, 177, 1
210 DATA 38, 126, 2
220 END
```

para enfatizar as freqüências em torno dos pontos de corte. A filtragem do envelope é um caso especial: ela tem um efeito diferente das outras, pois os valores de ADSR digitalizados estabelecidos para a envolvente 3 podem ser lidos do chip SID e aplicados a um sinal, de tal forma que a estrutura harmônica muda através do curso de uma nota. Ela funciona como um filtro variável.

Estas características variadas permitem a formação de sons altamente complexos e, com eles, a produção de efeitos interessantes e emulações convincentes de instrumentos convencionais. O aspecto desfavorável do SID é que o CBM BASIC V2, dialeto fornecido juntamente com o Commodore 64, não apresenta qualquer espécie de comandos dedicados ao som. O controle é exercido através da utilização dos registros de controle 29 SID por meio dos comandos PEEK e POKE. Assim sendo, uma grande quantidade de código BASIC se faz necessária para gerar efeitos ainda que simples, e em alguns casos o BASIC não tem velocidade suficiente para fazer justiça a todas as possibilidades do SID.

Uma descrição completa dos registros de controle SID exigiria um espaço maior do que uma edição completa do MICROCOMPUTADOR — CURSO BÁSICO, mas ainda assim é possível apresentar a forma de obtenção de algumas notas harmoniosas, como se vê no programa à esquerda.

Embora o programa tenha 22 linhas, ele toca apenas cinco notas de uma melodia simples em um oscilador. A linha 20 desliga o filtro dos osciladores, a linha 30 eleva o volume principal ao máximo e as linhas 40 e 50 especificam um envelope do tipo piano. A linha 80 estabelece a freqüência da nota, 90 e 100 dão início e fim ao ciclo ADSR e selecionam uma onda serrilhada para a voz 1; e a cronometragem é realizada através de loops do tipo FOR-NEXT nas linhas 100, 120 e 140.

Para se programar som no Commodore 64 em BASIC, é necessário esforço em termos de aprender e escrever um código. Além do mais, isso pode se constituir em um exercício frustrante, uma vez que a única maneira de descobrir se um conjunto de instruções mais complexas em BASIC será executado em tempo hábil é através de tentativa e erro. Para simplificar, vale a pena pesquisar os diversos programas de edição de som. Eles são geralmente escritos em linguagem de máquina e tiram o máximo proveito das maravilhosas características do Commodore 64.

finidos dessa forma, cada um ocupando seus 256 ou 128 bytes de memória.

Cada um dos jogadores tem a figura de um míssil associada a ele, cuja largura é de 2 bits. Para criar jogadores e mísseis, é necessário inserir através do comando POKE os padrões de bits que irão definir sua forma numa determinada área da memória. A área de RAM usada pode ser escolhida pelo programador, mas o computador deve ser informado colocando-se um indicador no começo da área.

Se o programador decidir usar resolução vertical de pixel unitário, então será preciso utilizar o dobro da memória necessária para resolução vertical com dois pixels. O programa a seguir desenha o jogador 0 em resolução vertical com dois pixels, na forma de espaçonave:

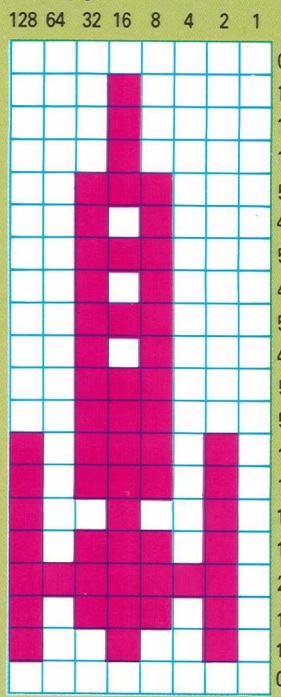
```
10 REM *** DEFINIR UM JOGADOR ***
20 P = PEEK(106) - 8:REM ATRIBUI 2 K A P
   ABAIXO DO TOP DA RAM
30 POKE 54279,P: REM ATRIBUI O POINTER A
   AREA PM
40 BASE = 256*P:REM ATRIBUI O ENDERECO
   BASE DA AREA PM
50 FOR I = BASE+512 TO BASE+640
60 POKE I,0:REM LIMPA A AREA DO JOGADOR 0
70 NEXT I
80 FOR I = BASE+512+50 TO BASE+530+50
90 READ A:POKE I,A:REM DEFINE FIGURA
100 NEXT I
110 DATA 16, 16, 16, 56, 40, 56, 40, 56, 40
120 DATA 56, 56, 186, 186, 146, 186, 254, 186, 146
```

Cada jogador tem diversos registros associados a ele. Estes controlam cor, posição horizontal e tamanho. O último deles permite que o programador aumente a largura de um jogador com um coeficiente 2 ou 4. Outros registros controlam a prioridade da relação jogador-cenário de fundo. Os mísseis têm a mesma cor do seu jogador, mas seu tamanho pode ser mudado separadamente. Para aplicações de jogos, uma série de registros é separada para detectar colisões na tela entre jogadores, mísseis e cenário de fundo. Entretanto, não há registros para posição vertical de mísseis ou jogadores. Realiza-se o movimento vertical movendo o conteúdo de cada localização que mantém os padrões de bits para a figura através da área de memória destinada àquele jogador. Esta é uma tarefa executada diretamente quando se trata da linguagem Assembly, mas que seria um tanto lenta se processada em BASIC. É uma boa idéia fazer bem baixas e atarracadas as figuras que se movem no sentido vertical.

Os gráficos Player-Missile ampliam em muito o potencial gráfico do Atari, embora eles não sejam tão versáteis ou fáceis de usar como os sprites do Commodore 64. Eis aqui a continuação de um programa, para colorir uma espaçonave e movê-la da esquerda para a direita na tela.

```
130 POKE 559,46:REM PERMITE PM DISPLAY DE 2
    LINHAS
140 POKE 53277,3:REM DISPLAY DE PM
150 POKE 704,88:REM COR ROSA DO JOGADOR 0
160 GRAPHICS 0
170 SETCOLOUR 2,8,2:REM ATRIBUI
    AZUL-ESCURO AO FUNDO
180 FOR I = 0 TO 320
190 POKE 53248,I:REM ACERTA A POSICAO
    HORIZONTAL
200 NEXT I
210 END
```

**Foguete PM**
Antes que um objeto jogador possa ser definido, ele deve ser desenhado e os valores decimais para cada linha de pixels devem ser calculados.



# Trocando de lugar

**Após examinar meios para inserção de novos registros, passamos agora aos métodos para recuperá-los. Como previsto, o problema inicial consiste em determinar a combinação exata.**

Encerramos nosso último fascículo com um exercício para desenvolvimento de um programa do tipo "banco de dados", que admitia o fornecimento de dados. Vamos observar alguns dos procedimentos incluídos no fornecimento de um novo registro, em continuação ao nosso exame sobre procedimentos exigidos pelo estágio INICIALIZAR de nosso programa principal. Primeiramente, suponhamos a existência dos seguintes campos, com suas matrizes correspondentes:

| CAMPO | MATRIZ |
|---|---|
| 1 campo do NOME | CAMPNOM$ |
| 2 campo do NOME MODIFICADO | CAMPMOD$ |
| 3 campo da RUA | CAMPRUA$ |
| 4 campo da CIDADE | CAMPCID$ |
| 5 campo do ESTADO | CAMPEST$ |
| 6 campo do NÚMERO TELEFÔNICO | CAMPTEL$ |
| 7 campo do ÍNDICE | CAMPIND$ |

É relativamente fácil entender o sentido da maior parte desses campos, com exceção, talvez, dos campos 2 e 7. Examinemos primeiro o campo no NOME MODIFICADO. Quando inicialmente analisamos o problema do formato dos dados para o nome, discutimos quanto a fazer com que o formato especificado de modo preciso (rígido) ou de modo vago (sem padrão), tendo optado por esta última alternativa. Uma vez que os nomes podem ser fornecidos de forma muito variada, o formato rígido tornará complexas as rotinas de busca e de ordenação. Para solucionar este problema, optamos pela conversão de todos os nomes para um formato padrão: todas as letras convertidas para maiúsculas, todos os caracteres não alfabéticos (como os espaços, sinais de ponto, apóstrofos etc.) eliminados, e um único espaço entre o nome (se houver) e o sobrenome.

A necessidade de uma padronização de nomes como essa deve-se ao fato de que as rotinas de busca e ordenação precisam de um método de comparação entre os elementos semelhantes. Por outro lado, ao recuperarmos nomes e endereços de um banco de dados, vamos querer a apresentação dos dados sob a forma inicialmente fornecida. Há dois métodos de lidar com o problema: ou cada campo de nome é convertido para uma forma padrão, apenas quando os procedimentos de ordenação e de busca estão se realizando, ou, então, o campo do nome pode ser convertido para uma forma padrão e armazenado como um campo separado, de modo que as rotinas de busca e ordenação possam ter acesso imediato aos nomes padronizados.

Ambos os métodos apresentam vantagens e desvantagens. A conversão temporária dos campos de nomes requisitados por outras rotinas economiza espaço de memória, pois é necessário armazenamento de menos dados no arquivo. Por outro lado, este procedimento consome muito tempo. Entretanto, se reservarmos um campo separado para a forma padronizada do nome, a conversão deverá ser executada uma única vez no registro. E, embora haja consumo de memória adicional, os procedimentos de busca e ordenação serão executados de modo mais rápido.

Outro campo que pode ocasionar confusão é o campo do ÍNDICE, que é incluído como um campo de reserva para possibilitar futura ampliação ou alteração do banco de dados, sem necessidade de reestruturação complementar do programa. Sua inclusão introduz a questão da "ligação" — termo que significa a determinação das relações de dados e de processamento. Todos os campos ou elementos em cada um dos registros são ligados porque possuem o mesmo índice (o mesmo elemento numérico ou índice em suas respectivas matrizes), e porque todos os campos de um registro serão armazenados juntos em um arquivo. Isto pode tornar difícil o procedimento de acréscimos de novos tipos ou relações de dados em estágios avançados, com a possível exigência de reorganização completa da estrutura do arquivo e uma reestruturação mais detalhada do programa. A incorporação do campo ÍNDICE neste estágio tornará muito mais simples as alterações futuras do programa.

Antes de tentar acrescentar novos registros ao banco de dados, faremos algumas suposições quanto à estrutura dos arquivos. Primeiro, limitaremos o número de registros a um total de 50, embora este número seja muito reduzido para uma agenda de endereços prática; posteriormente examinaremos como lidar com grandes quantidades de dados. Vamos supor também que todos os dados já foram transferidos — como parte do procedimento INICIALIZAR — para matrizes.

O acréscimo de um novo registro é de execução mais fácil no fim do arquivo (isto é, no primeiro elemento vazio de cada matriz). A possibilidade de que o novo registro esteja fora de ordem com relação aos demais é grande, porém este é um problema que poderá ser examinado posteriormente. Assim, a primeira coisa a fazer será determinar o tamanho da matriz. Uma vez que esta unidade de dados provavelmente será útil em muitas partes do programa, o melhor lugar para fazê-lo é o procedimento INICIALIZAR. Trata-se de um exemplo claro da necessidade de variáveis globais (isto é, variáveis que podem ser utilizadas em qualquer parte do programa). Nós a denominaremos TAMANHO. Outra variável global que provavelmente será útil consiste no índice do registro em uso corrente. Uma vez que nenhum registro estará em uso corrente, quando o programa for processado pela primeira vez, a atribuição de um

valor inicial à variável CORR deverá aguardar até que o programa realize algum procedimento com os dados. Todavia, a variável CORR pode ser inicializada em 0 no procedimento de INICIALIZAÇÃO. A inicialização de variáveis em 0 não é estritamente necessária na linguagem BASIC, pois se realiza de forma automática. Entretanto, é uma boa prática e deve ser sempre realizada em variáveis locais, a fim de evitar "efeitos colaterais" resultantes do emprego da mesma variável em outro ponto do programa.

Quando o programa for processado pela primeira vez, vários tipos de inicialização terão lugar e os dados serão fornecidos pelo disco ou fita e transferidos para as variáveis alfanuméricas. O menu ESCOLHA será então apresentado. Se o usuário escolher a opção 6 (para acrescentar um registro ao arquivo), o valor que a variável ESCOLHA apresentará será 6, o que chamará a sub-rotina ACREREG. Esta pressupõe que a variável TAMANHO já posssui um valor que lhe foi atribuído e pode, desse modo, iniciar a solicitação de entradas (obs.: também supõe que o procedimento INICIALIZAR já DIMensionou corretamente as matrizes necessárias).

O acréscimo de novos registros significa ainda que o arquivo está agora, pelo menos potencialmente, fora da ordem. Uma vez que a ordenação pode exigir um certo tempo, talvez não seja necessário ordenar os registros após a realização de cada acréscimo — essa é uma decisão que deixaremos de lado por enquanto. Em vez disso, estabeleceremos uma flag que indicará que o novo registro foi acrescentado.

Agora estamos em condição de começar a fazer uma lista experimental das possíveis matrizes, variáveis e flags que podem ser necessárias ao programa.

**TABELAS**

| | |
|---|---|
| CAMPNOM$ | (campo do nome) |
| CAMPMOD$ | (campo do nome modificado) |
| CAMPRUA$ | (campo da rua) |
| CAMPCID$ | (campo da cidade) |
| CAMPEST$ | (campo do Estado) |
| CAMPTEL$ | (campo do número telefônico) |
| CAMPIND$ | (campo do índice) |

**VARIÁVEIS**

| | |
|---|---|
| TAMANHO | (tamanho atual do arquivo) |
| CORR | (índice do registro corrente) |

**FLAGS**

| | |
|---|---|
| RACR | (novo registro acrescentado) |
| ORDEM | (ordenado depois da modificação do registro) |
| GRAV | (gravação realizada após a modificação do registro) |
| RMOD | (modificação realizada depois da última gravação) |

É provável que no decorrer do desenvolvimento do programa mais algumas matrizes sejam necessárias. Seguramente haverá necessidade de mais variáveis. Embora seja evidente que outras flags terão de ser incluídas, é possível que as quatro acima apresentadas não venham todas a ser requisitadas. Não será preciso gravar nem ordenar o arquivo (supondo-se que já esteja gravado e ordenado), a menos que alguma alteração tenha se realizado; assim, a flag RMOD será provavelmente a única necessária. Porém, se decidirmos utilizar as quatro flags, o subprograma de INICIALIZAÇÃO deverá atribuir a todas elas seu valor correspondente. Como treino complementar da elaboração de programas top-down, observamos a facilidade com que se pode codificar a variável *ACREREG*.

**I 4(EXECUTAR)6(ACREREG)**

```
INICIAR
   Localizar o tamanho atual do arquivo
   Indicar para entradas
   Atribuir as entradas no final das matrizes
   Acertar a flag RMOD
ENCERRAR
```

**II 4(EXECUTAR)6(ACREREG)**

```
INICIAR
   (o tamanho do arquivo é TAMANHO)
   (indicação para entradas)
   Limpar a tela
   Imprimir mensagem indicando para a primeira
   matriz (TAMANHO)
   Fornecer os dados para a matriz (TAMANHO)
   (indicar e dar entrada para todas as matrizes)
   Atribuir o valor 1 a RMOD
ENCERRAR
```

Todo este processo é direto e não exige loops ou outras estruturas complexas. O passo seguinte poderá ser a codificação direta para a linguagem BASIC. O único ponto a observar consiste na variável TAMANHO, que deve ser estabelecida durante a execução do procedimento INICIALIZAR e não precisa ser codificada como parte desta unidade.

**III 4(EXECUTAR)6(ACREREG)CÓDIGO BASIC**

```
CLS: REM OU USAR PRINT CHR$(24) ETC. PARA
   LIMPAR A TELA
INPUT "FORNECER O
   NOME";CAMPNOM$(TAMANHO)
INPUT "FORNECER A
   RUA";CAMPRUA$(TAMANHO)
INPUT "FORNECER A
   CIDADE";CAMPCID$(TAMANHO)
INPUT "FORNECER O
   ESTADO";CAMPEST$(TAMANHO)
INPUT "FORNECER NÚMERO TELEFÔNICO";
   CAMPTEL$(TAMANHO)
LET RMOD = 1
LET CAMPIND$ = STR$(TAMANHO)
GOSUB *NOMEMOD*
RETURN
```

Na antepenúltima linha, há a atribuição do campo CAMPIND$ ao valor da variável TAMANHO (convertida em uma variável alfanumérica pela função STR$), de modo a funcionar como índice em estágios posteriores. A sub-rotina *NOMEMOD*, chamada logo após o encerramento do programa, corresponde ao programa descrito pormenorizadamente na página 254. Algumas leves alterações serão necessárias nesse programa, porém apenas como detalhamento. Esta sub-rotina tem como função tomar a entrada do nome na ordem habitual (sem padrão definido) e colocá-la na forma padrão. A saída desta sub-rotina será um elemento (TAMANHO) na matriz denominada CAMPMOD$. Todos os procedimentos de busca de nomes e de ordenação podem agora ser realizados entre os elementos na tabela

CAMPMOD$ e, uma vez que o elemento terá o mesmo índice que os outros campos no registro, será fácil apresentar o nome e o endereço do modo como foram inicialmente fornecidos. Em outras palavras, a busca será realizada na tabela CAMPMOD$, porém a apresentação provirá da variável CAMPNOM$.

Isto é quase tudo o que se exige para o acréscimo de novos registros ao arquivo, embora não tenhamos feito concessões a erros de verificação nem previsões para o caso de não haver espaços disponíveis na matriz. Uma vez que nossos programas são desenvolvidos de forma modular, alterações e elaborações como essas poderão ser efetuadas posteriormente, sem a necessidade de reestruturação de todo o programa.

Os subprogramas REGMOD e REGELIM (respectivamente, para alterar e eliminar registros) são muito semelhantes ao programa ACREREG, exceto pelo fato de termos de localizar o registro que desejamos alterar para poderem ser executados. Por conseguinte, esses dois subprogramas deverão iniciar pela chamada do subprograma ENCREG, que segue uma rotina de busca semelhante à descrita na página 273. Aqui, a principal diferença está em que (muito provavelmente) não haverá dois itens de dados idênticos, pois poucas são as pessoas que possuem o mesmo nome.

Há dois modos de realizar uma rotina de busca. O primeiro deles consiste em examinar uma lista não ordenada, o que torna a busca mais lenta do que seria necessário. Na pior das hipóteses, a rotina deverá examinar todos os itens antes de localizar o item buscado. Entretanto, a busca em uma lista não ordenada tem a vantagem de que as rotinas de busca não são necessárias a cada vez que um novo registro é acrescentado, eliminado ou modificado.

Se houver algum tipo de ordenação dos dados — numérica ou alfabética, por exemplo —, o programa deverá examinar apenas uma pequena parte dos itens na lista. Quanto maior a lista, mais eficiente se tornará a busca binária, em comparação com a busca através de uma pilha não ordenada. De fato, se no arquivo houver dados suficientes que o garantam, a ordenação dos registros após uma alteração pode ser acelerada pela realização de uma busca preliminar para localizar a primeira e a última ocorrência, na matriz, da letra inicial do sobrenome no registro em questão.

Outro modo de acelerar a rotina de ordenação pode ser a manutenção de uma tabela de consulta das posições na matriz em que pela primeira vez cada letra do alfabeto aparece. Todavia, essa tabela deverá ser cuidadosamente mantida (atualizada) sempre que houver alterações nos dados.

O problema da busca e da ordenação é uma das áreas mais amplas da programação, e muitos livros foram escritos sobre a questão. Não tentaremos encontrar a solução definitiva para o programa da agenda de endereços, uma vez que isso depende de uma série de fatores, inclusive do número de registros no arquivo e da disponibilidade ou não de unidades de discos.

Apresentamos, agora, programas em pseudolinguagem para um exame dos elementos da tabela CAMPMOD$. A variável alfanumérica CHAVE$ é a chave para a busca. Neste contexto, o termo "chave" corresponde ao grupo de identificação de caracteres utilizados na especificação do registro (ou registros) exigido.

```
Indicar para a busca do nome
LETCHAVE$=nome (a ser buscado)
LET BTM=1
LET BUSCA=0
LET TOP=TAMANHO
executar um LOOP enquanto
   (BTM<=TOP)E(BUSCA=0)
   LET MID=INT((BTM+TOP)/2)
   IF CHAVE$=CAMPMOD$(MID)
      THEN
         PRINT CAMPNOM$(MID)
         PRINT CAMPRUA$(MID)
         PRINT CAMPCID$(MID)
         PRINT CAMPEST$(MID)
         PRINT CAMPTEL$(MID)
         LET BUSCA=1
      ELSE
         IF CHAVE$>CAMPMOD$(MID)
                  THEN LET BTM=MID+1
                  ELSE LET TOP=MID-1
                   ENDIF
   ENDIF
ENDLOOP
IF BUSCA=0 THEN PRINT "REGISTRO NÃO
   ENCONTRADO"
END
```

Esta parte de pseudolinguagem segue basicamente o programa empregado na busca das contagens dos jogos de futebol da página 275, porém você perceberá que não há uma saída conveniente se o registro não puder ser encontrado (a última instrução PRINT), o que será realizado apenas se o loop não puder localizar uma correspondência exata entre as variáveis CHAVE$ e CAMPMOD$(MID).

Infelizmente, é pouco provável o encontro de uma combinação perfeita, mesmo que o nome e o número telefônico que você deseja estejam no banco de dados. Isto se dá porque a instrução IF CHAVE$=CAMPMOD$ é totalmente invariável e não pode admitir a mínima diferença entre a série de caracteres fornecida pelo usuário na resposta à indicação e a série de caracteres armazenada na variável CAMPMOD$(MID). Em agendas de endereços comuns, examinamos a página com o olhar e isso nos capacita a aceitar os vários tipos de pequenas diferenças na apresentação efetiva do registro e aquilo que estamos buscando. O computador, porém, não pode realizar isso.

Entretanto, há recursos para superar o problema, embora todos exijam esforço de programação e um pouco mais de tempo para seu processamento. A primeira elaboração consistirá em verificar inicialmente só o sobrenome, e por este motivo é conveniente que o nome armazenado na variável CAMPMOD$ se apresente sob a forma de SOBRENOME (espaço) PRENOME. Desenvolvemos uma rotina para inverter a ordem de nomes em um fascículo anterior do curso de programação em linguagem BASIC e ela pode ser incorporada como uma sub-rotina no interior de uma rotina ACREREG, quando o campo CAMPMOD$ for criado.

Tendo conseguido localizar a primeira ocorrência

do sobrenome procurado, a rotina ENCREG deverá conferir a parte prenome desse elemento, de modo a verificar se é idêntico ao nome fornecido (CHAVE$). Se for, não haverá problema — o registro terá sido localizado. Entretanto, se não for, o problema começa a se complicar e temos de planejar nossa estratégia com cuidado. Poderíamos, por exemplo, examinar todos os prenomes e, se não encontrarmos uma correspondência exata, começaremos a procurar uma correspondência aproximativa. A dificuldade aqui é a seguinte: o que precisamente constitui uma correspondência exata?

Em vez de uma mensagem "REGISTRO NÃO ENCONTRADO", como no programa acima, poderá ser melhor uma mensagem como "CORRESPONDÊNCIA EXATA NÃO ENCONTRADA, DEVE SER TENTADA CORRESPONDÊNCIA APROXIMATIVA? (S/N)" O que significam as palavras "correspondência aproximativa"? Beto pode ser considerado uma correspondência aproximativa a Roberto? E quanto a Robert? Ambos apresentam entradas possíveis para o programa ENCREG. Vamos tentar definir o que queremos dizer com correspondência aproximativa e começar a desenvolver um programa em linguagem BASIC para determinar a correspondência mais aproximativa de uma série fornecida.

Suponhamos que a série na memória corresponda a ROBERTO. Qual das duas correspondências é a mais aproximativa: ROB ou RBRT? A segunda apresenta quatro das sete letras, enquanto a primeira, apenas três entre sete. Por outro lado, a primeira apresenta três letras na seqüência correta, enquanto a segunda, apenas duas.

A escolha é muito arbitrária. Daremos prioridade a uma correspondência exata entre a variável CHAVE$ e uma parte da variável do nome na memória. Se não for possível encontrar correspondência exata com uma parte da variável, o programa tentará obter o número maior de letras em comum. Eis aqui o programa apresentado em termos de entrada e saída:

**ENTRADA**
Uma série de caracteres
**SAÍDA**
A correspondência mais aproximada à série fornecida

O programa seguinte, em pseudolinguagem, próxima à linguagem BASIC, examinará as variáveis alfanuméricas em uma matriz, bem como as primeiras "n" letras em cada uma, onde "n" corresponde ao número de letras na chave (CHAVE$). Se não houver correspondência, será apresentada uma mensagem nesse sentido.

```
DIM MATRIZ$(4)
FOR L = 1 TC 4
READ MATRIZ$(L)
NEXT L
DATA "ROBERTO", "RICARDO", "ROBIANA",
   "ROBERTA"
LET CHAVE$ = "RON"
LET LCHAVE = LEN(CHAVE$)
LET BUSCA = 0
LOOP PARA ÍNDICE = 1 TO 4
   IF CHAVE$ = LEFT$(MATRIZ$(ÍNDICE),
   LCHAVE)
      THEN PRINT "A CORRESPONDÊNCIA E ";
         MATRIZ$(ÍNDICE)
         LET BUSCA = ÍNDICE
   ENDIF
ENDLOOP
IF BUSCA = 0
      THEN PRINT CHAVE$; "NÃO CORRESPONDE
      A NENHUM DOS"
         PRINT LCHAVE; "PRIMEIROS CARACTERES"
```

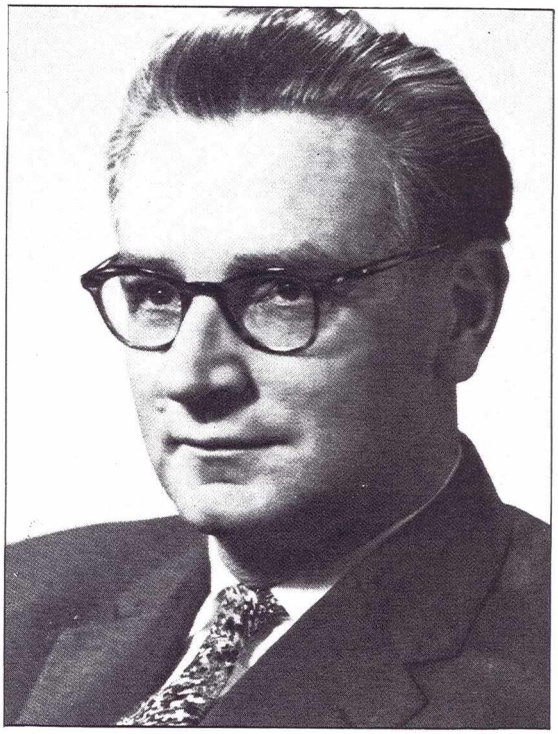
Após isso, o programa poderá prosseguir examinando os grupos de caracteres de comprimento LCHAVE, a começar pelo segundo caractere em cada matriz. Se nenhum desses grupos corresponder, pode-se buscar grupos que se iniciam com o terceiro caractere, e assim por diante. Finalmente, se não houver correspondência entre quaisquer dos grupos de três caracteres na série, o programa tentará determinar qual das séries apresenta o maior número de letras em comum com a variável CHAVE$. Esta tarefa é deixada ao leitor como exercício.

Podemos, de fato, escrever muitas páginas sobre a questão da ordenação "embaralhada" e a das diversas técnicas empregadas em pacotes de bancos de dados à venda no mercado. A maioria apresenta o recurso de buscar nos primeiros poucos caracteres do campo, como o código que acabamos de desenvolver. Outros recuperarão um registro se a seqüência especificada de caracteres se apresentar em algum local na tela, ou na verdade em qualquer ponto do registro. O recurso do "wildcard" é especialmente útil: a especificação J?N identificará JONAS ou JANE, porém não JOANA. A forma mais sofisticada de correspondência pouco definida funciona foneticamente, de modo que o fornecimento do nome MORAIS também identificará o nome MORAES.

## A propósito...

| Comando/Instrução/Função | Ação/Resultado | MS BASIC | Applesoft | TRS-80 | Sinclair |
|---|---|---|---|---|---|
| GET x$ | Provoca a parada do programa e aguarda por x$ | + | + | | |
| GET"arquivo",número | Lê um registro de um arquivo randômico | + | | | |
| GET (x1,y1)-(x2,y2),matriz | Lê informações gráficas da tela | + | | | |
| GOSUB linha | Chama uma sub-rotina da linha especificada | + | + | + | + |
| GOTO linha | Conecta o programa à linha especificada | + | + | + | + |
| GR | Comando para gráficos em baixa resolução (40x40) | + | + | | |

# Konrad Zuse



**Enquanto von Neumann desenvolvia seu trabalho pioneiro nos EUA, Zuse obtinha na Alemanha resultados iguais.**

Muitas vezes, descobertas são feitas simultaneamente, em diferentes partes do mundo, a partir de idéias desenvolvidas independentes umas das outras. Na década de 40, enquanto o primeiro computador a válvula (ENIAC) estava sendo desenvolvido nos Estados Unidos, o engenheiro alemão Konrad Zuse criava uma calculadora programável — discutivelmente, o primeiro computador do mundo.

Zuse nasceu em Berlim, em 22 de junho de 1910. Após cursar a Universidade de Tecnologia em sua cidade, trabalhou como engenheiro aeronáutico na Henschel Aircraft Company, onde desenvolveu novos projetos para asas de avião. Os princípios matemáticos básicos empregados para reforçar as asas da aeronave, a fim de suportarem as pressões do vôo em alta velocidade, foram estabelecidos na década de 20. Para fazer os cálculos necessários à produção das asas, eram empregadas várias equipes de técnicos, que trabalhavam com máquinas de somar mecânicas, ou seja, não elétricas, e réguas de cálculo. Zuse logo percebeu a necessidade de uma máquina que pudesse fazer rapidamente esse trabalho. E à noite, com a ajuda de amigos, no apartamento de seus pais, empenhou-se em construir um computador que pudesse realizar as tarefas.

Sua primeira máquina, Z1, fazia as quatro operações aritméticas, calculava raiz quadrada e convertia números decimais em notação binária e vice-versa. Desconhecendo as realizações de Charles Babbage (ver p. 220), que havia criado a máquina diferencial para fazer os complicados cálculos necessários à elaboração de tabelas de marés, Zuse chegou a muitas conclusões semelhantes e a algumas ainda mais avançadas. Sua maior descoberta foi a constatação de que uma alavanca era uma chave ou interruptor que, ficando em duas posições — ligada e desligada —, podia ser usada tanto para armazenar dados quanto como um dispositivo de controle.

Zuse fixou-se na idéia de representar os dados e as instruções em forma binária e, em 1941, construiu um computador eletromagnético que chamou de Z2. A princípio, o governo alemão, envolvido na guerra, demonstrou pouco interesse pela invenção de Zuse; mais tarde, porém, reconhecendo o potencial militar do invento, forneceu fundos para que ele desenvolvesse o novo Z3. Este seria um computador elétrico, com fiação elétrica em lugar das juntas mecânicas, que haviam sido usadas nos aparelhos anteriores, e cujo desenho seria mais compacto e elegante.

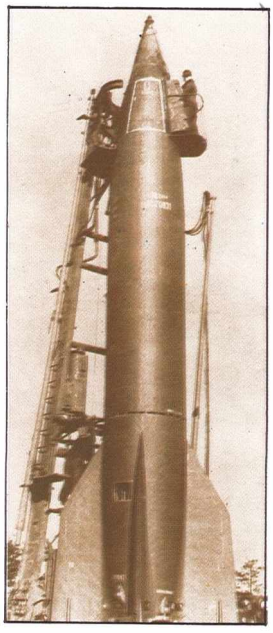
Zuse construiu o Z3 a despeito de todas as dificuldades. O bombardeio de Berlim pelos Aliados forçou-o a mudar sua oficina de lugar várias vezes e, além disso, em duas ocasiões foi convocado para lutar no front, de onde voltava para continuar seu trabalho. O racionamento de materiais durante a guerra obrigou-o a improvisar componentes a partir de peças usadas de telefone e, ao invés da fita de papel, velhos filmes de cinema, perfurados com códigos de oito furos por quadro.

O Z3 podia armazenar 64 palavras de 22 bits cada. A informação era introduzida através de um teclado e os resultados exibidos visualmente em um arranjo de lâmpadas montadas numa prancha. Infelizmente, tanto o Z3 quanto os computadores anteriores de Zuse foram destruídos no pesado bombardeio concentrado sobre Berlim, em 1945.

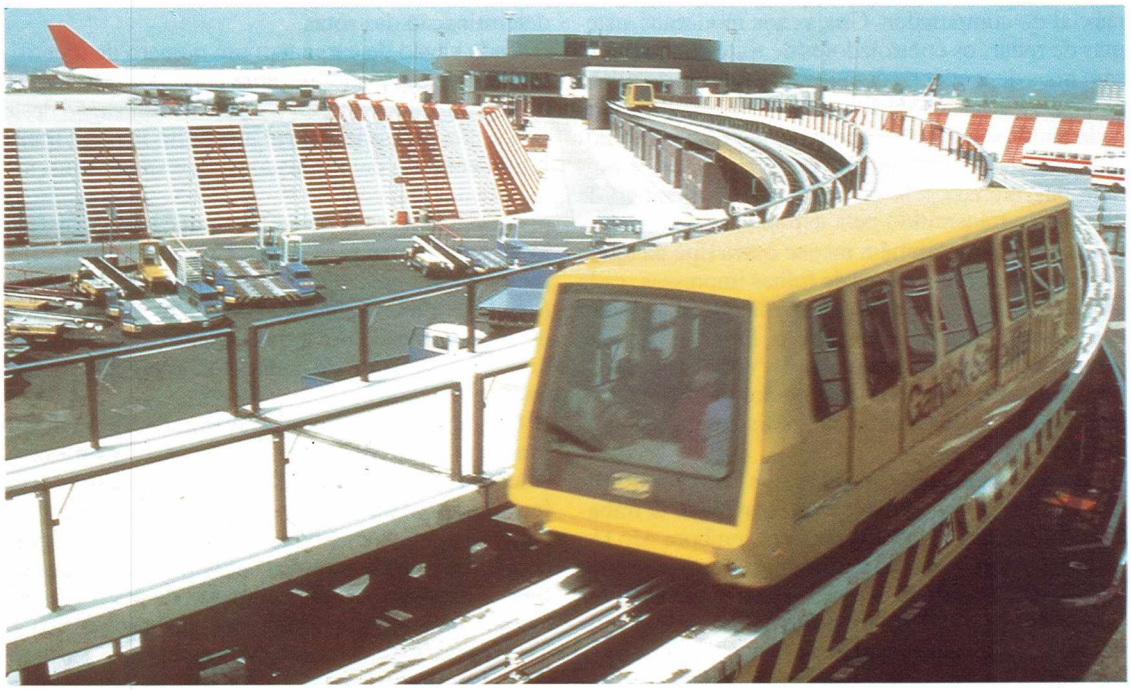
Um dos computadores foi adaptado pela Henschel Aircraft Company para trabalhar na construção da bomba voadora HS-293, que consistia em um pequeno avião não tripulado, lançado de um bombardeiro em pleno vôo e guiado para o alvo através de controle de rádio. O Z4, o último computador de Zuse na época da guerra, aumentou o tamanho das palavras para 32 bits. Foi levado para Göttingen quando os Aliados se aproximavam de Berlim e acabou em Basiléia, na Suíça, onde operou até 1954; era um dos computadores mais importantes da Europa, na época.

Zuse não conseguiu fabricar computadores na Alemanha do pós-guerra e, assim, concentrou-se na teoria de computadores, desenvolvendo uma linguagem sofisticada chamada PLANKALKÜL, que podia ser usada tanto para matemática quanto para informações mais generalizadas. Ao conseguir novamente fabricar computadores, montou a Zuse Company, a maior fabricante de computadores da Alemanha até 1969, então incorporada pela Siemens. Em 1984, o professor Zuse ainda continuava trabalhando na indústria de computadores.

**Bomba voadora**
Os computadores de Zuse foram desenvolvidos para substituir equipes de técnicos que trabalhavam com réguas de cálculo em projetos para a aeronáutica. Particularmente, foram utilizados para fazer o projeto das bombas V-1 e V-2 (foto), tão usadas na Segunda Guerra Mundial.



# Viajando



**"People Mover"**
O Aeroporto de Gatwick, ao sul de Londres, como muitos aeroportos americanos, instalou um revolucionário sistema de transporte interterminal, que combina a estabilidade direcional e a capacidade de operação automática da ferrovia com o conforto e a comodidade do ônibus. Projetado e produzido pela Westinghouse, famosa pelos seus sistemas de sinalização ferroviária, o "People Mover" (Transportador de Gente) pode transportar até cem passageiros.

**O transporte de mercadorias e pessoas de um lugar para outro, neste mundo cada vez mais populoso, é uma tarefa árdua. O uso do computador facilita bastante o trabalho.**

Há um século e meio, uma viagem da Europa à Austrália levava três meses. Nos anos 80, a mesma viagem não requer mais do que meio dia. Este milagre tecnológico seria impossível sem os sofisticados métodos de controle computadorizado de veículos.

De todos os meios de transporte, o aéreo é o que apresenta os problemas mais prementes. O Aeroporto de Heathrow, em Londres, por exemplo, tem um movimento superior a mil vôos diários, com um trânsito de 120 aviões por hora, nos períodos de pico. Sem a ajuda do computador para controlar essa atividade, o sistema seria inoperável.

Vejamos o caso de alguém que quer viajar de Londres a Nova Iorque. Desde a agência de viagens, onde a passagem é comprada e o lugar reservado (por meio do sistema Prestel como forma de acesso ao computador da empresa de aviação), até o pouso no Aeroporto Kennedy, cerca de quinze computadores diferentes estarão diretamente relacionados com a viagem. Examinemos com detalhes o papel do computador nas viagens aéreas.

O primeiro aspecto a ser considerado é o padrão da própria aeronave. Os modernos aviões de passageiros são caros. A fim de maximizar o retorno sobre o investimento, as empresas que operam as linhas aéreas devem mantê-los em condições impecáveis, o que requer uma "manutenção planejada". Após um determinado número de horas de vôo, o avião volta a sua base de engenharia, onde o pessoal tem acesso aos registros computadorizados do histórico completo do aparelho — desde o primeiro dia de sua construção até o número de série de todas as peças em uso ou usadas. Tudo é registrado nos mínimos detalhes: qualquer operação de engenharia por que o avião tenha passado; relatórios sobre seu desempenho, feitos por engenheiros de vôo e outros membros da tripulação; dados sobre consumo de combustível e qualquer outra informação que possa vir a ser de interesse. O usuário de um microcomputador doméstico poderia aplicar estes mesmos métodos — talvez não com tantos detalhes — para fazer a manutenção de seu carro.

O avião só volta ao serviço depois que o programa de manutenção estiver completo e atualizado. Neste ponto, ele se torna componente de um outro sistema computadorizado — o de controle operacional da linha aérea. Este sistema aloca o avião para as diferentes rotas; coloca avisos para abastecimento em vários pontos ao longo dessas rotas; faz os preparativos para a tripulação, refeições, diversões durante o vôo e uma infinidade de outros arranjos necessários para transportar trezentas ou quatrocentas pessoas pelo mundo todo.

Outro sistema de controle por meio de computador opera no próprio aeroporto, onde os funcionários têm de atender à enorme demanda das linhas aéreas, para as quais um atraso de poucos minutos pode representar uma perda considerável de dinheiro. O êxito dessa operação depende de tabelas de horário computadorizadas. Outros itens controlados pelo sistema do aeroporto são, por exemplo, a chamada de passageiros para embarque e o fornecimento de dados para os painéis que informam as chegadas e partidas.

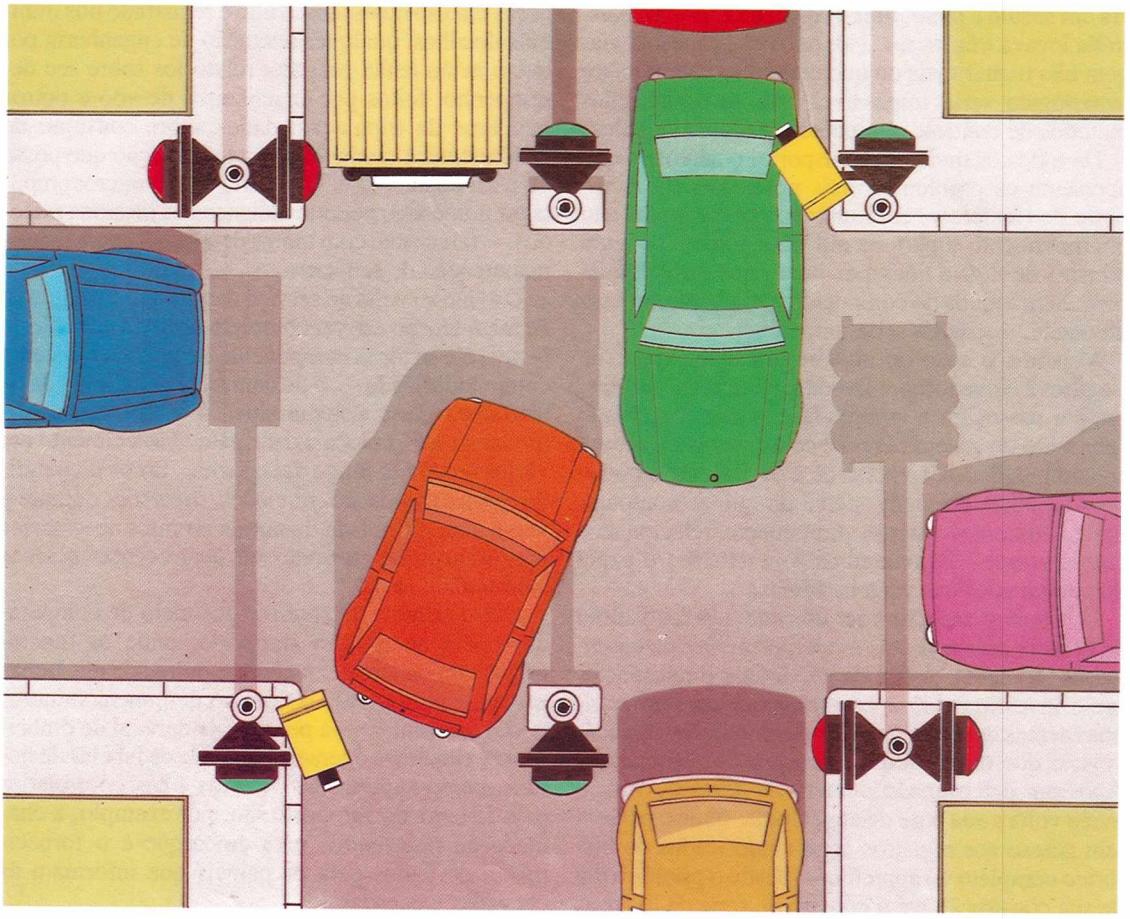
Antes mesmo que os passageiros cheguem ao aeroporto, o piloto registra um pormenorizado plano de vôo junto ao Controle de Tráfego Aéreo. Surpreendentemente, o trabalho do CTA tem apenas ajuda parcial do computador. Graças aos modernos sistemas de radar, os controladores de tráfego não precisam mais depender da comunicação verbal do piloto para saber a posição do aparelho. Na tela do radar, cada sinal vem identificado pelo seu respectivo número de vôo, acompanhado de uma leitura de altitude interpretada pelo computador e um código de destinação transmitido pela aeronave.

O controlador também recebe outro tipo de ajuda do computador, em forma de folhas impressas, cada uma delas cobrindo um segmento da rota planejada e baseada no plano de vôo do piloto. Essas folhas, que informam o curso e a altitude do vôo, carga útil e tipo de avião, ajudam o controlador a dirigir o vôo através de sua área da maneira mais rápida e econômica.

Um dos meios de transporte que também utiliza amplamente o computador é o ferroviário. O trabalho do sinaleiro da estrada de ferro, embora não tão complexo, tem algumas coisas em comum com o do controlador de tráfego aéreo. Ele tem também a função de dirigir o tráfego de carga e passageiros através de sua área, com segurança e a baixo custo. A British Rail está usando sistemas de controle computadorizados desde a metade da década de 70, seguindo o trabalho pioneiro empreendido nos Estados Unidos pela Southern Pacific Railroad. O Total Operations Processing System — TOPS (Sistema Total de Processamento de Operações) executa todos os itens do controle de trens de carga, desde a manutenção de um inventário preciso e superatualizado de todas as locomotivas e material rodante até sua utilização na montagem de trens completos e a determinação das rotas.

Cada vagão de carga tem um número de identificação, que é registrado, junto com sua localização, pelo computador do TOPS. Quando um vagão deste tipo se faz necessário em algum outro lugar, é transferido para um determinado trem e sua destinação, registrada. Quando ele chega ao destino, sua nova posição é registrada pelo TOPS, que pode transferi-lo novamente, se necessário. Considerando o volume de tráfego de carga que a British Rail transporta (havia 185.000 vagões em serviço no fim da década de 70, perfazendo um total de quase 2.000 trens por dia), estes sistemas de controle por computador tornam-se imperativos.

Operação de aeroportos e de estradas de ferro tem problemas em comum, e muitas vezes as mesmas soluções são empregadas. Entretanto, as estradas de ferro apresentam um fator que deve ser levado em consideração — o uso cada vez maior de estações automáticas. Muitas inovações vêm ocorrendo neste tipo de estações, como o emprego de microcomputadores para esclarecer dúvidas dos passageiros e, em alguns casos, de sintetizadores de voz para anunciar chegadas e partidas de trens. A computação também possibilitou a criação de trens automáticos. No metrô de Londres, por exemplo, a linha Victoria possui essa capacidade, embora não tenha sido posta em prática porque o público não confia em trens automáticos, dirigidos apenas pelo computador.



**Sinal verde**
A sincronização de semáforos foi empregada durante algum tempo pelos engenheiros de trânsito para regular o fluxo de veículos nas principais ruas e avenidas das grandes cidades. Atualmente, semáforos individuais podem monitorar a densidade de tráfego das proximidades, por meio de detectores de radar, passando os dados para um sistema de computador. A freqüência da mudança dos sinais é ajustável para adequar-se às condições do momento.




**Anunciando a partida**
O aspecto mais conhecido do sistema operacional de computador de um aeroporto é o quadro de chegadas e partidas. À medida que as informações são recebidas, os dispositivos eletromecânicos vão sendo modificados pelo computador central.

Pode-se estabelecer um paralelo entre sistemas de trens automáticos e as mais sofisticadas redes ferroviárias. Cada modelo de locomotiva tem um código de identificação, chamado "número de peça", armazenado em um microcomputador baseado num único chip. O controlador emite sinais para cada um dos trens, modulando a energia conduzida ao longo dos trilhos, de tal forma que só uma locomotiva será capaz de decodificar a mensagem a ela dirigida. Desta maneira, grande número de trens pode trafegar no mesmo sistema, a qualquer tempo, sob controle direto do microcomputador central.

Os trens automáticos tornaram-se exeqüíveis porque correm sobre trilhos, mas é pouco provável que o conceito se aplique aos veículos rodoviários. No entanto, computadores são usados no transporte rodoviário, principalmente no transporte coletivo e de carga. Eles ajudam a programar e determinar o itinerário dos veículos e também executam tabelas de horários, o que vem a ser um trabalho bem complexo numa cidade do tamanho de Londres, onde há necessidade de integrar os serviços de ônibus, trem e metrô em um único sistema de transporte coletivo.

O problema consiste em colocar em operação um número suficiente de veículos, isto é, que não sejam poucos a ponto de provocar atrasos inaceitáveis para os passageiros, mas que também não sejam em número tão grande a ponto de diluir os lucros operacionais. Trata-se de um problema estatístico complexo, e foi justamente para resolver esse tipo de questões que o computador foi criado. Outro problema que se beneficia de métodos estatísticos é a determinação do itinerário de caminhões de carga, minimizando a distância entre os pontos de entrega e a alocação de carga para cada caminhão. Uma variante interessante deste método é encontrada nas organizações que enviam veículos para atendimento de chamados, especialmente táxis e carros de polícia, que ficam "rodando" em vez de voltar à base. A localização de cada veículo entra no computador com o nome da rua em que se encontra e é convertido em uma referência de um mapa quadriculado. Um pedido de táxi ou de assistência em caso de emergência entra da mesma forma, e a tarefa de combinar recursos com pedidos é uma simples questão de comparar referências segundo regras preestabelecidas.

Uma das experiências mais interessantes envolvendo computadores no transporte coletivo é o sistema "ligue para um ônibus", atualmente em operação nos subúrbios de Hanover, na Alemanha. Baseado numa frota de microônibus que não têm rota preestabelecida, o sistema permite que o passageiro ligue do ponto de ônibus para a estação central de controle, informando seu destino. Então, um microterminal (que parece um terminal de caixa automático) imprime o horário em que o ônibus chegará (nunca em mais de cinco minutos), a duração da viagem (contando com as necessidades dos outros passageiros) e o preço da passagem.

O transporte de mercadorias e passageiros é responsável por 20% do comércio mundial. O uso de computadores nesta área está mais adiantado que em outras e contribuiu significativamente para seu crescimento. Embora a maioria dos exemplos que analisamos impliquem o uso de minicomputadores ou computadores de grande porte, muitos valem também para os micros pessoais. Há uma variedade de pacotes de software apropriados para distribuição, organização, programação de horário etc., que têm grande aplicação em equipamentos pequenos.



**Navios mercantes**
O emprego de computador na marinha mercante é menor que em outros meios de transporte, mas os serviços relacionados com *containers* representam uma área importante para sua aplicação. Microcomputadores são utilizados na distribuição da carga (diversos expedidores compartilhando o mesmo container), organização e planta do terminal de containers, bem como no carregamento dos navios.

# Idiomas diferentes

**Fácil de aprender, porque tem uma construção matemática familiar, o BASIC, no entanto, é grosseiro quando comparado a outras linguagens.**

Existe uma grande probabilidade de que o seu computador use o BASIC como linguagem de programação. Mas isto não quer dizer que você tenha de se restringir a esta escolha e, embora o BASIC seja considerado particularmente fácil de se aprender, há outras linguagens mais adequadas para escrever programas de aplicações específicas.

Para incorporar essas linguagens a seu computador, será necessário substituir as ROMs que contêm os interpretadores BASIC, ou carregar a nova linguagem em RAM — neste caso, você precisa de uma máquina com capacidade razoável de memória, de forma que sobre espaço de RAM para conter seus programas. Alguns microcomputadores, como o MZ-711 da Sharp, resolveram este problema carregando até mesmo o interpretador BASIC em fita cassete.

## PASCAL-PORTUGUÊS

## PORTUGUÊS-PASCAL

A linguagem PASCAL foi estruturada no começo da década de 70, para suceder ao BASIC. Sua variedade de dados e estruturas de controle deriva do grupo de linguagens FORTRAN/ALGOL, que tem por finalidade estimular o estudante a fazer a programação do computador de uma forma sistemática e a escrever códigos bem estruturados e de fácil entendimento. Este é um ponto necessário para o desenvolvimento de técnicas de programação de boa qualidade, mas significa que os primeiros estágios do aprendizado de programação são mais difíceis para o iniciante.

Eis aqui uma amostra de um programa em PASCAL, equivalente ao programa em BASIC:

```
VAR                    BEGIN
NOME        :PACKED      EXEC:=TRUE;
              ARRAY (1..30)  WHILE EXEC DO
              OF CHAR;       BEGIN
                             WRITE ('Qual e o seu nome?');
IDADE,CONT :INTEREGER;       READLN (NOME);
RESPOSTA   :PACKED           WRITE ('e qual sua idade?');
              ARRAY (1..3)   READLN (IDADE);
              OF CHAR;       FOR CONT:=1 TO IDADE DO
EXEC        :BOOLEAN;          WRITE (CONT:3, 'Alo':10, NOME);
                             WRITE ('Quer mais uma vez?');
                             READLN (RESPOSTA);
                             IF RESPOSTA (1)='N'
                             THE RUNNING:=FALSE;
                             END;
                           WRITELN ('Adeus', NOME)
                           END
```

## BASIC-PORTUGUÊS

## PORTUGUÊS-BASIC

A linguagem BASIC, muito difamada, foi desenvolvida a partir do FORTRAN (uma das primeiras linguagens de programação de alto nível e ainda a mais popular para aplicações científicas e de engenharia) como uma introdução tutelar à programação para estudantes universitários. Por ter sido criado para um uso na forma autodidática, o BASIC é geralmente mais interpretado que compilado (ver p. 184), e este foi o fator de haver se tornado a linguagem original de quase todos os microcomputadores. Linguagens interpretadas são fáceis de se implantar, usam comparativamente pouca memória do computador e são bastante adequadas para desenvolvimento de programas.

O BASIC é uma linguagem forte por si só, mas tem o inconveniente de uma variedade de dialetos não padronizados (todo BASIC de um computador é exclusivo da máquina) e de lhe faltarem dados e estruturas de controle especializados.

Este programa curto ilustra não só a atração e a generalidade, mas também as limitações do BASIC.

```
100 INPUT "Qual e o seu nome?";N$
200 INPUT "e qual sua idade?";I
300 FOR K = 1 TO I
400 PRINT K, "Alo";N$
500 NEXT K
600 INPUT "Quer mais uma vez?";R$
700 IF LEFT$(R$,1)="S" THEN GOTO 100
800 PRINT "Adeus";N$
900 END
```

## COMAL-PORTUGUÊS

## PORTUGUÊS-COMAL

A linguagem COMAL foi organizada para combinar a acessibilidade do BASIC com as sólidas estruturas e a execução disciplinada do PASCAL. Assim, ela se parece com as duas e teria servido de modelo para o desenvolvimento do BASIC específico do micro inglês BBC, que quase se transformou em uma nova linguagem. O COMAL é muito usado na computação escolar e na Escandinávia (seu lugar de origem), mas parece improvável que venha a substituir uma de suas antecessoras como linguagem introdutória à programação.

Este é o programa "Alô", em COMAL:

```
100  EXEC:=TRUE
200  WHILE EXEC DO
300     INPUT "Qual e o seu nome?":N$
400     INPUT "e qual sua idade?";I
500     REPEAT
600        FOR K:=1 TO I DO
700           PRINT K,"Alo";N$
800        NEXT K
900     INPUT "Quer mais uma vez?";R$
1000    IF R$="N" THEN EXEC:=FALSE
1100 ENDWHILE
1200 PRINT "Adeus";N$
```

# LISP-PORTUGUÊS

## PORTUGUÊS-LISP

O LISP foi desenvolvido no começo da década de 60 como uma linguagem de processamento de listas, e desde então tem sido amplamente utilizado na área da inteligência artificial, que envolve contínua busca e comparação de listas de dados, conexões e respostas. Ao contrário do BASIC, no qual se destaca o fluxo do programa por uma seqüência de instruções e procedimentos, o LISP é uma linguagem "funcional", em que o conjunto básico de comandos pode ser desenvolvido para construir funções mais sofisticadas, com nomes definidos pelo próprio programador. Por exemplo:

```
(SETQ MATRIZ1 '(4 7 2 5 1))
```

cria uma lista chamada MATRIZ1, cujos elementos são os números (4 7 2 5 1)

```
(CAR MATRIZ1)
```

fornece o primeiro elemento da lista MATRIZ1 (4, neste caso).

```
(CDR MATRIZ1)
```

fornece a lista MATRIZ1 com o primeiro elemento removido — (7 2 5 1) neste caso.

```
(SETQ MATRIZ1 (CDR MATRIZ1))
```

transforma a lista MATRIZ1 em uma cópia de si mesma, excluindo o primeiro elemento.

O LISP também é útil para aplicações "recursivas" — casos em que certos problemas, após a aplicação de uma simples função, são reduzidos a problemas menores, porém idênticos.

# FORTH-PORTUGUÊS

## PORTUGUÊS-FORTH

A linguagem FORTH se parece com o LOGO por ser funcional e interativa, mas apresenta uma diferença importante: é a primeira, depois do BASIC, a ser implementada num computador pessoal — o micro inglês Jupiter Ace. Ela consiste em um número de funções definidas, chamadas "primitivas", e tem a capacidade de definir novas funções a partir daquelas. As operações matemáticas em FORTH são "stack-oriented", o que significa que a memória do computador é tratada como uma lista de dados que aumenta e diminui e, como resultado, a última operação sempre encabeça a lista. Outra conseqüência deste método é que a notação algébrica não é usada. Em lugar de escrever (12 + 4)/2 para achar a média de 12 e 4, em FORTH escreve-se 12 4 + 2 /, que é a mesma quantidade em "notação inversa", em vez de notação algébrica.

Tudo isso transforma o FORTH num tipo de linguagem muito diferente. Ele pode ser considerado quase um retrocesso na hierarquia das linguagens de alto nível.

Este fragmento em FORTH define duas novas palavras chamadas GRITO e CORO:

```
:GRITO (prints "SHAZAM!")
   ."SHAZAM!";
:CORO (uses GRITO in a loop)
   0 DO GRITO LOOP;
```

Agora, se teclarmos n CORO, aparecerá SHAZAM! escrito n vezes na tela.

# LOGO-PORTUGUÊS

## PORTUGUÊS-LOGO

A linguagem LOGO foi estruturada pela equipe do professor Seymour Papert, que trabalhava com inteligência artificial. Ela se parece com o FORTH, tanto na interatividade quanto no uso de um número de funções "primitivas", que podem ser incorporadas em funções definidas pelo usuário. Esta linguagem é baseada em um princípio fundamental: uma maneira de aprender algo é ensinar alguém como fazê-lo — neste caso, o computador. O LOGO cria um método completamente novo de ensinar crianças a pensar. Costuma ser chamado de linguagem "tartaruga", porque é usado para controlar um pequeno robô sobre rodas, a tartaruga (ver p. 34).

Aqui apresentamos um fragmento em MLOGO (uma versão do LOGO em português), que desenha uma casa simbólica, traçando um quadrado de tamanho especificado, com um triângulo em cima:

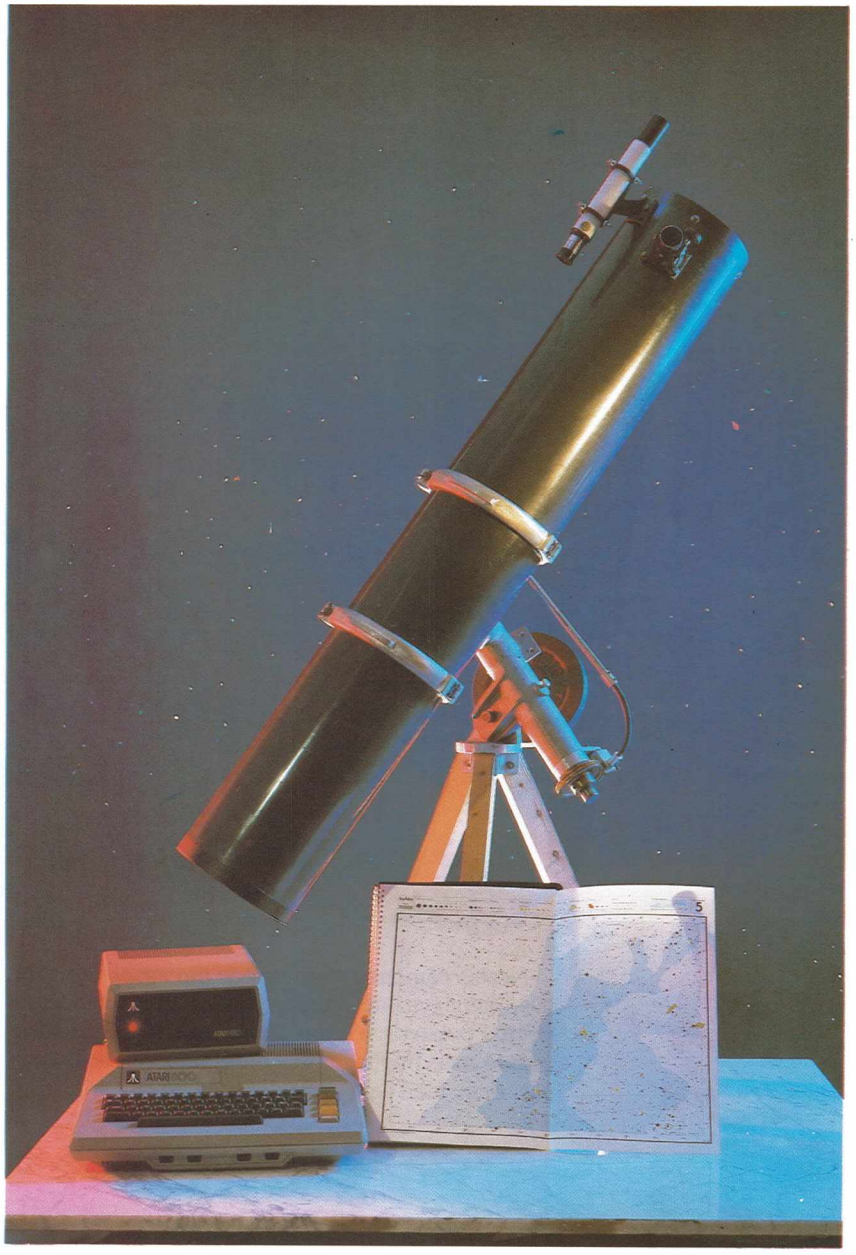
```
AP TRIANGULO: TAMANHO
   REPITA 3 (FRENTE: TAMANHO DIREITA 120)
FIM
AP QUADRADO: TAMANHO
   REPITA 4 (FRENTE: TAMANHO DIREITA 90)
FIM
AP CASA: TAMANHO
   DIREITA 30
   TRIANGULO: TAMANHO
   ESQUERDA 90
   QUADRADO: TAMANHO
FIM
```

Se agora teclarmos CASA 15, aparecerá o desenho de uma "casa" cujos lados medirão 15 unidades.

Nestas páginas apresentamos um apanhado geral das linguagens de programação mais comuns para micros pessoais. Assim como os idiomas falados, quanto mais linguagens de programação você dominar, mais fácil será aprender uma nova.

# Observando os astros

**Os computadores têm dois usos principais na astronomia: manter o banco de dados dos astros observados e calcular-lhes a posição atual para ajudar no alinhamento do telescópio.**



**Guiando a luz**
A astronomia óptica torna-se bem mais fácil quando se dispõe de uma previsão exata de localização de determinada estrela ou planeta, em determinado dia. Um microcomputador pode conter um banco de dados com essas informações e, com o auxílio de servomecanismos, até posiciona o telescópio.

Quando observamos o céu, à noite, e localizamos a estrela mais próxima, estamos vendo, na realidade, como ela era e onde estava há quatro anos, pois este foi o tempo que a luz levou para vir de Proxima Centauri até aqui. Nesse mesmo período, o microcomputador evoluiu, de uma novidade rara e dispendiosa, para um acessório relativamente comum no ambiente doméstico. Aplicações em astronomia utilizam ao máximo o potencial dessas máquinas para a manipulação de dados, computação e robótica.

Estudando o espaço celeste, o astrônomo encontra três problemas principais: a observação inicial do astro, a manipulação dos dados obtidos a partir da observação e a análise significativa destes dados. Em todas essas áreas, o computador é um auxiliar muito útil.

Vejamos como um microcomputador pode ajudar a fazer os cálculos necessários para construir o instrumento básico do astrônomo — o telescópio. A posição e o modelo das lentes e dos espelhos dentro desse aparelho são fundamentais para a qualidade da imagem final e podem ser calculados matematicamente, a fim de proporcionar os melhores resultados. Astrônomos amadores gostam de montar seus próprios sistemas ópticos. E quando o uso de microcomputadores ainda não estava difundido, era mais rápido e prático montar um modelo experimental do que executar os complicados cálculos necessários para obter os diagramas dos raios luminosos. Com o uso do computador, os cálculos que levavam uma semana passaram a ser feitos em poucos minutos.

Localizar uma estrela no céu é outro problema que pode ser resolvido com a ajuda do computador. Estrelas não são objetos fixos; elas traçam trajetórias no decurso da noite (ou seja, um efeito causado pela rotação da Terra), e suas posições estão sujeitas a variações periódicas. O método para localizar uma estrela é semelhante ao sistema de latitude e longitude usado na geografia terrestre. Se imaginarmos um sistema de coordenadas projetado através da abóbada celeste à noite, os objetos do firmamento podem ser localizados por duas coordenadas chamadas declinação e ascensão.

Cada objeto deve, então, ser marcado em um mapa estelar, de acordo com suas coordenadas, e os mapas individuais são agrupados em um atlas celeste. Tais atlas são muito importantes para a observação de planetas e outros corpos que se movimentam sobre um fundo de estrelas "fixas", ou para descobrir corpos novos — cometas, por exemplo. Os atlas estão sendo postos em bancos de dados de microcomputadores. Os programas desses bancos de dados também incluem informações a respeito do brilho e luminosidade dos corpos celestes, qualidade do espectro da luz por eles emitida (espectro obtido quando a luz passa através de um prisma ou é analisada por um espectrômetro), assim como o tipo e a idade de uma estrela. Todas as informações podem ser exibidas no monitor de vídeo ou no televisor acoplado ao computador, sob a forma de mapas que mostram o firmamento visto de qualquer latitude e em qualquer horário desejado.

Por causa do movimento de rotação da Terra, as estrelas saem do campo de visão em poucos minutos, ainda que o campo do telescópio seja amplo. Usando-se um que focalize uma área muito pequena

do céu, torna-se indispensável movimentá-lo continuamente para compensar a rotação da Terra. Durante muitos anos foram empregados motores de tração mecânica para movimentar os telescópios, mas recentemente os astrônomos amadores tiveram acesso aos sistemas controlados por computador. Dessa forma, o telescópio é montado sobre um eixo "equatorial" que aponta para o norte verdadeiro, e o motor gira este eixo na velocidade exatamente igual à da rotação da Terra, mantendo o objeto de forma permanente dentro do campo de visão. Ligados ao eixo e ao telescópio, codificadores e digitalizadores de raios luminosos emitem sinais para o computador, fornecendo as coordenadas celestes, e monitoram a atividade do motor de tração. Desta maneira, um telescópio pode passar a noite inteira, controlado apenas pelo computador, rastreando uma pequena estrela, cuja imagem vai sendo lentamente projetada sobre uma chapa fotográfica. Nos Estados Unidos há um adaptador chamado Celestial Navigator Mk II, que liga o telescópio ao microcomputador através de interfaces de expansão.

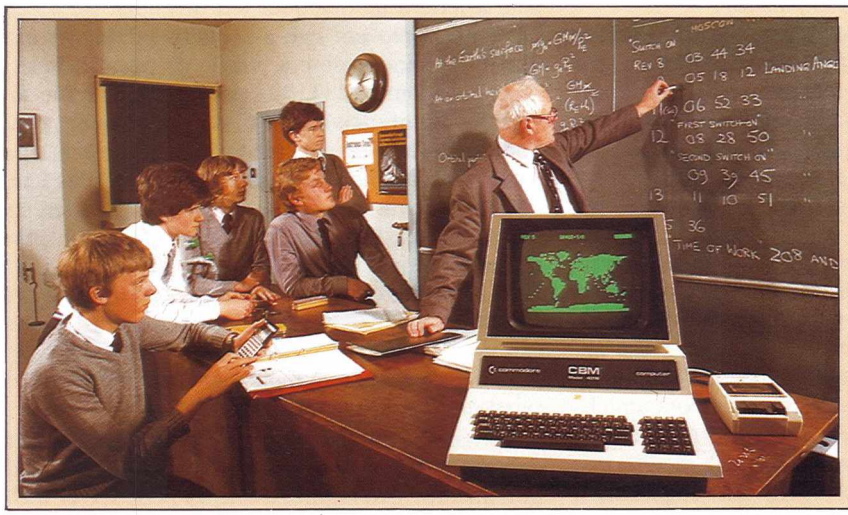
Alguns astrônomos amadores americanos já utilizam as novas tecnologias de reconhecimento da fala e sintetizadores de voz. Um computador é programado para reconhecer certas palavras de comando, de modo que quando um observador entra no recinto e diz "abra", a cúpula se abre; em seguida, ao dizer "gire a cúpula", o motor é ligado automaticamente, fazendo o domo girar sobre as rodas. O sintetizador de voz também é muito útil na escuridão de um observatório, emitindo informações provenientes do computador. Ele pode, por exemplo, contar o tempo em voz alta para ajudar o observador a fazer exposições fotográficas com precisão.

Astrônomos profissionais usam telescópios que registram a parte do espectro eletromagnético que não é luminosa — tais como ondas de rádio e raios X. Com o aumento da abertura do telescópio, e o conseqüente aumento de peso, os problemas de engenharia tornaram-se críticos. Em 1964, o Jodrell Bank Mark II, disco elíptico de 38 × 25 m, foi o primeiro telescópio a usar um computador digital para converter coordenadas celestes em instruções necessárias aos motores de tração — processo repetido continuamente quatro vezes por segundo.

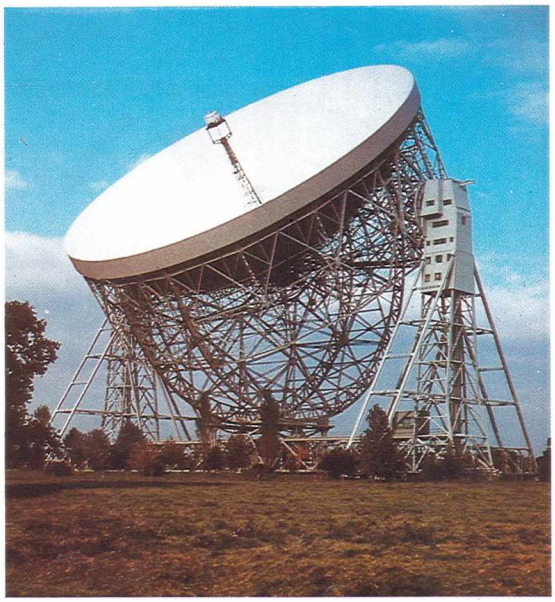
No trabalho com telescópios, os computadores são programados para levar em conta também as modificações atmosféricas — variações de temperatura e umidade, que refratam e desviam a luz; compensam e corrigem distorções da imagem recebida pelo telescópio, mediante diversas técnicas para realçar imagens.

A astronomia tem se desenvolvido graças ao uso do computador, e este, por sua vez, beneficiou-se com esta aplicação, atingindo maior aperfeiçoamento. A linguagem FORTH foi inventada em 1971 por um astrônomo, Charles H. Moore, no Observatório Kitt Peak, no Arizona, para controle de radiotelescópio e processamento de dados.

Atualmente, existem livros dedicados à programação de caráter amador para entusiastas da astronomia, e há uma revista, a *Apex*, em que os proprietários de microcomputadores trocam e publicam programas. Existe software para calcular os dias em que cairá a Páscoa, a conversão de datas históricas para o calendário juliano, o horário em que a Lua nasce e se põe, e tabelas diárias para marcar no céu a posição exata em que nós e os astrônomos veremos o retorno do cometa de Halley em 1986.




**Astros do colégio**
Os alunos e o corpo docente do Colégio Kettering, em Northamptonshire, na Inglaterra, são muito respeitados pelo seu trabalho em astronomia e na área de rastreamento de satélites. Em muitas ocasiões, esse colégio foi a primeira estação observadora a detectar a presença de um novo satélite em órbita.



**Ouvindo estrelas**
Após a descoberta da presença de objetos que emitem sinais de rádio em galáxias remotas, foi construído este imenso radiotelescópio em Jodrell Bank, na Inglaterra. Radiotelescópios operam com enormes antenas e detectam objetos invisíveis até para os mais potentes telescópios ópticos.

# Passo a passo

**Na medição de um movimento linear ou angular por meio de um sensor óptico, a codificação binária não produz bons resultados — por isso foi estruturado o código Gray.**

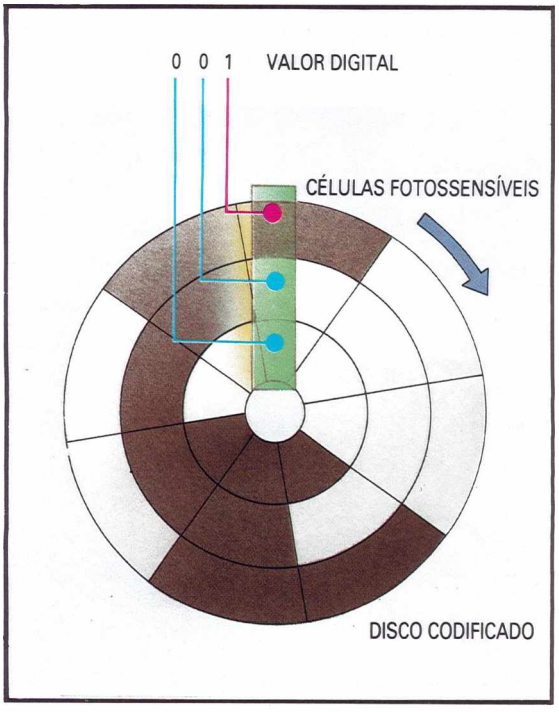
**Sistemas diferentes**
Abaixo, os decimais de 0 a 15, nos códigos binário e Gray.

| Decimal | Binário | Código Gray |
|---|---|---|
| 0 | 0000 | 0000 |
| 1 | 0001 | 0001 |
| 2 | 0010 | 0011 |
| 3 | 0011 | 0010 |
| 4 | 0100 | 0110 |
| 5 | 0101 | 0111 |
| 6 | 0110 | 0101 |
| 7 | 0111 | 0100 |
| 8 | 1000 | 1100 |
| 9 | 1001 | 1101 |
| 10 | 1010 | 1111 |
| 11 | 1011 | 1110 |
| 12 | 1100 | 1010 |
| 13 | 1101 | 1011 |
| 14 | 1110 | 1001 |
| 15 | 1111 | 1000 |

Existem muitas tarefas em que a posição física de um objeto móvel deve ser determinada com precisão e passada para o computador. Em robótica, por exemplo, o computador precisa ter conhecimento de todas as posições e orientações dos membros de um robô e, na fabricação de máquinas com controle computadorizado, a posição da mesa fresadora deve ser estabelecida com precisão. Mas, como é que uma posição pode ser convertida em valor binário, para ser processada pelo computador?

Um dos métodos inclui o uso de um sistema analógico. Consiste na ligação da peça móvel a uma resistência variável, e a voltagem resultante é passada para um conversor analógico-digital (ou diretamente para a porta analógica, caso seu computador tenha uma). Entretanto, este sistema não oferece muita precisão e as partes mecânicas estão sujeitas a desgaste.

A alternativa seria gravar um código binário na peça móvel, passando-o diretamente para o computador. Em geral, o código é gravado em forma de arranjos de blocos brancos e pretos, na parte superior da peça, e lido por um foco de luz que incide sobre um desses arranjos, em conjunto com uma linha de células fotossensíveis, cada uma responsável por um dos dígitos no arranjo binário. À medida que a peça se move, os diferentes arranjos vão passando sob as células luminosas, e isto produz um resultado binário, que define a posição do objeto. Além dos arranjos lineares, são também empregados padrões radiais, que codificam movimentos angulares, como o da articulação do braço de um robô.




**Ângulo de visão**
A posição angular de uma peça pode ser lida por um computador através de um disco com código gravado. Uma luz incide sobre a peça e uma linha de células fotossensíveis detecta o arranjo do código. O sinal digital que é produzido simultaneamente muda à medida que a peça se move. O inconveniente de se usar o sistema binário como código é que, se o disco parar na junção de dois valores, o resultado produzido pode não fazer sentido. O código Gray evita este problema.

Entretanto, quando a peça se movimenta de um código binário para outro, podem surgir alguns problemas, principalmente se a peça parar a meio caminho entre dois códigos. A precisão da gravação tem uma tolerância finita e, quando a peça pára numa junção entre dois códigos, as células luminosas fazem a leitura de qualquer um dos dois. Se a peça parar no ponto em que as células luminosas estiverem incidindo sobre a junção da posição binária 11 (1011) com a 12 (1100), por exemplo, então só o bit mais significativo (isto é, o 1 da esquerda) é que pode ser levado em consideração para fornecer a posição correta, enquanto as outras três células luminosas encontrarão valores conflitantes para ler. Em determinadas situações, todos os bits mudam, como no caso da junção do binário 7 (0111) com o 8 (1000); portanto, a menor falta de precisão na gravação pode produzir leituras incorretas em todas as células. O resultado seria um valor totalmente falso para a posição, e o computador não teria meios de saber que isso estava acontecendo.

Portanto, é preciso que haja um sistema de contagem alternativo, onde apenas 1 bit mude em cada movimento. Isto significa que só pode haver dúvidas sobre 1 bit em cada junção e o resultado pode apresentar um erro de, no máximo, uma posição. Este sistema alternativo é chamado código Gray, determinado pelas seguintes condições: passando de um valor para o seguinte, um único bit se modifica, e este deve ser sempre o que fica mais à direita, resultando na formação de um padrão único. Desta forma, se começarmos com 0000, como no sistema binário, o número 1 será representado por 0001. No entanto, para representar o número 2, devemos mudar o segundo bit da direita, obtendo 0011. Para o número 3, já é possível mudar o primeiro da direita e obter 0010. Observe como esta seqüência difere da binária para esses mesmos números: 0000, 0001, 0010, 0011.

O quadro mostra este processo ampliado até o equivalente ao número 15 do sistema decimal, e os números vêm acompanhados de seus respectivos equivalentes no sistema binário para simples referência. Como exercício, você pode calcular os valores do código Gray além deste ponto.

Computadores poderiam ser projetados para fazer cálculos matemáticos e funcionar internamente pelo código Gray, mas isto seria ineficiente e desnecessário. Assim, deve-se usar um meio de converter o código Gray no binário, o que pode ser feito com hardware ou software.

# JR Sysdata

## Um micro que pode ser expandido conforme as necessidades — desde as domésticas até as profissionais e comerciais.

Lançado em 1982 no mercado, o JR da Sysdata é uma máquina derivada do modelo I da linha TRS-80 — comercializada nos Estados Unidos pela Radio Shack, uma grande cadeia de lojas de equipamentos eletrônicos. Pesquisas feitas junto aos usuários brasileiros de micros revelaram posteriormente, contudo, que há entre eles uma preferência maior por máquinas compatíveis com o modelo III da mesma linha. Em conseqüência, a Sysdata passou, a partir de 1984, a dispor das duas compatibilidades: o JR I e o JR III.

São máquinas que, além do uso doméstico, servem a profissionais liberais e pequenas empresas. Têm também boa aceitação em escolas, por causa de sua resistência, flexibilidade de recursos e baixo custo do módulo básico.

As duas versões existentes podem ser fornecidas com um teclado tipo chiclete ou profissional e em versões de 16 a 64 Kbytes de RAM. Quem optar por uma versão mais simples e, portanto, mais barata, terá sempre a oportunidade de implementá-la de diversas maneiras — instalando um teclado profissional, adquirindo uma expansão de memória, uma interface para impressora ou uma placa controladora para permitir que o conjunto acione de uma até quatro unidades de discos flexíveis (disquetes) de 5 1/4 ou 8 polegadas.

E mais: ao teclado profissional podem ser acoplados um teclado numérico reduzido e um joystick para jogos.

Na parte posterior da máquina há uma via de expansão livre para outros acréscimos, como uma interface serial RS232 C para interligar o JR, através de um modem, a grandes computadores, a máquinas similares ou ainda para acessar o serviço de Videotexto da Telesp e o Projeto Cirandão da Embratel.




**Na linha TRS-80**
Os micros JR operam tanto no sistema DOS, dos TRS-80, quanto no CP/M, quando com memória expandida acima de 48 Kbytes, e com uma unidade de disquete. São indicados para uso doméstico, profissionais liberais e pequenas empresas. Dispõem de recursos flexíveis a um custo baixo.

Na placa controladora de unidade de disco, o JR I aceita a implementação do sistema operacional CP/M 2.2, que pode ser solicitado diretamente de dentro do sistema operacional DOS, original da máquina. Da mesma forma, é possível retornar ao DOS de dentro do CP/M, mediante um simples comando de teclado. Os JR III não necessitam de comando externo para identificar o sistema operacional adotado. Programam-se automaticamente, em função do disco introduzido no drive. Esse recurso permite que o JR III use vários programas aplicativos e utilitários para CP/M, como os consagrados DB II (gerenciador de bancos de dados) e o Wordstar (processador de palavras).

Com disco, o JR pode operar, em memória de 64 Kbytes, linguagens como COBOL, FORTH, FORTRAN, LISP, PASCAL e PILOT, além do BASIC compilado, interpretado e Assembler do seu microprocessador Z80.

A Sysdata, que em meados de 1984 já tinha mais de 2.000 máquinas instaladas, oferece seis meses de garantia para seus equipamentos e dá manutenção em sua própria fábrica em São Paulo, exceto nos casos de usuários de cidades distantes. Estes devem recorrer a um revendedor autorizado.

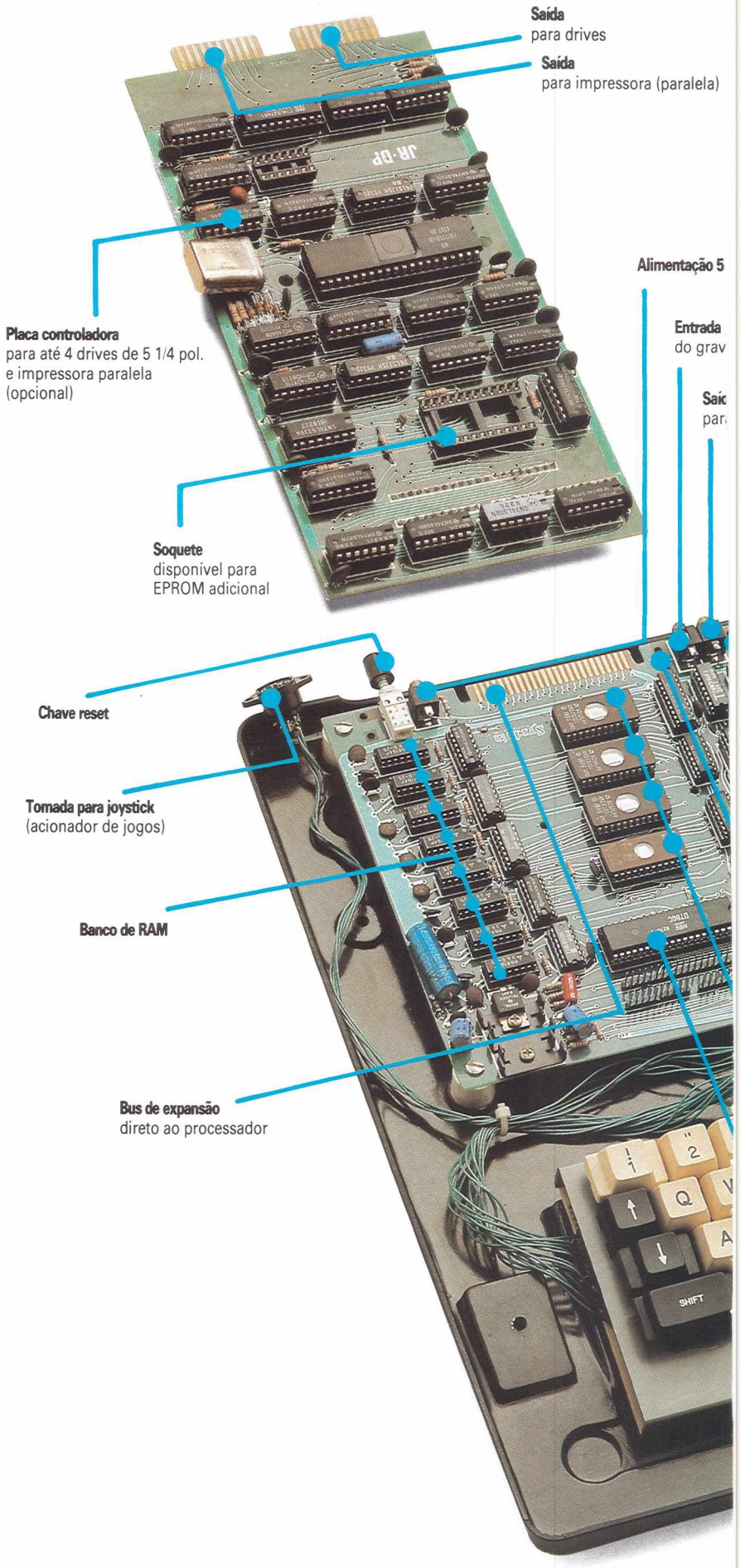


## O Sysdata III

Com uma RAM que vai de 48 a 128 Kbytes, o Sysdata III é um microcomputador muito mais poderoso do que o JR, e também mais caro. Lançado no mercado em 1984, é de concepção diferente da máquina anterior. Compatível com o modelo IV do TRS-80, opera no vídeo com 80 colunas e 24 linhas, ou 64 x 16. Isso lhe dá a mesma vantagem do JR III: roda integralmente as duas maiores bibliotecas de software para micros disponíveis no mercado internacional — a do sistema operacional DOS e a do CP/M.

Mas o seu melhor desempenho não se esgota aí, porque opera também com o CP/M 3.0, um sistema operacional que enxerga qualquer quantidade de memória disponível nessa máquina.

O teclado alfanumérico é de 69 teclas, incluindo um teclado numérico reduzido e quatro teclas programáveis de funções. Aceita até duas interfaces RS232 C (síncronas ou assíncronas). Além disso, vem com uma saída para impressora paralela e placa controladora para até 4 drives de 5 1/4 polegadas com dupla densidade, face simples ou dupla.



## JR Sysdata

**MICROPROCESSADOR**

Z80 A

**CLOCK**

2,0 MHz

**MEMÓRIA**

16 K até 64 K de RAM,
16 K de ROM

**VÍDEO**

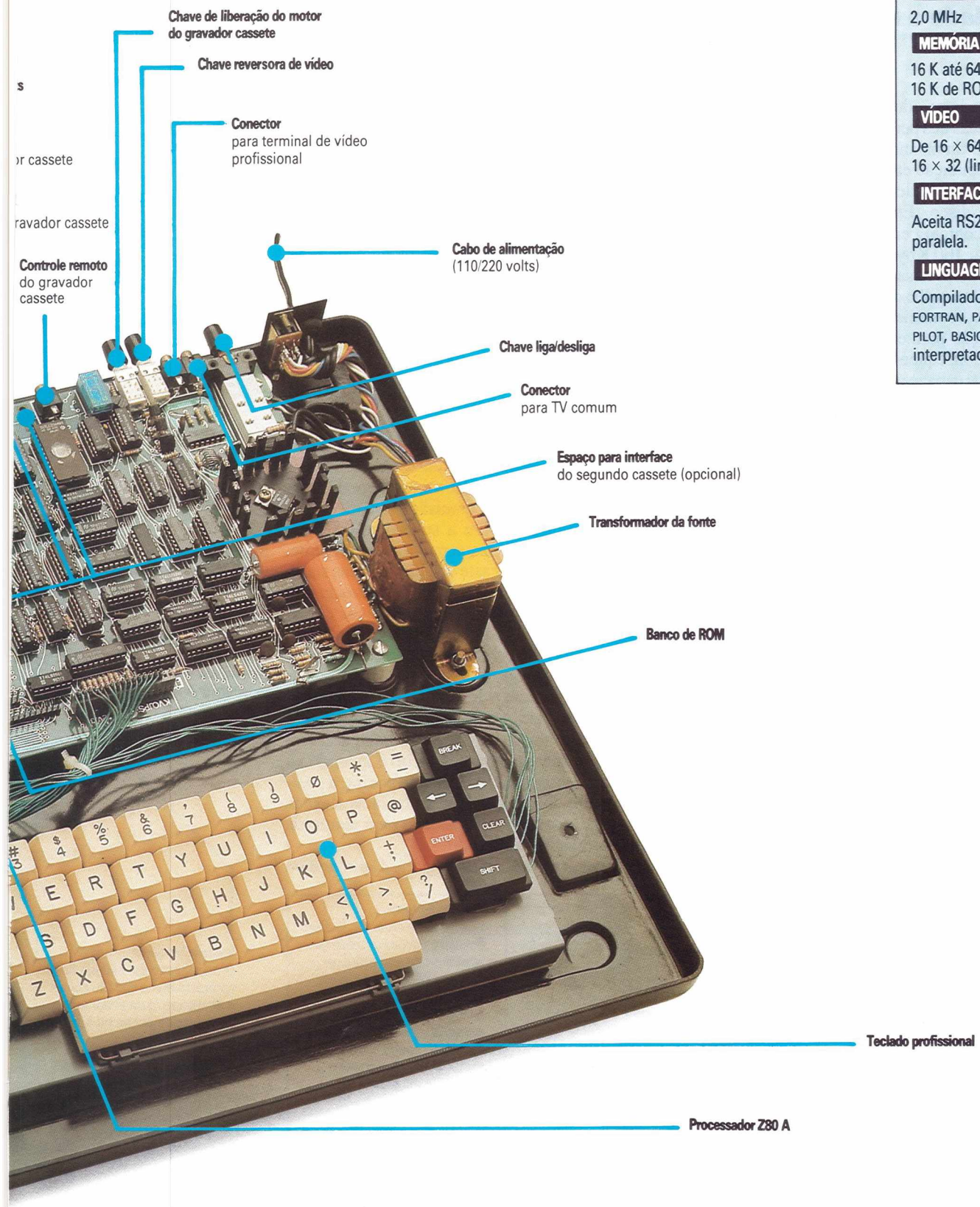
De 16 × 64 ou
16 × 32 (linhas × colunas)

**INTERFACE**

Aceita RS232 C ou saída paralela.

**LINGUAGENS**

Compiladores COBOL, FORTRAN, PASCAL, FORTH e PILOT, BASIC compilado, interpretado e Assembler.



# Plena carga

**Discos rígidos exigem condições de limpeza impecável para operar. O disco Winchester, em seu estojo lacrado, proporciona alta capacidade e rápido acesso ao usuário.**

Os microcomputadores domésticos assimilaram muitas características das pequenas máquinas de uso comercial nesses últimos anos, mas há uma área na tecnologia da computação em que eles permanecem relativamente pouco sofisticados: o espaço para armazenamento em disco. Enquanto um usuário de micro pessoal se contenta em ter um disco flexível capaz de comportar 100 Kbytes de dados, as máquinas de uso comercial precisam de espaço consideravelmente maior. Uma pilha de discos flexíveis, com toda a informação comercial espalhada entre eles, não constitui solução para esta grande demanda de armazenamento; conseqüentemente, foram produzidos discos rígidos capazes de armazenar quantidades bem maiores de dados.

O trabalho pioneiro com estes discos rígidos começou na IBM, na década de 60. Como os discos originais apresentassem uma capacidade de armazenamento de 30 megabytes em cada unidade, foram apelidados de 30/30 e daí, por analogia à carabina, chamados de discos Winchester.

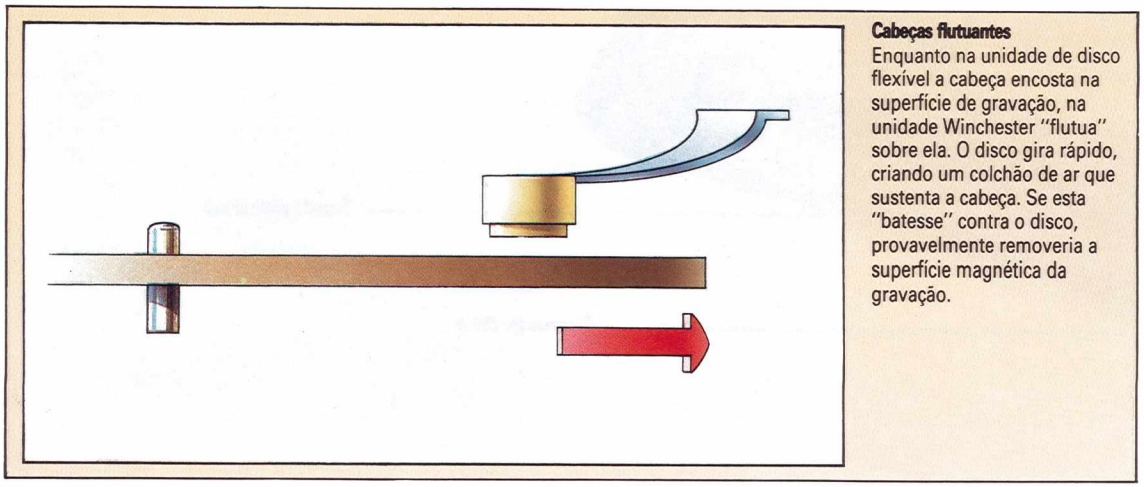
As unidades Winchester utilizam discos rígidos em lugar do plástico maleável usado nos discos flexíveis. Isso resulta em um aumento no número de trilhas de, no máximo, 96 trilhas por polegada (tpi) em discos flexíveis, para centenas de tpis em discos Winchester. A crescente sofisticação da tecnologia de discos tornou comum o armazenamento de 5, 10 ou até 20 megabytes numa caixa do mesmo tamanho de uma unidade de disco flexível de 5 1/4 pol.

Usuários de microcomputadores estão se beneficiando com a tendência de usar discos rígidos, pois já se encontram disponíveis os microdiscos flexíveis Sony 3 1/2 pol. e Hitachi 3 pol. São discos semi-rígidos, com capacidade para armazenar tanta informação quanto qualquer disco flexível de 5 1/4 pol. Micros ingleses, como o BBC Micro e o Oric-1, foram os primeiros a ter estes dispositivos em forma de acessórios, enquanto computadores como o Apricot da ACT já vêm equipados com eles.

Este aumento da densidade de armazenamento, no entanto, criou outros problemas. A precisão exigida pelo mecanismo de posicionamento da cabeça, por exemplo, pedia um modo completamente novo para movimentá-la. A solução deste problema foi encontrada na indústria de áudio: as cabeças Winchester são freqüentemente colocadas em posição por uma bobina eletromagnética do tipo usado em alto-falantes. Passando-se uma corrente elétrica por uma bobina, cria-se um campo magnético que, por sua vez, faz com que uma tomada de ferro especial, localizada no centro da bobina, se movimente numa distância exata. Ligando-se a cabeça à ponta da tomada (isolada de quaisquer efeitos magnéticos, é claro), ela pode se movimentar sobre a superfície do disco com muita rapidez e precisão.

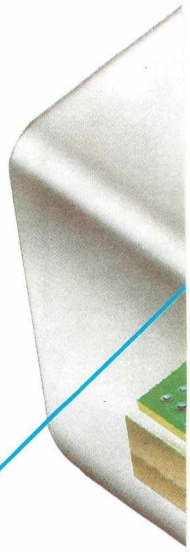
A cabeça "flutua" sobre a superfície do disco apoiada num colchão de ar, sem a tocar realmente. Isto reduz bastante o desgaste dos discos, mas significa que eles devem ser conservados em caixas hermeticamente fechadas para evitar problemas causados pela poeira e outros corpos estranhos. Em geral, isso significa que o disco é fixado dentro da unidade e não pode ser retirado, embora já existam discos Winchester em cartuchos removíveis. Esses cartuchos apresentam geralmente vedação automática e só abrem para permitir que a cabeça tenha acesso à superfície do disco quando o cartucho é inserido na unidade. Para evitar que entre alguma poeira, a pressão do ar dentro do cartucho é mantida mais elevada que a externa, e todo o ar bombeado para dentro da unidade é previamente filtrado.

Outra vantagem dos discos rígidos é que podem



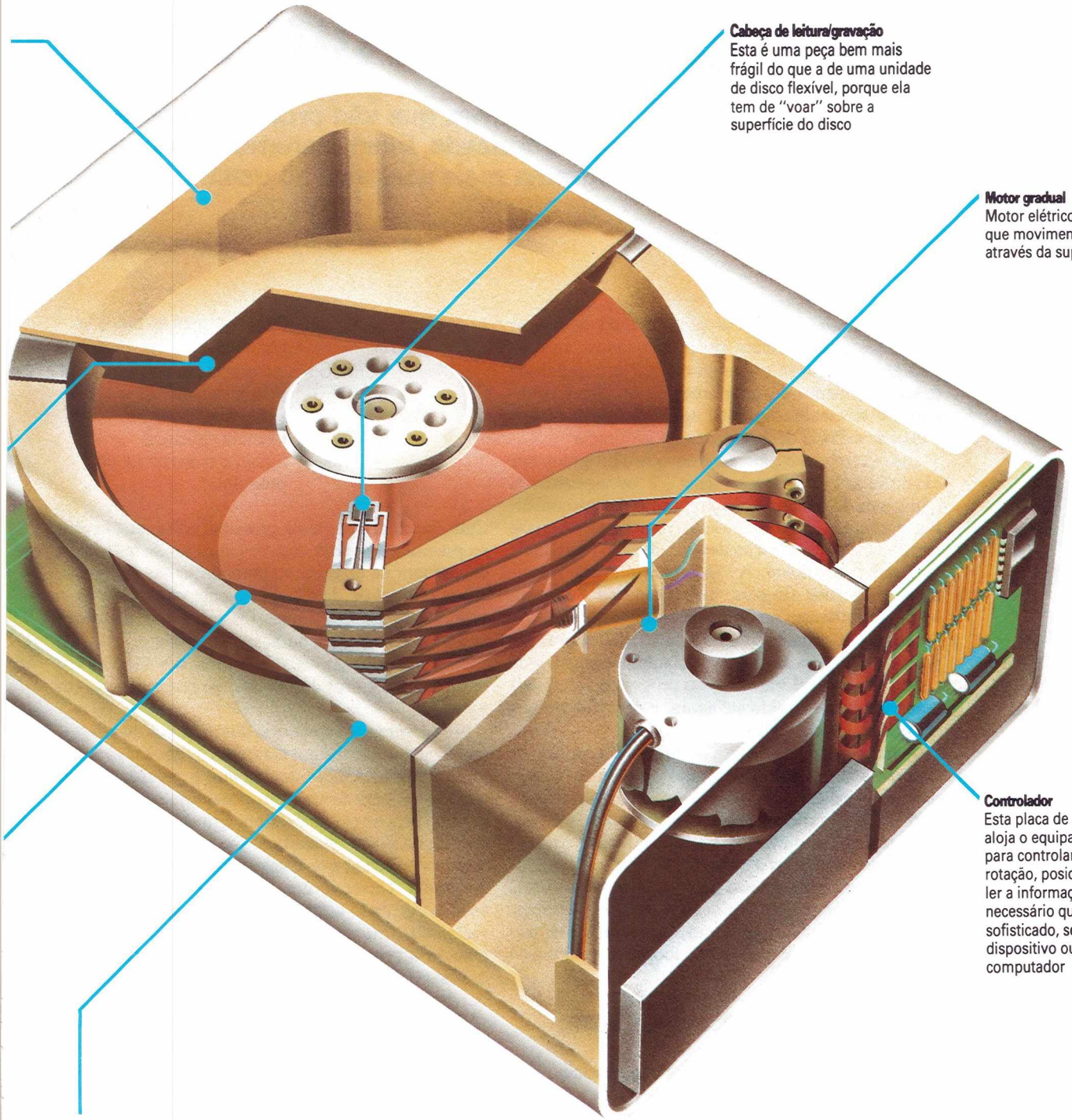
**Cabeças flutuantes**
Enquanto na unidade de disco flexível a cabeça encosta na superfície de gravação, na unidade Winchester "flutua" sobre ela. O disco gira rápido, criando um colchão de ar que sustenta a cabeça. Se esta "batesse" contra o disco, provavelmente removeria a superfície magnética da gravação.

**Estojo**
A maioria das unidades Winchester estão contidas em um estojo feito de uma liga de metal fundido, parcialmente responsável pelo seu peso. O estojo é necessário para manter os componentes alinhados com precisão



**Discos**
As unidades Winchester de maior capacidade simplesmente apresentam mais discos no mesmo eixo. A unidade da ilustração tem cinco discos, mas a maioria tem dois ou três. As cabeças de leitura/gravação são interligadas, de forma que só um bloco pode ser lido por vez

**Vedação hermética**
O mecanismo da unidade fica completamente vedado contra a atmosfera, para evitar que partículas de poeira ou fumaça estraguem a cabeça



**Cabeça de leitura/gravação**
Esta é uma peça bem mais frágil do que a de uma unidade de disco flexível, porque ela tem de "voar" sobre a superfície do disco

**Motor gradual**
Motor elétrico de alta precisão que movimenta a cabeça através da superfície do disco

**Controlador**
Esta placa de circuito impresso aloja o equipamento eletrônico para controlar a velocidade de rotação, posicionar a cabeça e ler a informação. Além disso, é necessário que haja um DOS sofisticado, seja em outro dispositivo ou na memória do computador

**Motor**
Um motor DC (corrente contínua) e um pequeno gerador são montados no mesmo eixo que gira o disco. A saída do gerador é uma medida da velocidade do motor e alimenta um circuito de controle especial. É dessa forma que se determina a velocidade com tamanha precisão

ser usados discos múltiplos. Uma unidade de disco Winchester de 10 megabytes pode ser constituída de dois discos de 5 megabytes colocados na mesma caixa. Consegue-se controlar todo esse espaço de armazenamento dividindo-o em um grande número de seções. Para manter compatibilidade com o software existente, próprio para trabalhar só com discos flexíveis, essas seções geralmente se aproximam da capacidade de um disco flexível normal. Uma unidade de disco Winchester se parece com um conjunto de unidades separadas de discos flexíveis.

Quando um disco Winchester é formatado (isto é, as trilhas e setores são demarcados), o DOS (ver p. 324) deve ser capaz de pular os "setores defeituosos", que apresentam irregularidades na superfície magnética de gravação. Num disco flexível, um setor defeituoso resulta na inutilização de um disco inteiro, enquanto num Winchester o programa de formatação simplesmente registra que o setor está inutilizado, bloqueando-o para evitar que seja usado. Afinal, com 5 milhões de bytes de espaço, quem vai sentir falta de algumas centenas?

Acompanhando os avanços dos outros setores da indústria de computadores, o Winchester está diminuindo de tamanho e já existe o disco micro-Winchester de 5 megabytes. Com o desenvolvimento de unidades para armazenamento de discos, um micro pessoal por menos de 1.000 dólares e com capacidade para armazenamento de 10 megabytes de dados não está tão longe como se imagina.

# Montagem de programas

**Podemos agora unir os subprogramas que vão processar nossa agenda de endereços computadorizada. Examinaremos também métodos para tornar o programa mais acessível ao usuário.**

Embora muitos detalhes do programa da agenda de endereços ainda tenham de ser estudados, a estrutura geral já está se tornando clara. Neste ponto do desenvolvimento de programas de qualquer tamanho, é conveniente desenhar um diagrama de blocos e avaliar o fluxo de atividades no programa.

Este é também o momento em que o programador examina a "interface humana" e os aspectos da "imagem do usuário" do programa. Esses conceitos e práticas, embora muito importantes, quase nunca recebem a atenção devida, mesmo no caso de software profissional. A "interface humana", definida em termos simples, corresponde à "ergonomia" do software, ou seja, sua facilidade de uso. A "imagem do usuário" vincula-se ao modo como ele percebe o programa em uso. Examinaremos esses conceitos com relação a nosso programa, da forma como foi desenvolvido até agora, e determinaremos até onde se pode complementá-los.

A lista abaixo apresenta os principais blocos do programa já examinados. Como convenção, e apenas para manter certa ordem no procedimento de verificação, empregaremos nomes com seis caracteres para procedimentos ou sub-rotinas, com oito caracteres (inclusive o caractere $) para matrizes alfanuméricas, com quatro caracteres para variáveis numéricas simples, e com cinco caracteres (inclusive $) para variáveis alfanuméricas simples de uso geral. As variáveis locais (nos loops, por exemplo) terão letras isoladas.

**BLOCOS DE PROGRAMA PRINCIPAL**

| | | |
|---|---|---|
| INICIL | CRIMAT | (cria matrizes e inicializa variáveis) |
| | LERARQ | (lê arquivos em fita ou disco) |
| | ESTFLG | (estabelece flags e altera variáveis) |
| SAUDAR | | (imprime mensagem de saudação) |
| ESCLHA | MENESC | (imprime menu de escolhas) |
| | ATRESC | (atribui a escolha à variável ESCL) |
| EXECUT | ENCREG | (encontra e imprime um registro) |
| | ENCNOM | (encontra nomes a partir de nomes incompletos) |
| | ENCCID | (encontra o registro de determinada cidade) |
| | ENCINI | (encontra nomes a partir de iniciais) |
| | LISREG | (faz a listagem de todos os registros) |
| | ACRREG | (acrescenta novos registros) |
| | MODREG | (modifica registros já existentes) |
| | ELMREG | (elimina registros) |
| | SAIPRG | (grava arquivos e sai do programa) |

Cada bloco do programa principal na segunda coluna precisa ser subdividido em duas unidades e estas necessitam de elaboração complementar até que se atinja o nível de detalhe suficiente para escrever o próprio código em BASIC. Os processos incluídos nessa forma de "elaboração por estágios" foram exemplificados, no caso de muitos dos blocos, em unidades anteriores deste curso de programação em BASIC.

Admitamos que a maioria dos módulos do programa ou todos eles foram elaborados, codificados em BASIC e testados individualmente. É preciso, agora, unificá-los para formar um programa completo. O melhor processo consiste em gravar cada módulo em fita ou disco, atribuindo-lhe o mesmo nome de arquivo utilizado nas notas de desenvolvimento do programa. Dessa forma, o bloco ACRREG pode ser escrito e testado até onde for possível e então gravado sob o nome de arquivo ACRREG. Em geral, quando o programa é carregado em fita ou disco, usamos o comando LOAD, seguido do nome de arquivo, como acontece na instrução LOAD "ACRREG". No entanto, tal procedimento resulta na limpeza total dos conteúdos da memória; assim, se carregarmos o bloco ACRREG e logo após o bloco MODREG, todo o programa ACRREG será apagado.

Em boa parte, porém, o problema pode ser solucionado. O comando MERGE carrega programas procedentes de fitas ou discos sem apagar programas já armazenados na memória. Mas há uma condição importante. Se quaisquer dos números de linha no programa que recebeu a instrução MERGE coincidirem com os já presentes na memória, os novos números de linha serão sobrepostos aos antigos, gerando caos. Versões do BASIC com o comando RENUM podem contornar esse inconveniente pela renumeração das linhas no módulo de programa, antes de realizar seu armazenamento. Assim, quando elas se unificarem, não haverá conflito.

Muitas versões do BASIC para computadores pessoais não possuem o comando RENUM, tornando-se necessário um planejamento cuidadoso dos números de linha desde o início. Quando se tiver elaborado

um mapa completo de todos os módulos do programa principal (como fizemos parcialmente na lista), pode-se atribuir números de linha de início para cada bloco. Partes do programa passíveis de grandes modificações — como o programa principal ou as unidades de manipulação de arquivos — devem ser numeradas a intervalos de 50 ou mesmo 100 unidades. Assim, haverá espaço para eventuais inclusões. Os intervalos dos números de linha em módulos de programa menos propensos a alterações, como a rotina SAUDAR, podem ser de 10 unidades. A inclusão, no programa, de numerosas instruções REM em branco facilita sua leitura e possibilita o posterior acréscimo de instruções ou de chamadas de sub-rotinas adicionais. Se o BASIC de seu micro não possuir o comando MERGE, digite os vários módulos à medida que se desenvolvem e grave-os juntos.

Na lista, os blocos de programa foram unificados como um "processamento experimental", para ilustrar os erros em que se pode incidir ao fazer a abordagem do tipo "experimente e veja" que o BASIC encoraja. Nosso programa não funcionaria a contento porque não se elaborou o fluxo de controle do programa com atenção suficiente. Seria trabalho inútil digitar todo um programa no computador só para verificar que não funciona; mas, se você gravou as rotinas de unidades anteriores do curso, e se o BASIC de seu micro posssui o comando RENUM, pode experimentar a renumeração e a seguir unificá-la para produzir uma listagem semelhante.

O primeiro bloco do programa principal é INICIL, com a finalidade de inicializar variáveis, dimensionar matrizes, ler arquivos, atribuir dados às matrizes, estabelecer flags etc.

A sub-rotina *INICIL* está subdividida na sub-rotina *CRIMAT* (para criar matrizes); na sub-rotina *LERARQ* (para ler arquivos e atribuir os dados às matrizes correspondentes) e na sub-rotina *ESTFLG* (para estabelecer flags etc.).

O programa seguirá então para *SAUDAR*, uma sub-rotina que imprime mensagem de saudação na tela. A última parte faz com que o programa se interrompa até que o usuário pressione a barra de espaço para seu prosseguimento.

O programa então continua até a rotina *ESCLHA*, subdividida em duas partes: uma apresenta repertório (menu) de opções oferecidas pelo programa da agenda de endereços; a segunda recebe a escolha fornecida por meio do teclado e atribui o valor (numérico) a uma variável denominada ESCL.

O valor dessa variável é utilizado pelo bloco seguinte do programa, EXECUT, para selecionar um entre nove outros blocos de programa. Todos eles, exceto o SAIPRG, deverão retornar à sub-rotina *ESCLHA* após sua execução, de forma que o usuário terá oportunidade de fazer outra escolha. Tal procedimento não será necessário se optar pelo bloco 9 (SAIPRG), pois este tem como finalidade o encerramento da operação do programa.

Do modo como se apresenta, o principal problema corresponde à incorreção do fluxo de controle. A sub-rotina INICIL exige a leitura de um arquivo da memória de armazenamento em massa, exista ou não esse arquivo. Se o programa estiver sendo processado pela primeira vez, não haverá dados já fornecidos — e nem arquivos de dados na fita ou no disco. Qualquer tentativa de abrir e ler um arquivo não existente resultará numa mensagem de erro e o programa não funcionará.

Para contornar essa dificuldade, chama-se a rotina *LERARQ* por meio de apenas um dos módulos EXECUT, e depois uma única vez, sempre que o programa for processado. Isso indica que deve haver uma flag ARQV com o valor inicial 0, que assumirá o valor 1, após a leitura do arquivo. Tendo assumido o valor 1, a flag não admitirá outras tentativas de leitura do arquivo. Desse modo, a sub-rotina ACRREG sempre examina as matrizes para localizar o primeiro elemento vazio e aí registra os dados. É bem provável que esse registro não esteja na seqüência adequada de ordenação; assim, deve haver uma flag RMOD que assume o valor 1 durante a execução. A flag RMOD também deve assumir o valor 1, se as rotinas MODREG ou ELMREG forem executadas. Você pode desenvolver o código relevante para obter esse resultado, ou, se só quiser processar o programa, alterar a linha 1310 para RETURN.

Procedimentos como a inclusão, a eliminação ou a modificação de registros indicam que a seqüência deles está fora de ordem; assim, qualquer módulo (ENCREG, por exemplo) deve primeiro examinar a flag RMOD para verificar se ocorreram alterações. Em caso afirmativo, pode-se insistir numa ordenação, antes de efetuar uma busca. Ou, então, admitir um procedimento ineficiente de busca no arquivo. A sub-rotina SAIPRG examina de modo automático a flag RMOD e chama a rotina de ordenação, se esta receber uma atribuição (do valor 1) antes de gravar os dados no arquivo em fita ou disco.

Os aspectos da "interface humana" dos programas mencionados subdividem-se nestas categorias:

Interface do usuário
Imagem do usuário
Recuperação de erros
Segurança
Adaptabilidade

A interface do usuário corresponde ao modo como este se comunica com o programa. Optamos pelo uso de menus em todas as ocasiões (em vez de comandos). Muitas pessoas preferem comandos, mas a questão importante é que, não importa a forma de comunicação empregada, ela tem de ser coerente. Não convém que comandos semelhantes realizem tarefas diferentes em partes diferentes do programa. Se isso ocorrer, é necessário que o usuário leia com atenção cada menu, antes de realizar a escolha, e evite "reflexos".

Do modo como se apresenta, nosso programa mostra-se frágil no seguinte aspecto: o pressionamento da barra de espaço encerra a mensagem de saudação; o menu de opções é automaticamente encerrado pela digitação de uma das teclas numéricas de 1 a 9; e o fornecimento de dados na sub-rotina ACRREG encerra-se (em cada campo) com o pressionar da tecla RETURN. Admite-se uma incoerência como essa em programas de "amadores", mas ela é inaceitável no caso de software profissional.

A "imagem do usuário" relaciona-se ao modo como ele percebe o funcionamento do programa e sofre a influência da qualidade da interface do usuário. O operador não tem ciência da maior parte das

operações que se processam no interior do computador. O único modo pelo qual pode avaliar o que se passa no programa é proporcionado pela entrada visual recebida na tela em resposta às entradas de dados fornecidos pelo teclado. A "imagem do usuário" que esperamos de nosso programa da agenda de endereços computadorizada corresponde à de um livrinho de endereços fisicamente real. De modo semelhante, a "imagem do usuário" de um programa processador de palavras é a de uma folha de "papel" (na tela), na qual digitamos. Nesse caso, teoricamente, os tipos em negrito aparecem em negrito na tela, os tipos sublinhados mostram-se sublinhados e os tipos justificados (dispostos de modo que formem a margem direita em linha reta) surgem desse modo na tela.

A "imagem do usuário" quase nunca se mostra perfeita — nenhum livrinho de endereços real aguarda o usuário "PRESSIONAR 1" para localizar um nome. No entanto, a boa imagem do usuário implica a elaboração bem feita de esquemas na tela e um padrão coerente de operações. As indicações devem aparecer sempre na mesma posição na tela (alguns programas processadores de palavras, por exemplo, contêm algumas indicações na linha superior da tela e outras na inferior, de modo aparentemente aleatório). Programas que correspondem à boa imagem do usuário também podem informá-lo, a qualquer momento, em que posição do programa ele se encontra. Se você está no modo ACRREG, precisa de uma imagem sempre visível que lhe indique isso. Se você acaba de fornecer um campo (para acrescentar novo registro), convém que exista uma mensagem dizendo, por exemplo, PRESSIONAR A TECLA DE RETURN, SE A ENTRADA ESTIVER CORRETA, CASO CONTRÁRIO, PRESSIONAR ESCAPE. (Isso nos remete à importante questão da recuperação e indicação de erros, discutida mais adiante.)

Teoricamente, toda a formatação deve aparecer na tela de modo que, por exemplo, o registro apresentado tenha o mesmo formato do registro da impressora. Muitas unidades de software à venda incluem "menus auxiliares" que indicam o procedimento a ser adotado a seguir, caso você não se sinta seguro.

A "imagem do usuário" de um programa melhora quando é concreta — um pedaço de papel ou um cartão de fichário, por exemplo — e não abstrata, como "subarquivos", "memórias intermediárias" (buffers) etc. Muitos programas de bancos de dados à venda apresentam problemas com relação a esse fator; o usuário deve ter em mente que certas unidades de dados estão em subarquivos ou em campos auxiliares temporários, o que tende a dificultar o uso de programas.

A recuperação de erros é outro item importante. O que acontecerá, por exemplo, se você fornecer o nome de alguém, mas perceber, logo depois, que cometeu um erro de digitação? Você deve prosseguir e chamar a sub-rotina MODREG para corrigi-lo? Ou o programa lhe dá a possibilidade de "resolver" antes de seguir adiante? A maioria das versões do BASIC indica a existência de erro na entrada do programa, no momento em que ele é cometido ou quando se processa o programa. Mas isso não faz parte da "interface do usuário". O BASIC inclui uma série de mensagens que fazem uma segunda indicação ao usuário para o fornecimento dos dados corretos, quando se realiza uma entrada incorreta (por exemplo, a indicação REDO quando é fornecido um dado inadequado à instrução INPUT).

A manipulação de erros apresenta dois aspectos — o relato de erros e sua recuperação. Um programa muito conhecido de processamento de palavras tem boa capacidade de relato de erros, mas pouca de recuperação; se você criar um documento extenso e tentar gravá-lo em disco que já esteja repleto, o programa fornece uma mensagem muito útil: DISK SPACE EXHAUSTED (CAPACIDADE DO DISCO ESGOTADA). Mas isso não basta para a recuperação do erro; não será possível a formatação de novo disco, sem a destruição do texto que você levou tanto tempo para digitar.

Qualquer operação realizada pelo usuário que resulte em perda de dados (MODREG, por exemplo) deve ser examinada antes da execução. Convém que existam perguntas do tipo ESTE PROCEDIMENTO FARÁ DESTRUIR O REGISTRO, TEM CERTEZA DE QUERÊ-LO? (S/N). No programa processador de palavras uma mensagem equivalente seria: O COMANDO "SAVE" NÃO MANTERÁ CÓPIA DO DOCUMENTO ANTERIOR. ESTÁ OK DESSE MODO? (S/N).

Leve em conta a manipulação de erros (detecção e relato) na elaboração do programa, sempre que houver possibilidade de fornecimento incorreto de dados; escolha errada do menu; digitação de comandos impróprios; e dados passíveis de mudança ou gravação, sobretudo se o procedimento de gravação implicar registros sobrepostos a dados anteriores.

Dê atenção à segurança: o que acontecerá ao programa ou aos dados se ocorrer um erro irremediável (como a interrupção do fornecimento de energia elétrica)? O programador precisa considerar o limite admissível de perda dos dados e criar métodos que possibilitem recuperá-los em grande parte ou utilizar os restantes. Um programa processador de palavras mais sofisticado inclui outro, chamado RECUPERAÇÃO, de modo que, se ocorrer uma falha irremediável (como a interrupção do fornecimento de energia ou o desligamento do computador antes da gravação do documento), quase nada se perde. Tais técnicas avançadas de programação estão além dos objetivos deste curso. De todo modo, a questão consiste em tornar seguros os programas pela previsão de eventuais falhas graves (com as quais se possa lidar), e desenvolver rotinas para dar conta delas.

A flexibilidade — a rapidez com que o programa pode se adaptar às necessidades do usuário — também é importante. Já tratamos desta questão algumas vezes. Em nível mais simples, consiste no empenho de sempre deixar bastante espaço entre os números de linha (em BASIC) e incorporar instruções REM suficientes para posterior preenchimento com instruções GOSUB, se necessário. Ao elaborar matrizes, pelo menos uma delas deve ser redundante, de modo a permitir ampliação futura. Uma regra básica da técnica de desenvolvimento de programas diz que não se prevêem as necessidades futuras. Mas não resta dúvida de que um bom programa sempre pode ser aperfeiçoado — e seu aprimoramento provavelmente significará a inclusão de novas instruções.

## A propósito...

| Comando/Instrução/Função | Ação/Resultado | MS BASIC | Applesoft | TRS-80 | Sinclair |
|---|---|---|---|---|---|
| HCOLOR=expressão | Fornece a cor para desenhos em gráficos de alta resolução | | + | | |
| HEX$(n) | Converte n em hexadecimal | + | | | |
| HGR | Comando para gráficos de alta resolução | + | + | | |
| HIMEM | Gera limite superior da memória para programa BASIC | | + | | |
| HLIN x,y AT z | Traça uma linha a partir de x para y na posição z da tela | + | + | | |
| HOME | Limpa a tela | + | + | | |
| HPLOT x,y | Desenha ponto ou linha em gráfico de alta resolução | + | + | | |
| HTAB y | Posiciona o cursor numa coluna da linha atual | | + | | |
| IF expr THEN c11 ELSE c12 | Executa c11 se expr for verdadeira. Se não, executa c12 | + | + | + | |
| IF expr THEN instrução | Se expr for verdadeira, então expressão será executada | | | | + |
| IN#slot | Indica o número do slot para as próximas entradas | | + | | |
| INIT "nomearquivo" | Inicializa um disco com o arquivo da memória | | + | | |
| INKEY$ | Lê um caractere do teclado | + | | + | + |
| INP (porta) | Fornece o valor atual da porta especificada | | | + | |
| INPUT variável | Pára e aguarda uma entrada de dados | | | | + |
| INPUT"mensagem"; lista de var | Lê variáveis a partir do teclado | + | + | + | |
| INPUT#-1,lista de variáveis | Carrega dados provenientes de uma fita cassete | | | + | |
| INPUT"arquivo",lista de variáveis | Lê dados de um arquivo | + | | | |
| INPUT$(n,"arquivo") | Lê n caracteres de um arquivo | + | | | |
| INSTR(n,x$,y$) | Procura a 1.ª ocorrência de y$ em x$ e fornece a posição | + | | | |

```
10 REM "PRGPRN"
20 REM *INICIL*
30 GOSUB 1000
40 REM *SAUDAR*
50 GOSUB 3000
60 REM *ESCLHA*
70 GOSUB 3500
80 REM *EXECUT*
90 GOSUB 4000
100 END
1000 REM SUB-ROTINA *INICIL*
1010 GOSUB 1100: REM SUB-ROTINA *CRIMAT* (CRIA MATRIZES)
1020 GOSUB 1300: REM SUB-ROTINA *LERARQ* (LE ARQUIVOS)
1030 GOSUB 1380: REM SUB-ROTINA *ESTFLG* (ESTABELECE FLAGS)
1040 REM
1050 REM
1060 REM
1070 REM
1080 REM
1090 RETURN
1100 REM SUB-ROTINA *CRIMAT* (CRIA MATRIZES)
1110 DIM CAMPNOM$(50)
1120 DIM CAMPMOD$(50)
1130 DIM CAMPRUA$(50)
1140 DIM CAMPCID$(50)
1150 DIM CAMPEST$(50)
1160 DIM CAMPTEL$(50)
1170 DIM CAMPIND$(50)
1180 REM
1190 REM
1200 REM
1210 LET TAMA = 0
1220 LET RMOD = 0
1230 LET GRAV = 0
1240 LET CORR = 0
1250 REM
1260 REM
1270 REM
1280 REM
1290 RETURN
1300 REM SUB-ROTINA *LERARQ*
1310 ON ERROR GOTO 1370
1320 OPEN "I", #1, "AGEN.DAD"
1330 FOR L = 1 TO 50
1340 INPUT# CAMPNOM$(L), CAMPRUA$(L), CAMPCID$(L),
     CAMPEST$(L), CAMPTEL$(L)
1350 NEXT L
1360 CLOSE #1
1370 RETURN
1380 REM ROTINA FICTICIA *ESTFLG*
1390 RETURN
3000 REM SUB-ROTINA *SAUDAR*
3010 PRINT
3020 PRINT
3030 PRINT
3040 PRINT
3050 PRINT TAB(14); "*BENVINDO AO*"
3060 PRINT TAB(4); "*MICROCOMPUTADOR — CURSO BASICO*"
3070 PRINT TAB(1); "*AGENDA DE ENDERECOS COMPUTADORIZADA*"
3080 PRINT
3090 PRINT " (PRESSIONAR BARRA DE ESPACO PARA CONTINUAR) "
3100 FOR L = 1 TO 1
3110 IF INKEY$ <> " " THEN L = 0
3120 NEXT L
3130 PRINT CHR$(12)
3140 RETURN
3500 REM SUB-ROTINA *ESCLHA*
3510 REM
3520 REM "MENESC"
3530 PRINT CHR$(12)
3540 PRINT "ESCOLHER UM DOS SEGUINTES:"
3550 PRINT
3560 PRINT
3570 PRINT
3580 PRINT "1. ENCONTRAR UM REGISTRO (PELO NOME)"
3590 PRINT "2. ENCONTRAR NOMES (POR TRECHO DE UM NOME)"
3600 PRINT "3. ENCONTRAR REGISTROS (DE UMA CIDADE)"
3610 PRINT "4. ENCONTRAR REGISTROS (DE UMA INICIAL)"
3620 PRINT "5. LISTAR TODOS OS REGISTROS"
3630 PRINT "6. ACRESCENTAR UM REGISTRO"
3640 PRINT "7. MODIFICAR UM REGISTRO"
3650 PRINT "8. ELIMINAR UM REGISTRO"
3660 PRINT "9. SAIR E GRAVAR"
3670 PRINT
3680 PRINT
3690 REM "ATRESC"
3700 REM
3710 LET L = 0
3720 LET I = 0
3730 FOR L = 1 TO 1
3740 PRINT "ESCOLHER DE 1 A 9"
3750 FOR I = 1 TO 1
3760 LET A$ = INKEY$
3770 IF A$ = " " THEN I = 0
3780 NEXT I
3790 LET ESCL = VAL(A$)
3800 IF ESCL <1 THEN L = 0
3810 IF ESCL >9 THEN L = 0
3820 NEXT L
3830 RETURN
4000 REM SUB-ROTINA *EXECUT*
4010 ON ESCL GOSUB 310,330,350,370,390,410,430,450,470
4020 RETURN
9000 REM SUB-ROTINA *ACRREG*
9010 PRINT CHR$(12)
9020 INPUT "FORNECER O NOME" ; CAMPNOM$(TAMA)
9030 INPUT "FORNECER A RUA" ; CAMPRUA$(TAMA)
9040 INPUT "FORNECER A CIDADE" ; CAMPCID$(TAMA)
9050 INPUT "FORNECER O ESTADO" ; CAMPEST$(TAMA)
9060 INPUT "FORNECER O NUMERO TELEFONICO" ; CAMPTEL$(TAMA)
9070 LET RMOD = 1
9080 LET CAMPIND$(TAMA) = STR$(TAMA)
9090 GOSUB *NOMMOD*
9100 RETURN
```

# Atendendo pacientes

**Além de cuidarem da administração dos consultórios médicos, os micros começam a ser usados como auxiliares de diagnóstico.**

Os microcomputadores começam a ser utilizados nos consultórios médicos. Embora essa prática ainda se mostre tímida, considerando-se o pequeno número de consultórios que dela fazem uso, a qualidade dos serviços prestados, o campo ao qual ela atende e as facilidades que propicia deixam prever seu rápido crescimento.

E isso fica evidente quando se analisam as tarefas usualmente desempenhadas num consultório. Evitar, por exemplo, que dois pacientes tenham suas consultas marcadas no mesmo horário é algo extremamente simples (embora por vezes o fenômeno ocorra), mas a tarefa se complica se for preciso conciliar o horário de um paciente destinado a uma primeira consulta com os curtos intervalos requeridos ao retorno para simples acompanhamento. Somando-se a isso as desistências de horários, o preenchimento de novos, a previsibilidade do tempo a ser gasto em cada caso — em função de sua maior ou menor gravidade —, já se esboça um campo rudimentar, mas propício, à organização pela informática. Esse campo torna-se muito mais amplo se acrescentarmos questões ligadas aos contatos com pacientes, laboratórios e hospitais; a interpretação e estocagem de dados e exames; a confrontação de dados obtidos em diversas datas.

Os microcomputadores passam a se mostrar úteis a partir daí, e já chegam a ser empregados para facilitar a solução de questões mais sofisticadas, como a própria obtenção (ou fornecimento) de dados do e para o cliente, é a interpretação mais ágil de determinados exames solicitados pelo médico.

Em julho de 1983, cerca de 5% dos consultórios médicos nos Estados Unidos utilizavam o micro. E uma série de operações passou a ser efetuada pela máquina, com resultados extremamente satisfatórios.

## Consulta ginecológica

Um exemplo curioso é o de alguns ginecologistas e obstetras que colocaram micros à disposição das pacientes na sala de espera, para consulta direta. Neste caso, o computador opera a partir de três programas distintos. No primeiro, oferece respostas à paciente a respeito de problemas ginecológicos e obstétricos mais elementares, sem necessidade da interferência do médico. No segundo, o micro fornece informações a respeito de métodos de controle da natalidade, indicando riscos e vantagens de diferentes anticoncepcionais. Finalmente, mediante consulta ao computador, a paciente será informada sobre a utili-




**Saúde pública**
O microcomputador permite manter arquivos detalhados sobre pacientes e o cruzamento das diversas informações coletadas facilita a elaboração de estatísticas, fundamentais para os serviços de saúde pública. Em São Paulo, o cirurgião Azzo Vidman consegue, por exemplo, saber imediatamente qual a incidência de determinada doença.

zação de medicamentos durante a gravidez: quais e em que períodos poderão causar-lhe dano ou ao feto.

Comprovou-se que essa prática, além da economia de tempo destinado às consultas, teve grande aceitação: afora a facilidade de manejo do aparelho, a ausência da interferência do especialista permite que perguntas eventualmente consideradas embaraçosas pelas clientes sejam formuladas e respondidas sem inibições.

Outras tarefas têm sido confiadas ao micro nos EUA, onde auxilia no funcionamento interno do consultório e não apenas como um recurso restrito à sala de espera. Essas tarefas compreendem duas linhas de atuação: a relação médico/paciente e a obtenção de informações precisas e rápidas pelo médico. No primeiro caso, o acompanhamento torna-se mais ágil e eficiente pelo controle dos retornos e pela facilidade de triagem de correspondência respondida ou não pelo paciente. No segundo, a prática médica se enriquece, em virtude da ligação com hospitais, bancos de dados e até conferências, mediante acoplamento telefônico.

Os softwares específicos para gerenciamento de bancos de dados, como o dB/II ou dB Master, têm-se mostrado muito úteis nos serviços de consultório e passaram a ser recentemente utilizados por médicos também no Brasil. Uma vez aptos a operá-los e conhecedores de suas potencialidades, os médicos podem criar, manipular e relacionar os arquivos de dados segundo as suas necessidades, tendo assim acesso direto à informação desejada.

O arquivo do cirurgião Azzo Vidman, que trabalha para o INPS em São Paulo, pode ser formado de acordo com a seguinte classificação dos pacientes: nome, idade, cor, permanência no hospital e resultado da cirurgia. Tais itens são independentes entre si e, armazenados em determinada ordem, fornecem seqüencialmente relatórios e listas. A inter-relação de dados poderá esclarecer, por exemplo, o grau de incidência de determinadas moléstias; a relação entre a incidência e a idade; o índice de curas obtidas e o período médio de internações no caso de cada doença. Para o hospital, informações valiosas — do preço diário de uma internação à freqüência das causas que a determinam — também são obtidas assim.

## Problemas circulatórios

Ainda em São Paulo, o médico Fausto Miranda Júnior, especialista em moléstias vasculares, utilizando um Micro Engenho da Spectrum (equipamento compatível com a linha Apple) e o software dB Master, elaborou um programa para a análise de problemas circulatórios no cérebro. Tomando como base um exame efetuado por um doppler ultra-som, aparelho que permite a medida do fluxo sanguíneo, torna-se possível avaliar a irrigação cerebral. Os índices obtidos pelo exame de um paciente são então comparados aos índices considerados normais; numa verificação subseqüente, são comparados entre si para revelar se os problemas circulatórios se manifestam mais no hemisfério direito ou no esquerdo do cérebro. O mesmo microcomputador permite obter dados estatísticos relacionados à idade dos pacientes, à região cerebral onde é maior a incidência de uma doença específica e a freqüência de determinados distúrbios por ela provocados.

Finalmente, o mesmo programa fornece ao especialista relatórios completos a respeito de cada paciente — nome, idade e diagnóstico —, úteis para qualquer tipo de controle e dispensando consulta a arquivos manuais. Tais dados virão ainda a ser utilizados pelo especialista na elaboração de trabalhos científicos.




**Sangue no cérebro**
Com um micro da linha Apple e um software para gerenciamento de bancos de dados, o médico Fausto Miranda Júnior analisa problemas circulatórios na área cerebral. As informações por ele armazenadas poderão vir a ser utilizadas em futuros trabalhos de pesquisa científica.

# Leonardo Torres

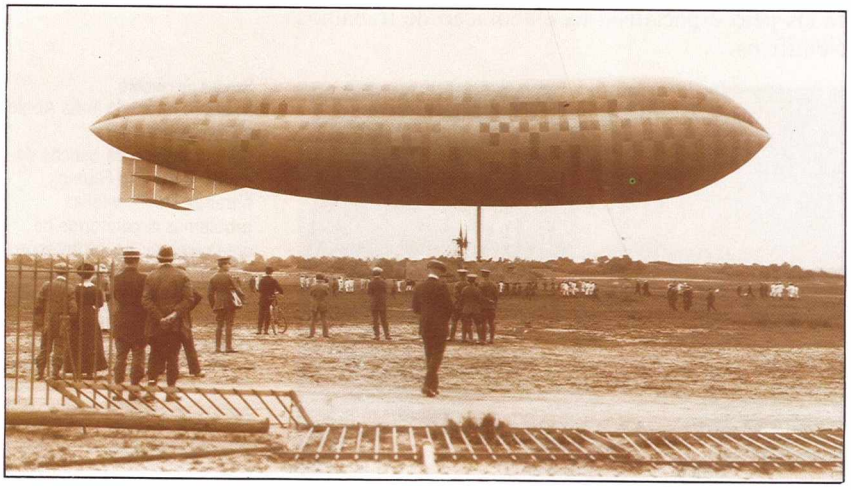
**Seus interesses estavam em aeronaves e bondinhos teleféricos, mas ele trouxe contribuições importantes para a computação.**

Leonardo Torres y Quevedo, o primeiro cientista a empregar a aritmética do ponto flutuante em computadores, nasceu em Santa Cruz, Espanha, em 28 de dezembro de 1852. Estudou no Instituto de Bilbao e na Escola de Engenharia de Madri, seguindo a carreira de engenheiro e inventor.

Seu gênio inventivo atingiu o ponto mais alto na maturidade. Ele projetou a ponte e o bondinho das cataratas do Niágara, em uso até os dias de hoje, e também uma aeronave semi-rígida, fabricada durante a Primeira Guerra Mundial. Mas Torres foi fundamentalmente um representante de sua época, e seu principal interesse residia nos aparelhos eletromecânicos. Em 1906, no porto de Bilbao, exibiu para o rei da Espanha um modelo de barco controlado pelo rádio. Em 1911, inventou o primeiro jogo de xadrez automático. O aparelho usava eletromagnetos sob o tabuleiro para mover as peças e era programado para ganhar uma partida simples contra um adversário de carne e osso.

O interesse de Torres pela automação era fruto de sua experiência nas linhas de montagem das indústrias européias no começo do século XX. Durante toda sua vida procurou separar os tipos de trabalho que exigiam raciocínio daqueles que podiam ser feitos automaticamente.

**Pontos flutuantes**
Como muitos de seus contemporâneos, Torres foi um verdadeiro politécnico. Seus interesses iam desde o projeto e construção de aparelhos mecânicos, como essa aeronave, passando por máquinas de calcular eletromecânicas, até o campo da matemática pura.



Em 1914, publicou um trabalho demonstrando que era possível construir a máquina analítica de Babbage (ver p. 220) com técnicas eletromecânicas, e foi neste trabalho que sugeriu, pela primeira vez, o uso da aritmética do ponto flutuante em algum computador do futuro. Em 1920, construiu uma calculadora eletromecânica que tinha uma máquina de escrever adaptada para dar entrada aos números e imprimir os resultados automaticamente. A máquina de escrever era ligada à calculadora por fios de telefone, e Torres anteviu a possibilidade de haver vários terminais ligados a uma calculadora central (ou unidade de processamento).

Por seus trabalhos, foi condecorado pela Academia de Ciências francesa, e mais tarde tornou-se presidente da Academia de Ciências da Espanha. Morreu em Madri, em 18 de dezembro de 1936.

## Aritmética do ponto flutuante

Uma caixa registradora mostra os totais em cruzeiros e centavos (Cr$ 12,50, por exemplo), e uma máquina deste tipo precisa de apenas dois lugares após a vírgula, para os decimais. No computador, entretanto, exige-se maior precisão e o número de lugares para decimais varia ou "flutua", de acordo com o problema, sendo o processo conhecido como "ponto flutuante" (equivalente à vírgula que usamos).

Qualquer número pode ser escrito de diversas maneiras. Por exemplo, 0,8752 m pode ser expresso em 875,2 mm ou $0{,}8752 \times 1000$ mm, ou simplesmente $0{,}8752 \times 10^3$. Este último é o método usado para fazer codificação para computador. Se um computador só aloca espaços de seis dígitos para representar os números (para maior clareza, usamos o sistema decimal em vez do binário), então o número acima pode ser armazenado como 875203, onde os dois últimos algarismos da direita são chamados de "índice" e representam a potência de 10 (3, neste caso), e os primeiros quatro números são chamados mantissa. Eis outro exemplo para este computador: o número 418302 representa $0{,}4183 \times 10^2$ ou 41,83.

A mantissa e o índice são geralmente "normalizados" para remover quaisquer zeros da frente da mantissa. Por exemplo, 41,83 poderia ser escrito 004104, mas será normalizado para 418302 — incluindo-se assim mais dígitos significativos na mantissa.

A forma índice/mantissa do ponto flutuante pode representar uma larga escala de números. O computador acima mencionado, que aloca dois dígitos para o índice, tem a capacidade de trabalhar com um número tão grande quanto $0{,}9999 \times 10^{99}$, ou tão pequeno que terá 98 zeros após a vírgula decimal, antes de aparecer o primeiro dígito diferente de zero.

Entretanto, a precisão deste sistema permanece limitada aos dígitos alocados para a mantissa. Conseqüentemente, alguns números só podem ser representados com aproximação e as técnicas de programação aritmética devem ser empregadas com muito cuidado para evitar erros. É por esta razão que em alguns computadores (1/3)*3 tem como resultado 0,9999999 em vez de 1, que é a resposta correta.

# Lance de mestre

## Jogos inteligentes como o xadrez são difíceis de programar, mas com algumas técnicas fundamentais você vai conseguir.



**Mão invisível**
Os jogos eletrônicos de xadrez têm a mesma estrutura dos computadores pessoais: CPU, memória RAM e programa gravado em ROM. A diferença fundamental fica por conta dos meios de entrada e saída. O Phantom, ao lado, usa ímãs e um servomecanismo para mover as peças em suas jogadas. Quando um cavalo, por exemplo, tem de saltar outra peça, entra em ação um sofisticado algoritmo, que remove a peça do caminho, executa o movimento do cavalo e coloca a peça de volta em seu lugar.

Um bom jogador de xadrez precisa ser uma pessoa de grande inteligência. Pouca gente poderá discordar dessa afirmação com argumentos sólidos. E, no mínimo por coerência, a maioria admitirá que máquinas capazes de jogar xadrez reproduzem, de certa forma, a inteligência humana.

É talvez por essa razão que muitas pessoas, logo que escrevem seus primeiros programas de computador, sonham com o dia em que poderão criar uma máquina inteligente que jogue xadrez com perfeição. Foi isso que levou o britânico Alan Turing, um dos pioneiros da informática, a investir tempo na estruturação de um programa desses, na esperança de criar inteligência dentro dos computadores.

Não é inteligência artificial, contudo, o que existe nos programas de xadrez e outros "jogos inteligentes": são rotinas muito bem construídas, que acompanham com precisão os movimentos do adversário — o ser humano — e avaliam todas as possibilidades com antecipação de várias jogadas. Essas rotinas, porém, nunca se alteram, por mais que o programa seja usado. Se houvesse inteligência artificial, o programa se modificaria à medida que fosse "aprendendo" novas técnicas de jogo.

Ainda assim, surgem várias dificuldades quando se pretende explicar a montagem de um programa que jogue xadrez como um mestre internacional. Aqui você encontra alguns dos princípios nos quais estão baseados os jogos inteligentes, para empregar quando decidir construir um, usando a linguagem BASIC.

Esses jogos não são fliperamas, aventuras ou simulações eletrônicas, que exigem diferentes técnicas de programação e bastante criatividade. Assim, nosso estudo começará com um exemplo que pode parecer insignificante, mas que demonstra muitos dos princípios de programação dos jogos inteligentes. É o jogo tesoura-papel-e-pedra, para dois participantes. Os dois estendem a mão ao mesmo tempo, num gesto que simboliza o objeto escolhido.

Há três regras para determinar quem vence: tesoura corta papel (nessa combinação, ganha quem escolheu tesoura); papel embrulha pedra (ganha o papel); e pedra cega a tesoura (a pedra ganha).

Para qualquer pessoa que acompanha nosso curso de programação em linguagem BASIC, é muito simples escrever um programa que faça o computador jogar tesoura-papel-e-pedra e armazenar a contagem dos dois adversários: o homem e a máquina.

As palavras que significam os objetos são armazenadas numa matriz de três elementos (três variáveis alfanuméricas) e selecionadas com o auxílio da

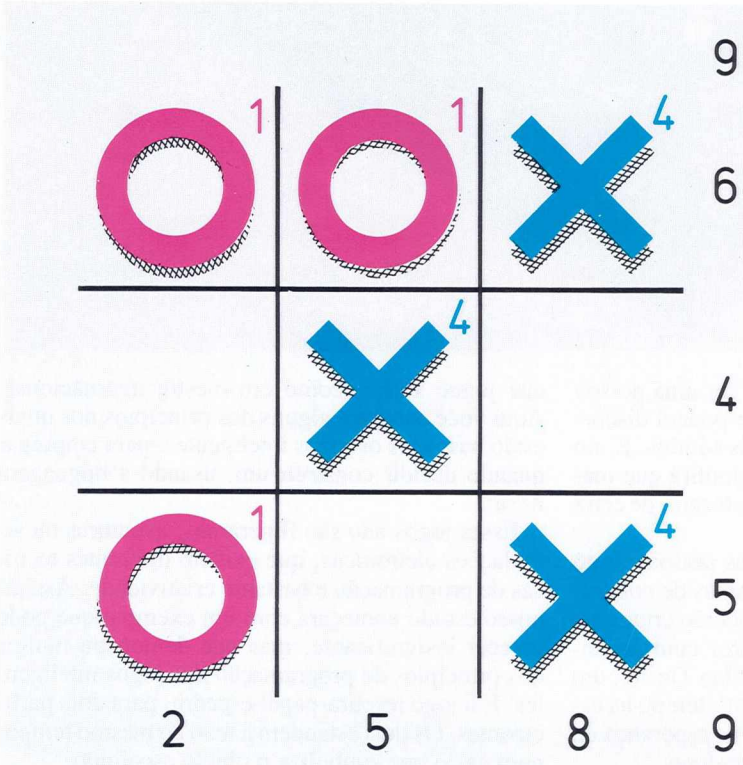
função RND. O nome do objeto escolhido pelo computador é impresso no monitor de vídeo quando a barra de espaço é pressionada.

O usuário, nesse momento, já deve também ter feito sua escolha e pode digitá-la para que a máquina verifique quem ganhou, registre a contagem e mostre, a cada rodada, o total de pontos de cada um. O programa, no entanto, não é à prova de trapaças: supõe-se que a pessoa faça sua escolha *antes* que o computador informe qual objeto escolheu.

## Estratégia computadorizada

Se a função RND for realmente randômica (ver p. 209), depois de grande número de rodadas o total de pontos de cada um deve ser quase o mesmo, seja qual for a estratégia adotada pelo jogador para derrotar a máquina. O problema que teremos de resolver, então, é como aperfeiçoar o programa para que, ao final de grande número de rodadas, o computador esteja vencendo.

Quando se analisam funções randômicas, observa-se que nem os homens nem os computadores conseguem gerar seqüências de números verdadeiramente aleatórios, embora os computadores sejam capazes de uma aproximação melhor do que a nossa.




### Contando os pontos

As avaliações são necessárias em qualquer jogo de tabuleiro, ainda que seja o trivial jogo-da-velha. Aqui, ele é montado com uma matriz de 3 × 3 elementos: o X marcado pelo computador tem valor 4 e o círculo do jogador vale 1. Desse modo, pela soma dos valores que encontra nas linhas, colunas ou diagonais, a máquina pode verificar em que posição ainda tem chance de vencer, se o jogo foi vencido por ela ou pelo adversário. Um total de 12 pontos indica que ela ganhou; de 3, vitória do adversário; de 8, que há dois X e uma casa vazia. Ou seja: cada total indica precisamente uma situação.

E o ser humano, ao jogar tesoura-papel-e-pedra, além de não fazer escolhas aleatórias, ainda dá preferência a um dos objetos. É aí que a máquina vai apanhá-lo. Pode-se, portanto, construir uma sub-rotina com outra matriz de três elementos, que registre as opções da pessoa e o número de vezes que cada objeto já foi escolhido, para que a máquina selecione o que lhe dará maior probabilidade de vitória. Esse método, contudo, apresenta inconvenientes.

Primeiro, se o computador escolher com freqüência o mesmo objeto, o jogador logo poderá notar e tirar vantagem disso. Portanto, o ideal é fazer com que o computador escolha por meio da função RND e, ao mesmo tempo, acrescentar ao programa uma rotina para assegurar que a máquina jogue com maior freqüência um objeto que ganhe da escolha preferida do adversário.

O segundo inconveniente reside na tendência do jogador em mudar sua escolha no decorrer do jogo. Então, ao invés de manter o registro das escolhas desde o início, é melhor que o programa registre, digamos, as vinte últimas escolhas.

Para isso, precisamos criar outra matriz, que se pode chamar de ESCOLHA, de 3 × 20 elementos, e uma sub-rotina que some as 3 colunas de vinte itens, para que o computador avalie melhor qual deve ser a jogada seguinte.

Contudo, a falha mais grave desse algoritmo fica evidente quando o jogador deduz a estratégia da máquina. Nesse caso, fica fácil para ele fazer suas escolhas de modo a derrotar a máquina em mais da metade das rodadas, já que se o jogador escolher muitas vezes o mesmo objeto e mudar de um momento para outro, o computador perde. Para superar esse problema, é necessário criar um algoritmo diferente.

Apesar disso, é sempre interessante desenvolver programas que usem tanto o método totalmente randômico quanto o randômico modificado que acabamos de explicar, e observar os resultados quando eles estiverem sendo usados por jogadores inexperientes.

Como um ser humano é incapaz de tomar uma decisão irracional ou aleatória, conclui-se que toda escolha é uma função de escolhas anteriores. Essa função pode ser muito complicada, e o jogador nem estará percebendo isso; mas se o computador puder determinar sua estrutura com boa aproximação, é provável que vença com muita freqüência. Como todo

jogador tem essa fórmula subconsciente, e a modifica no decorrer de um jogo demorado, o programa deve ser capaz de interpretá-la, de aprendê-la. Programas que fazem isso são chamados de heurísticos.

Um programa heurístico permite ao computador detectar alterações na estratégia do adversário e modificar a partir disso o algoritmo. Tal programa arquivaria de modo ordenado as, digamos, cinqüenta últimas jogadas de cada lado, analisando constantemente esse arquivo pela aplicação de uma avaliação estatística conhecida como "teste de correlação".

## "Grupos de trabalho"

Os testes de correlação obrigam o computador a centenas de comparações: da última escolha do jogador com a que ele fez antes; com a de duas jogadas anteriores; com a que fez cinco jogadas antes, ou até mesmo com suas próprias escolhas.

Vejamos, por exemplo, a correlação entre o último lance do jogador e seu lance anterior. Para isso, será preciso criar uma matriz de 3 × 3 elementos (são nove as combinações possíveis de tesoura, pedra e papel tomados dois a dois). Tesoura será o objeto 1, papel 2 e pedra 3. Assim, cada elemento da matriz registra uma combinação de dois objetos escolhidos em duas rodadas sucessivas.

Se, por exemplo, o jogador escolher pedra (3) depois de tesoura (1), soma-se uma unidade ao valor do elemento 1,3 da matriz. Se escolher papel (2) depois de tesoura (1), acrescenta-se uma unidade ao valor do elemento 1,2.

Caso o jogador esteja de fato fazendo escolhas aleatórias, então o total acumulado em cada elemento da matriz deverá ser aproximadamente o mesmo. Mas na prática isso não acontece. Assim, se sua última escolha foi papel (2), então o elemento da coluna 2 que tiver o maior valor mostrará qual objeto terá maior probabilidade de ser escolhido a seguir (quanto maior a diferença entre os elementos de uma coluna, mais forte a correlação e mais confiável a previsão que a máquina pode fazer).

No entanto, é possível que haja pouca correlação entre duas escolhas sucessivas. Nessa hipótese, os cálculos de correlação devem ser feitos, suponhamos, entre a última jogada e a segunda anterior a ela; ou entre uma escolha do jogador e a última escolha do computador, numa tentativa de localizar correlações.

Contudo, se diversas rotinas mostrarem resultados muito diferentes para indicar o próximo lance do jogador, isto é, correlação fraca em todas as análises, o problema se torna maior, e o programa não tem muitos dados para decidir qual o palpite mais confiável.

Neste jogo simples, basta que ele verifique qual dos testes apresenta a correlação mais acentuada.

Imagine que a matriz CORR1, que compara duas jogadas sucessivas, mostre as seguintes probabilidades para o próximo lance do jogador: tesoura, 51%; papel, 29%; pedra, 20%. E que a matriz CORR2, que compara, digamos, o último lance do jogador com o último do computador, mostre 24% para tesoura, 60% para papel e 16% para pedra. Fica evidente que CORR2 apresenta uma correlação mais forte e, por causa disso, deve ser selecionada para a previsão.

```
  5 CLS
 10 DIM C1(3,3),C2(3,3),C3(3,3)
 20 CR=0
 30 FOR I=1 TO 3
 40 IF C1(PL,I)>CR THEN BG=I:CR=C1(PL,I)
 50 IF C2(PP,I)>CR THEN BG=I:CR=C2(PP,I)
 60 IF C3(P3,I)>CR THEN BG=I:CR=C3(P3,I)
 70 NEXT I
 80 CT=BG-1
 90 IF BG=1 THEN CT=3
100 GET PT:IFPT=0 THEN 100
110 REM A LINHA 100 AGUARDA
120 REM POR UM DIGITO DO TECLADO
130 IF CT=PT-1 THEN CS=CS+1
140 IF CT=PT-2 THEN PS=PS+1
150 IF CT=PT+1 THEN PS=PS+1
160 IF CT=PT+2 THEN CS=CS+1
170 CLS
180 PRINT "SUA ESCOLHA: ";PT
190 PRINT "MINHA ESCOLHA: ";CT
200 PRINT "SEUS PONTOS SAO: ";PS
210 PRINT "MEUS PONTOS SAO: ";CS
220 C1(PL, PT)=C1(PL,PT)+1
230 C2(PP,PT)=C2(PP,PT)+1
240 C3(P3,PT)=C3(P3,PT)+1
250 P3=PP
260 PP=PL
270 PL=PT
280 GOTO 20
```

**Programa esperto**
Esta rotina simula o jogo tesoura-papel-e-pedra citado no texto e mostra como um programa pode "aprender" à medida que o jogo avança. O computador escolhe o número 1, 2 ou 3, compara com o que você escolheu e atualiza o total de cada um. A instrução GET foi usada para que você possa digitar as três teclas numa seqüência rápida. Se suas escolhas forem aleatórias, após algumas centenas de rodadas o computador estará ganhando. Você pode enganá-lo e então vencê-lo. Mas, se rotinas mais sofisticadas forem incluídas no programa, você não terá chance.

Um programa de jogo inteligente acaba exigindo várias sub-rotinas desse tipo, cada uma trabalhando numa avaliação diferente e sugerindo à rotina principal o melhor lance. São como grupos de estudos levando sugestões para que a rotina principal decida conforme a opinião da maioria. À medida que o jogo prossegue, o programa pode até atribuir notas às sub-rotinas, de acordo com seus erros e acertos, avaliando assim sua confiabilidade.

Caso seja constatada uma correlação entre os lances do jogador e os lances anteriores do computador, é possível criar mais uma arma para a máquina: o blefe, que enganará o jogador deliberadamente. Isso funciona melhor em jogos de azar, nos quais as apostas aumentam conforme avança o jogo, e quando vale a pena perder nas primeiras rodadas (aprendendo a estratégia do adversário) para ganhar nas seguintes (inclusive blefando).

Programas que jogam pôquer contêm recursos desse tipo. Em 1978, na Universidade Estadual de Nova Iorque, em Buffalo (conforme artigo da revista *Scientific American* de julho de 1978), vários desses programas foram postos frente a frente, e após milhares de partidas o vencedor foi o Avaliador de Oponentes Adaptável. Ele fazia uma avaliação inicial das cartas que supunha estarem com o adversário, modificando-a à medida que o jogo prosseguia.

Não há dois jogos de xadrez ou duas diferentes rotinas "inteligentes" que funcionem do mesmo modo. Assim, experimentando as técnicas aqui descritas, você poderá criar as suas e, quem sabe, até entrar para o seleto grupo dos programadores de jogos de xadrez.

# O mapa da mina

## Usando uma linguagem de alto nível você não precisa saber o que acontece na memória do computador. Mas, para usar linguagem de máquina, precisa saber o que se passa lá dentro.

A unidade central de processamento de um computador tem uma capacidade de endereçamento que determina o número máximo de posições de memória que pode manipular; na maioria dos micros, essa capacidade é de 64 Kbytes.

Nesse espaço devem caber os programas em ROM e RAM que vêm com a máquina, os eventualmente acrescentados e mais todas as portas e chips de interfaces, que também são considerados posições de memória. Assim, um dos aspectos mais importantes na arquitetura de um computador é seu mapeamento de memória — a lista ou o diagrama que especifica a parcela de memória destinada a cada uma das funções da máquina.

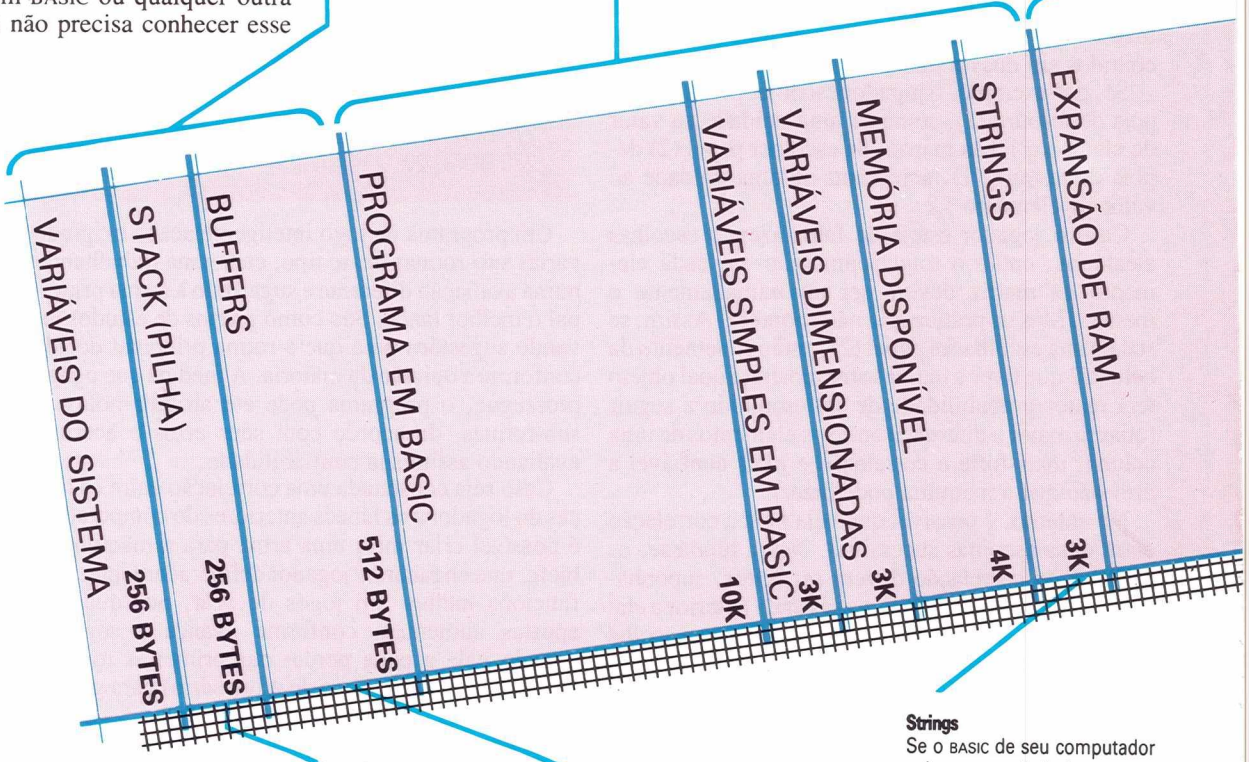
Quem só programa em BASIC ou qualquer outra linguagem de alto nível não precisa conhecer esse diagrama em detalhes; mas para quem quer utilizar linguagem de máquina ou construir acessórios de hardware esse conhecimento se mostra imprescindível.

Nestas duas páginas mostramos um típico mapa de memória, mais semelhante a um sistema baseado no microprocessador 6502 do que no Z80, embora quase todas as características sejam comuns aos dois Alguns fabricantes de microcomputadores incluem, no manual do proprietário, um completo mapa da memória. Mas há outros que fazem disso um segredo, por razões industriais. No entanto, grupos de usuários conseguiram, com a experiência, determinar todo o mapeamento.

**Overhead do sistema**
Um computador com 4 Kbytes de memória RAM pode ter apenas 3 Kbytes disponíveis para os programas do usuário. O restante fica para o overhead do sistema, parte da memória reservada pelo sistema operacional sempre que a máquina está ligada. Parte dessa área é ocupada com variáveis do próprio sistema, valores intermediários de cálculo e outros elementos

**Área do usuário (RAM)**
Talvez seja o ponto mais importante a ser avaliado na compra de um computador para uso pessoal: seu tamanho determina o grau de sofisticação dos programas que você pode usar

**Vazio**
O mapa de memória deve reservar espaço para expansão da RAM. Alguns sistemas permitem o acréscimo de mais do que 64 K, e um circuito especial seleciona os trechos que interessam ao trabalho, inserindo-os e removendo-os da memória principal




**Stack (pilha)**
Esta região está reservada para o uso da CPU e é organizada na forma de uma estrutura de dados acumulados (LIFO — Last In/First Out), onde só o que está no topo da pilha (justamente o último que entrou) pode ser manipulado. Quando uma instrução GOSUB é executada em BASIC, por exemplo, a CPU armazena no topo da pilha o endereço de memória para o qual deverá retornar (RETURN) depois de cumprir a sub-rotina

**Buffers**
A memória deve ter uma área que funcione como buffer de teclado, para que os caracteres não se percam caso entrem mais rápido do que podem ser processados. Deve haver também um buffer para o cassete, recebendo os lotes de dados que o sistema envia para serem gravados

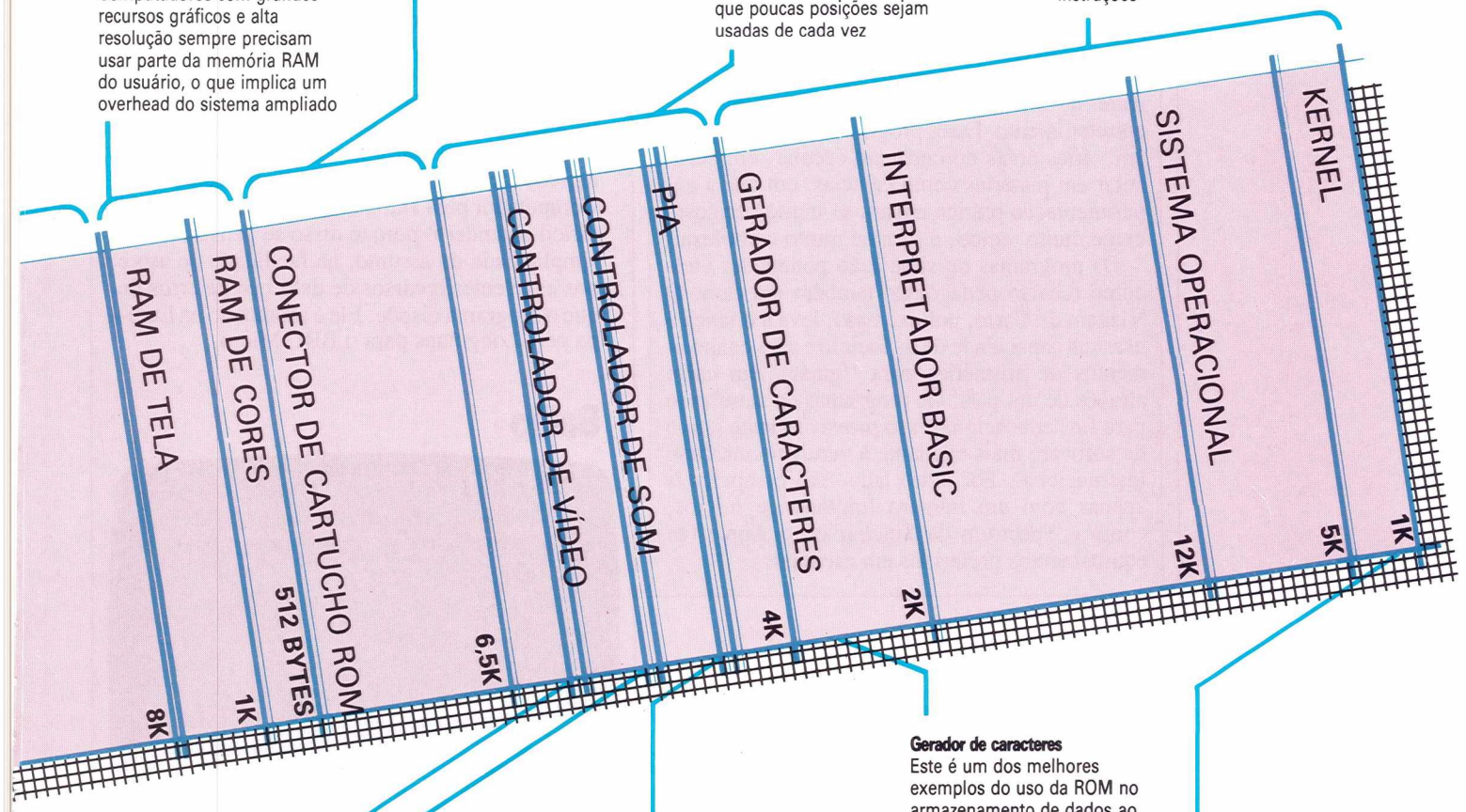
**Strings**
Se o BASIC de seu computador exige que você declare os tamanhos de todas as strings (variáveis alfanuméricas), elas são armazenadas numa tabela, como variáveis dimensionadas. Caso contrário, esses dados ficam numa região da memória que está sempre mudando de tamanho. Periodicamente, o sistema operacional faz uma "coleta de lixo" na área das strings, removendo os dados considerados desnecessários ao programa

**RAM do sistema**
Certos computadores dispõem de um trecho de memória para o sistema que não é discriminado como parte da RAM do usuário. Em geral essa região é usada como memória de vídeo (cada byte corresponde a uma posição na tela) e de registro de cores (cada byte corresponde à cor de um caractere e à cor de fundo de cada posição). Computadores com grandes recursos gráficos e alta resolução sempre precisam usar parte da memória RAM do usuário, o que implica um overhead do sistema ampliado

**Vazio**
Quando se introduz um programa por cartucho, ele é reconhecido no mapa de memória como uma expansão. Algumas máquinas têm conectores para cartuchos com linguagens diferentes do BASIC. O espaço para elas também está reservado no mapa de memória

**Chips de entrada e saída**
A CPU só se comunica com dispositivos que reconheça como posições da memória. Assim, todas as portas de interfaces e de outros circuitos devem estar no mapeamento. Isso inclui as interfaces de teclado, cassete, impressora e controlador de vídeo. Como a CPU faz endereçamento em lotes de 4 Kbytes, deve ser reservado um espaço de mesmo tamanho para os chips de entrada/saída, ainda que poucas posições sejam usadas de cada vez

**ROM do sistema**
Num computador, a ROM armazena informações que nunca se modificam. As mais importantes são as que compõem o sistema operacional — conjunto de programas que toma conta do funcionamento do computador. Esses programas cumprem tarefas como verificar se alguma tecla foi apertada e armazenar ou recuperar informação de um cassete. Na ROM fica também o interpretador BASIC, que transforma programas em instruções




**Controlador de vídeo**
Os recursos gráficos mais sofisticados têm sido baseados mais em hardware do que em software. No mapa de memória, o controlador de vídeo aparecerá como uma dúzia ou pouco mais de registradores de um byte, que determinam desde a cor de fundo para cada elemento a ser impresso até sua exata posição no vídeo

**Controlador de som**
Efeitos sonoros simples podem ser conseguidos com software. Mas os computadores com recursos mais sofisticados têm até controle de som, que alimenta um pequeno amplificador

**PIA**
Peripheral Interface Adaptors (adaptadores de interface de periféricos) são usados na comunicação do computador com periféricos simples como teclado, cassete, joystick e impressora. Os melhores (como o 6522 Versatile Interface Adaptor) convertem dados paralelos em seriais e têm cronômetros embutidos, usados em programação para controlar a velocidade de transferência dos dados

**Gerador de caracteres**
Este é um dos melhores exemplos do uso da ROM no armazenamento de dados ao invés de programas: ela guarda os bits que definem como cada caractere aparecerá no vídeo. Em certas máquinas, todos os caracteres ou parte deles são copiados na RAM; isso permite que o usuário defina seus próprios caracteres

**Kernel**
Constitui a alma do sistema operacional. Embora tenha um nome diferente em cada máquina, é o primeiro programa a ser executado pela CPU. Entre outras coisas, ele verifica o tamanho da memória disponível e se há programa em cartucho alimentando a máquina. Também manipula as formas mais elementares de entrada e saída de dados

# Faz de conta

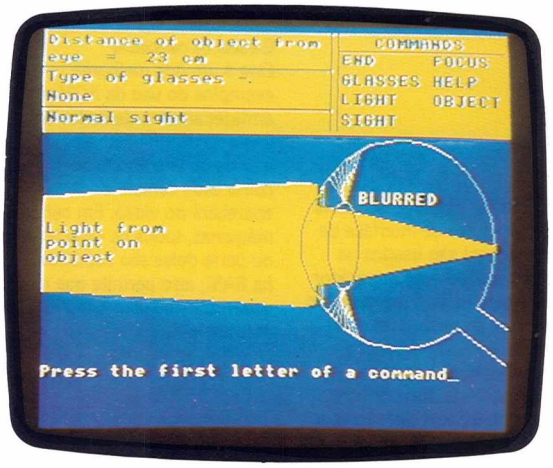
**Experiências sem amostras, material, equipamento: isso é possível com programas de simulação, que divertem e ensinam.**

Os programas de simulação, mesmo os triviais fliperamas que colocam o usuário no controle de um carro de corridas ou de um avião, são planejados para proporcionar uma experiência tão próxima quanto possível da situação real.

No entanto, existe grande quantidade de softwares de simulação com objetivos não apenas de entretenimento. Esses programas são muito úteis em várias áreas do currículo escolar, em particular em matérias como ciências, em que a experimentação prática mostra-se muito perigosa, exige muito tempo, é cara ou muito complexa.

Os programas de simulação podem ser úteis como recurso pedagógico também em casa. O Viagem de Carro, por exemplo, leva a criança a usar sua capacidade de raciocínio e seus conhecimentos de aritmética para "guiar" um carro através de um país. Os programas de simulação para fins educacionais são provavelmente o tipo de software mais excitante à venda no mercado internacional. Por outro lado, são compatíveis apenas com um número limitado de micros, como o Spectrum da Sinclair ou o Apple (os equipamentos preferidos em escolas).

## Olho




Esse programa mostra como funciona o globo ocular e como suas partes devem estar corretamente ajustadas para que haja visão nítida, simulando o percurso que os raios luminosos fazem de um objeto até a retina (a parte posterior do olho, onde as imagens se formam). Você age como cérebro, controlando fatores como a distância do objeto, o tamanho da pupila e a distância focal do cristalino, de modo que a imagem na retina esteja clara. O vídeo mostra o corte transversal de um olho, com as partes importantes identificadas. Usando o comando LIGHT, assinala-se a trajetória de um raio de luz entre um objeto e o olho. Só quando as demais variáveis forem corretamente escolhidas, o usuário receberá pelo vídeo a mensagem EM FOCO.

Uma vez entendido o funcionamento do olho normal, pode-se simular defeitos visuais como a miopia, por exemplo. Ao perceber que é impossível a focalização de objetos distantes, o usuário deverá acrescentar uma lente diante do olho. Se for escolhida a lente adequada, a visão normal estará restabelecida.

Embora tenha sido desenvolvido, de início, para aulas de física e biologia, esse programa é empregado também como auxiliar em cursos genéricos de computação, como Introdução ao Computador para alunos mais jovens. E não será difícil entender o porquê disso se pensarmos na simplicidade do assunto, na facilidade de uso e nos excelentes recursos de detecção de erros de que o programa dispõe. Ele é produzido na Europa pela Longmans para o BBC Micro.

## Balão



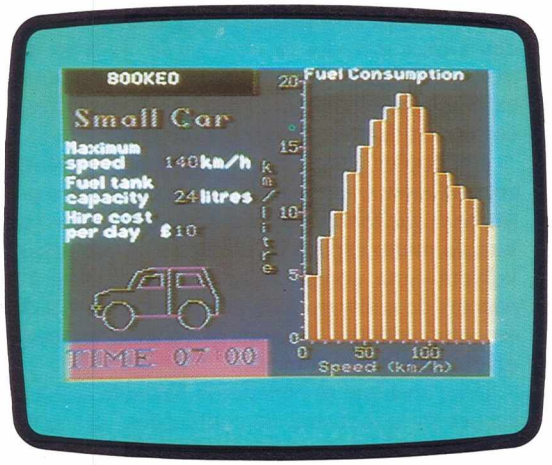

Trata-se de um programa educativo para uso doméstico. Encontrado em versões compatíveis com o computador Sinclair ZX Spectrum, é fabricado na Inglaterra pela Heinemann Educational Software. O usuário controla um balão de ar quente e deve fazê-lo voar. Na tela aparecem o perfil de uma região campestre com o balão, no solo, e os mostradores de quatro instrumentos: o indicador das velocidades de subida e descida, o medidor da temperatura do ar, o altímetro e o medidor de combustível. Existem dois controles: o do bico de gás para aquecer o ar e fazer o balão subir, e o da abertura, que deixa o ar sair e faz o balão descer.

Basta uma "lição de vôo" para ensinar o usuário a utilizar os instrumentos e controles, fazendo o balão decolar, voar e aterrissar. Domi-

nadas essas operações, você pode realizar sua "missão". Ela consiste em fazer o balão pousar em locais escolhidos (assinalados com um X), em que o balonista recebe algumas instruções. Uma das tarefas, por exemplo, é a de "ajudar o fazendeiro a encontrar carneiros" — os animais serão encontrados num campo marcado com S. Se o balão ficar sem combustível, deverá descer para receber novos cilindros de gás.

Bastam algumas decolagens experimentais e algumas colisões com árvores e outros obstáculos para se aprender a controlar o balão usando jatos curtos do bico de gás. Da mesma forma, não demorará muito até que você domine os instrumentos de modo a calcular quando deve ser usada a saída de ar ou o bico de gás. Talvez a vantagem mais importante seja aprender como controlar um sistema que inclui retardamento acentuado de tempo.

## Viagem de Carro

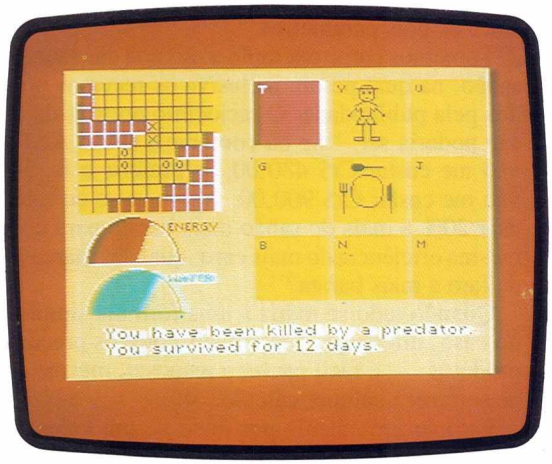



Compatível com o ZX Spectrum, é um programa educacional para uso doméstico, no qual o usuário faz o papel de proprietário de um pequeno serviço de entregas. São necessárias várias decisões relativas à escolha do contrato de entrega, à velocidade do carro e ao tipo de veículo a ser empregado. E, para tanto, o usuário precisa fazer cálculos que incluem dinheiro, distância, tempo, consumo de combustível. Um mapa do país aparece na tela, apresentando quinze cidades e as estradas principais. São também apresentados velocímetro, medidor de quilometragem, medidor de combustível e relógio.

A primeira tarefa é decidir por qual cidade começar e, então, será necessário escolher, entre as doze opções possíveis, um contrato. Um deles, por exemplo, consiste em receber uma remessa de diamantes às 12h e entregá-la em uma cidade a 350 km antes das 18h do mesmo dia. Para isso, você deve alugar um carro, ir ao local a fim de receber os diamantes e seguir para a cidade de destino. Se for bem-sucedido, receberá um prêmio em dinheiro e, se chegar antes do prazo, mais uma bonificação. O dinheiro deve ser gasto em pernoites, manutenção do carro, combustível, multas por excesso de velocidade ou por não cumprimento do contrato. Escolhendo um contrato para transporte de carga mais pesada, o carro deve ser substituído por um furgão, mais lento e de custo (aluguel e consumo de combustível) consideravelmente mais alto.

O programa ajuda a ampliar as capacidades mais abstratas de tomada de decisões e raciocínio lógico. Ensina até mesmo noções simples de Economia, pois, ao pensar nos prós e contras de determinado contrato, o usuário está fazendo uma análise de custo/benefício.

## Sobrevivência




Se você já imaginou o que deve ser a vida de um leão, ou mesmo de um rato, o programa Sobrevivência foi feito para você. Ele possibilita ao usuário fazer o papel de seis animais (águia, passarinho, leão, rato, mosca ou borboleta) e viver problemas de sua existência cotidiana.

O mundo é representado com um quadriculado na tela e você deve mover-se através dele (sua posição é mostrada pela letra A) pressionando as teclas. Suas principais preocupações devem ser encontrar alimento (os quadrados assinalados por um O) e evitar predadores (assinalados por um X). Ao se aproximar de um desses quadrados assinalados, uma ampliação da área na qual você se encontra aparece no lado direito da tela, mostrando que predadores ou alimentos estão por perto. Além disso, surgem na tela dois medidores que indicam quanta energia e água você ainda tem. Se seu nível de energia baixar, você deverá procurar alimento; e, se a água acabar, será preciso buscar um quadrado azul (um rio); entretanto, se por acaso "cair" num quadrado azul, você se afogará.

Alguns animais têm vida mais difícil do que outros: a única fonte de alimento da borboleta são as flores, que podem ser difíceis de encontrar. A águia, porém, alimenta-se de cobras, moscas e ratos; por outro lado, pode ser abatida por um caçador. Trabalhando com esse programa, você aprende como várias espécies animais se encaixam na cadeia alimentar e avalia alguns dos problemas enfrentados pelos animais na luta selvagem pela sobrevivência.

# A melhor opção

**Escolher a solução ideal para um problema nem sempre é simples. Às vezes, as alternativas são tantas que apenas um computador pode indicar com facilidade qual a melhor escolha.**

Nossas decisões sempre envolvem um compromisso — por exemplo, entre custo e eficiência ou entre custo e tempo —, sendo pouco provável que consigamos obter um máximo a um custo mínimo. A otimização do resultado estará situada em algum ponto entre os dois parâmetros.

Tomemos como exemplo a escolha das marcas de sabão, na qual pesam inclusive os recursos utilizados pela publicidade. O raciocínio por trás da decisão poderia ser: ''Se eu comprar o sabão A, 150 g vão me custar Cr$ 480,00; se preferir o B, 300 g vão me custar Cr$ 900,00. Agora, e se eu tiver de usar 20% a mais do sabão mais barato para obter a mesma eficiência do mais caro, qual das marcas será de fato a mais barata?''

Quando tudo se reduz a uma fórmula simples — no caso, diferenças percentuais entre os preços dos produtos —, é fácil prever a resposta, mesmo antes de se fazer qualquer cálculo.

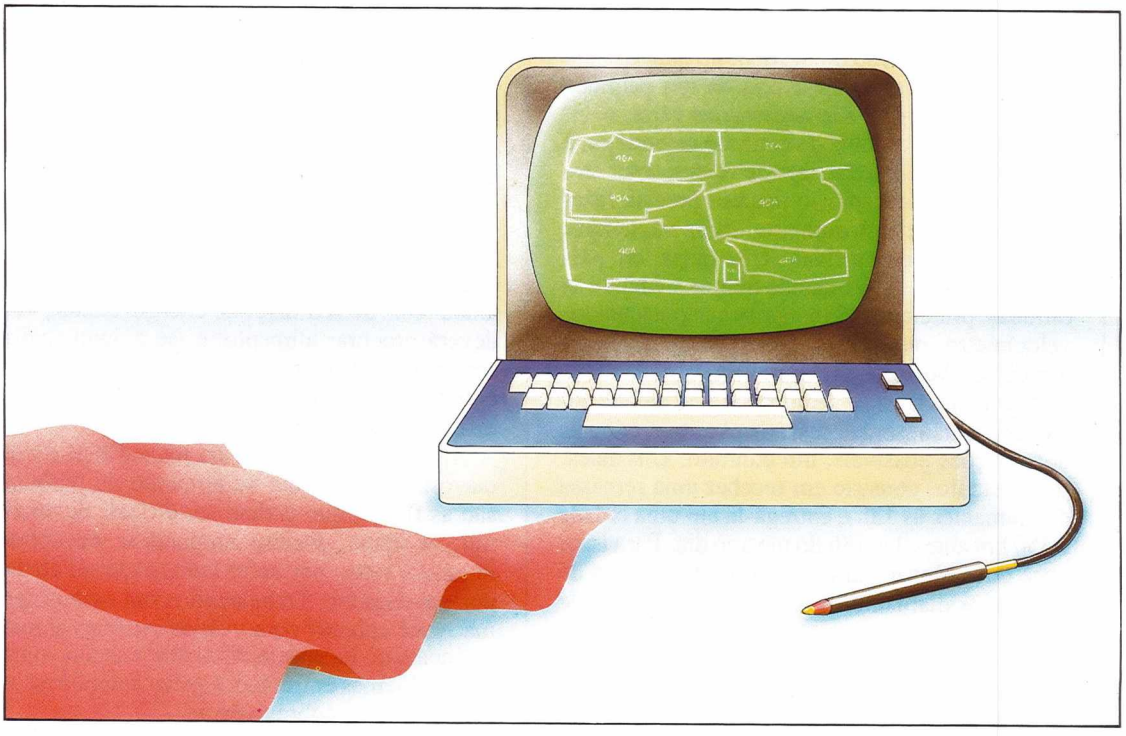
Estimar um cálculo por meio de um valor constante é normal e funciona bem quando as diferenças entre as variáveis (o preço, por exemplo, ou o peso) são também constantes. No entanto, quando essas diferenças mudam em proporções desiguais, a matemática torna-se mais complexa, e temos de recorrer a uma forma de cálculo específica (com a qual resolvemos uma série de equações que empregam os mesmos termos simultaneamente).

Quando o número de variáveis é pequeno, podemos colocá-las numa matriz e então fazer os cálculos. Outra forma consiste em ''chutar'' a resposta (ou seja, responder de modo aleatório) e modificá-la sucessivamente até que ela preencha todas as exigências do problema. Quanto melhor o ''chute'', mais depressa se chegará ao resultado.

Técnicas de otimização desse tipo revelam-se essenciais ao comércio e à indústria, sendo empregadas em todo o mundo, sobretudo na indústria e na construção. ''Programação linear'', ''Análise de caminho crítico'' e ''PERT'' (Programme Evaluation Research Technique) são alguns dos nomes dados a esses métodos de otimização. Já eram usados trinta anos antes do surgimento do computador, e exigiam muito trabalho para se chegar à resposta correta em tempo razoavelmente curto.

Aplicações desse tipo são adequadas para microcomputadores, embora as operações com matrizes (mesmo que tenham apenas duas dimensões) precisam de muito espaço na memória. Além disso, envolvem certa complexidade. Mas existem alguns pacotes de software para pequenos sistemas de microcomputadores, de modo que a técnica está disponível a qualquer momento.

Uma área que se beneficiou bastante dessas aplicações foi a indústria de confecções. Em geral, os tecidos têm largura padronizada — às vezes também



**O melhor arranjo**
Arrumar os moldes sobre o tecido de modo a minimizar as perdas é um bom exemplo de otimização computadorizada. Uma das aplicações desse método é o corte de lâminas de metal; outra, o corte de tecidos para confecção de roupas. Na ilustração ao lado, o computador sugere, no vídeo, a arrumação ideal, e um operador experiente pode fazer os ajustes finais com auxílio de uma caneta óptica.

## Dieta "perfeita"

O objetivo aqui consiste em determinar a combinação ideal de quatro alimentos que satisfaça a uma exigência nutritiva mínima a um custo mínimo. Para tanto, deve-se informar ao computador quais os componentes nutritivos (rosa), seus preços por unidade (azul) e qual a exigência mínima de cada um deles por semana (amarelo). O computador descobre o elemento mais crítico e combina-o com o resto até chegar à proporção ideal (em verde). No exemplo, as exigências foram hipoteticamente satisfeitas apenas com batata e leite.

| | PREÇO | PROTEÍNA (mg) | CARBOIDRATO (mg) | GORDURA (mg) | CALORIAS | QUANTIDADE REQUERIDA |
|---|---|---|---|---|---|---|
| BATATA, 400 g | 1.000 | 10 | 90 | 0 | 400 | 8,75 kg |
| SARDINHA, 200 g | 1.500 | 80 | 5 | 5 | 500 | — |
| CARNE, 400 g | 4.000 | 150 | 10 | 80 | 1.300 | — |
| LEITE, 0,5 l | 500 | 15 | 25 | 20 | 200 | 3,75 |
| MÍNIMO REQUERIDO POR SEMANA | | 250 | 1.000 | 150 | 10.000 | |

o comprimento — e um dos maiores problemas dos fabricantes de roupas é reduzir ao máximo as perdas no corte, tendo ainda de levar em conta fatores como a direção dos fios e até o modo de empilhamento do tecido.

## A moda e o computador

Nas mais avançadas casas de confecções européias, a colocação dos moldes sobre determinadas peças de tecido para a produção de roupas sob medida é feita com o auxílio de técnicas de otimização em computador. O resultado sugerido pela máquina é exibido em vídeo ou em desenho feito por plotter. Nesse caso, usando métodos de programação orientados para o processamento de formas, o operador do computador deve exercer seu critério de julgamento e sua experiência na tentativa de aperfeiçoar os cálculos da máquina. Em geral, ele consegue melhorar esses cálculos.

Como as exigências de cada serviço ou de cada peça de vestuário diferem bastante, esse constitui um excelente exemplo de uso racional da otimização computadorizada de baixo nível combinada à experiência do operador. Métodos mais aperfeiçoados são empregados nas indústrias que cortam muitas peças idênticas em material laminado; aí, a otimização acompanha todo o processo. Como o trabalho de corte ou estamparia faz parte de uma linha de produção, a mesma operação se repete milhares de vezes. Nesse caso, o custo da otimização, dividido pelo número de unidades produzidas, fica mais do que compensado pela redução na perda de material.

A "Análise de caminho crítico", como o próprio nome sugere, é um método para se determinar qual a linha de trabalho mais importante num processo de fabricação ou construção — ou seja, a parte do trabalho que, se não for completada no tempo previsto, tem possibilidade de interromper toda a seqüência.

Trata-se de um procedimento fundamentado no tempo; o período necessário para a execução de um segmento do processo corresponde a seu valor na tabela ou diagrama da análise. Seu uso mais comum está no planejamento dos cronogramas de obras civis, de forma que os construtores possam determinar as quantidades de mão-de-obra e de material em cada etapa da construção — encanamento antes do piso, pintores depois de pedreiros. Vários pacotes de software para microcomputadores apresentam esse método de análise.

Embora os cálculos matemáticos envolvidos no processo de otimização pareçam desanimadores aos principiantes, não se pode negar o sucesso e a força da técnica em si. Trata-se de um dos poucos processamentos normalmente executados por microcomputador que envolvem grande quantidade de números, sendo um componente importante de sistemas de inteligência artificial.



**Traçado ideal**
Projetos de estradas dependem muito das técnicas de otimização. O engenheiro preocupa-se com a inclinação da pista e com o ângulo das curvas; já o fazendeiro em cujas terras passará a rodovia tem outro tipo de preocupação. Quando uma nova estrada é planejada, deve-se reunir grande quantidade de dados, que serão usados para formar um modelo completo da situação. Esse modelo tem várias utilidades: desde simples representações gráficas até otimização da trajetória.

# Cobra 210

## É um micro potente: 128 K de RAM, três sistemas operacionais, sete linguagens.

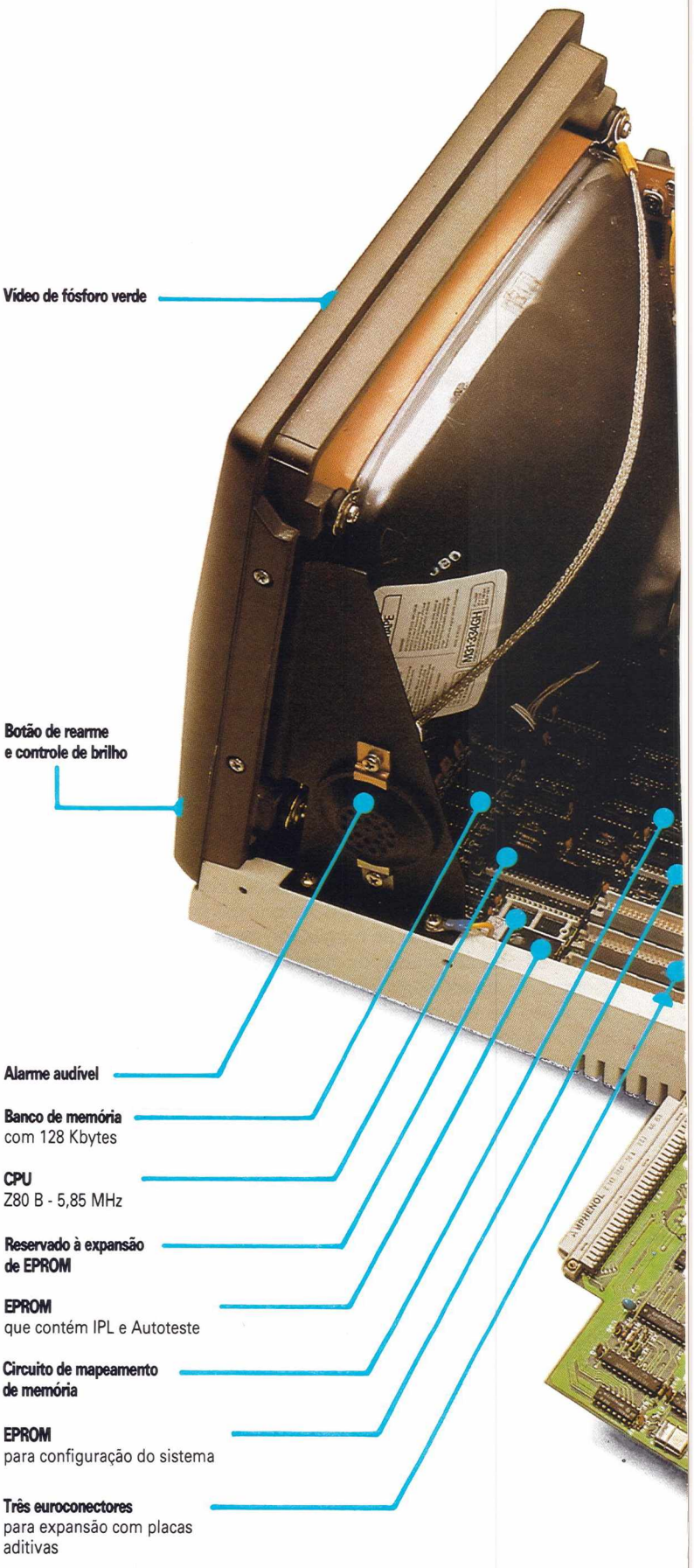
Ao ser ligado, é automaticamente submetido a uma série de testes que verificam o funcionamento do sistema; tem relógio interno e sua memória RAM faz inveja: 128 Kbytes. Essas são algumas características do Cobra 210, máquina compatível com os modelos Cobra 300 e Cobra 305, e que dificilmente pode ser selecionada para aplicações simples. Trata-se de um micro de uso profissional. Isso não impediu o fabricante de instalar nele uma interface para gravador cassete. Teclado, monitor de vídeo e gabinete com unidades de disco são módulos separados, possibilitando ao usuário maior flexibilidade na ocupação de espaço.

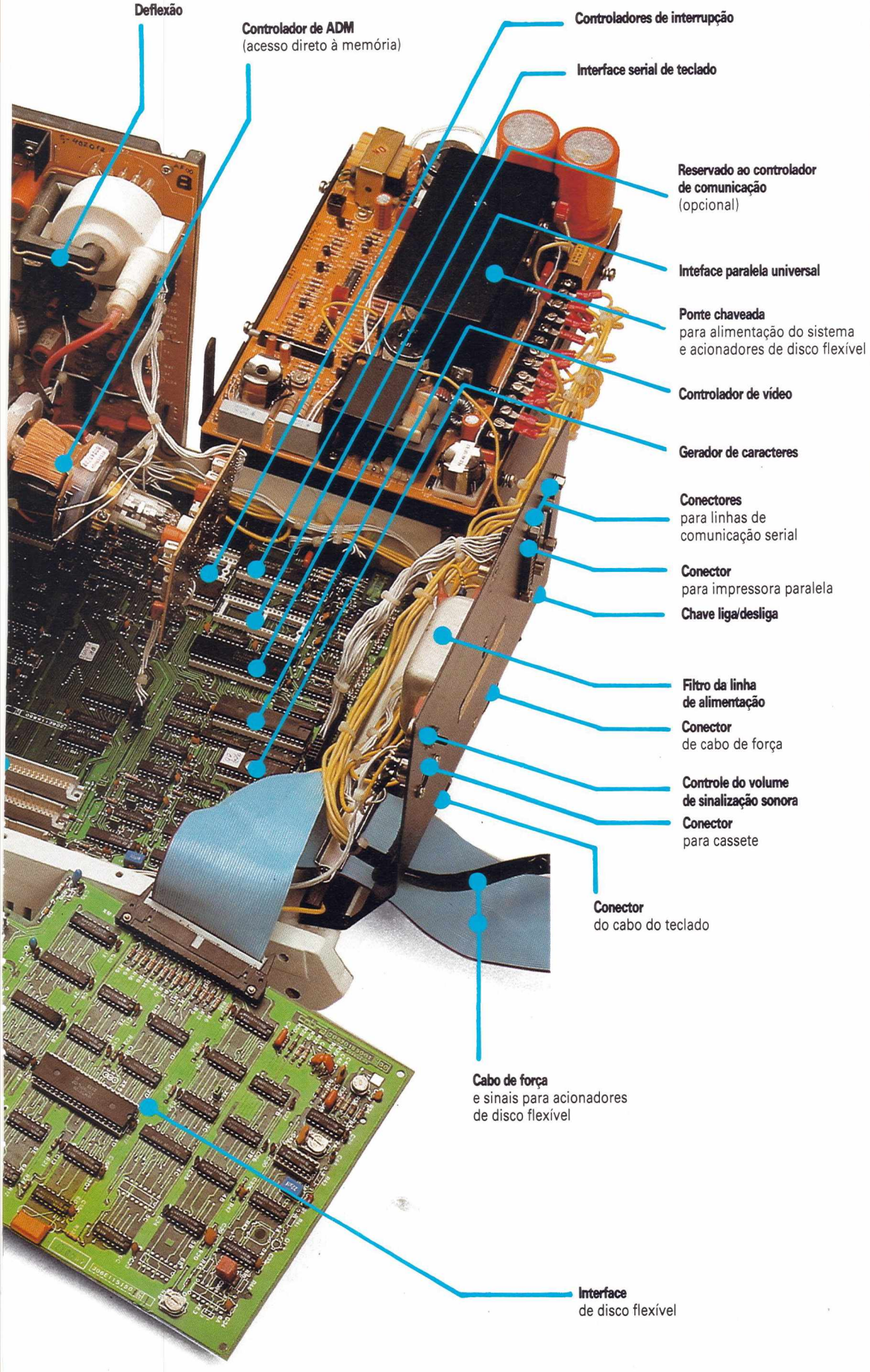
O Cobra 210 baseia-se no microprocessador Z80 B, funcionando com um clock de 5,85 MHz, com tempo de ciclo de 171 nanossegundos. Entre as 158 instruções de que dispõe, estão as de dois microprocessadores muito utilizados, o 8080 e o 8085. O monitor de vídeo é de fósforo verde, com 80 colunas e 27 linhas, uma das quais é constantemente ocupada com informações do sistema: a hora corrente, o nome do programa carregado na memória e mensagens dos sistemas operacionais.

Os caracteres usados (matriz de 9 × 11 pontos) são ASCII, com todos os símbolos utilizados em português (acentos, cedilha, til, trema), semigráficos ou qualquer outro conjunto determinado pelo usuário. Podem ser exibidos com estes atributos: intensificado, piscante, reverso, sublinhado e apagado.

Tem 88 teclas, incluindo as de acentos, de funções especiais e de especificação do modo de operação, com teclado numérico reduzido. Podem ser usadas até quatro unidades de disquetes de 8 pol., impressoras matriciais ou de linha (300 a 600 linhas/minuto), unidade de disco rígido de 5 ou 10 Mbytes e periféricos de comunicação. O Cobra 210 usa os sistemas operacionais MUMPS, SOM e SPM (compatível com CP/M).








## COBRA 210

**MICROPROCESSADOR**

Z80 B, de 8 bits, com tempo de ciclo de 171 nanossegundos

**CLOCK**

5,85 MHz

**MEMÓRIA**

RAM de 128 Kbytes para o usuário, EPROM de 64 Kbytes com rotinas de autoteste e carga inicial, EEPROM de 512 bytes para os parâmetros de configuração programáveis.

**VÍDEO**

De fósforo verde, com 27 linhas × 80 colunas (uma linha para mensagens do sistema), cursor selecionável (traço ou bloco), caracteres de 9 × 7 pontos em matriz 9 × 11; conjuntos ASCII, português, semigráfico ou qualquer outro selecionado pelo usuário.

**TECLADO**

De 88 teclas, incluindo caracteres especiais e acentos, teclas de funções especiais e teclado numérico reduzido. Conexão à unidade principal por meio de fio espiralado.

**LINGUAGENS**

COBOL I, LTD, LPS e FORTRAN IV (com o sistema operacional SOM); COBOL, ANS, LPS, BASIC (com o sistema SPM) e MUMPS (com o sistema MUMPS).

**PERIFÉRICOS**

Até quatro unidades de discos flexíveis de 8 pol. (1,2 megabyte cada), unidade de disco rígido Winchester de 5 ou 10 megabytes, gravador cassete, impressora matricial de 130 e 160 cps, impressora de linha de 300 e 600 lps; saídas para duas linhas de comunicação serial, com velocidades de 75 a 19.200 bps.

**DOCUMENTAÇÃO**

Manual de especificações da linha 200, manual de pré-instalação, guia para os sistemas operacionais, manuais de referência e de programação das linguagens disponíveis e manuais de operação dos programas utilitários.

# Imprimindo a jato

## Nitidez, rapidez e muita cor é o que oferecem as impressoras que usam minúsculos jatos de tinta, ao invés de martelar o papel.

Obter resultados impressos de processamento com elevada qualidade constituiu, durante muito tempo, uma dificuldade. As impressoras do tipo margarida resolveram o problema apenas em parte. Mas as mais aperfeiçoadas, que trabalham com microjatos de tinta, conseguem oferecer, além de boa qualidade, impressão em cores, desde que o papel tenha grande capacidade de absorção.

As impressoras destinadas a usuários de microcomputadores apresentam níveis de impressão bastante diversos. Para imprimir a uma só cor, os melhores resultados são obtidos com as impressoras de impacto (o melhor exemplo desse tipo é a margarida); já as impressoras eletrostáticas e térmicas produzem material de qualidade menos satisfatória; e a impressora matricial, embora barulhenta e de resultado tipográfico apenas moderado, é o sistema mais popular e um dos mais velozes.

Quando periféricos do tipo impressora/plotter como o Tandy CGP 115 apareceram, as limitações da impressora matricial tornaram-se mais evidentes. As impressoras/plotters têm canetas esferográficas em miniatura, com as quais criam caracteres e traçam gráficos no papel, em geral a quatro cores.

As impressoras com maiores possibilidades de superar as matriciais são as que usam jatos de tinta.

Esses periféricos, de uso já consolidado nos setores industrial e comercial (assim como a impressora a laser), começam a surgir no mercado de microcomputadores. O processo consiste em bombear tinta líquida de um reservatório e lançá-la, através de um bico, num jato muito fino. Antes de serem ejetadas, porém, as gotículas de tinta recebem uma carga elétrica de alta tensão; o efeito de válvula de lançamento é quase sempre obtido com material piezelétrico, permitindo que as gotas se formem por vibrações de alta freqüência.

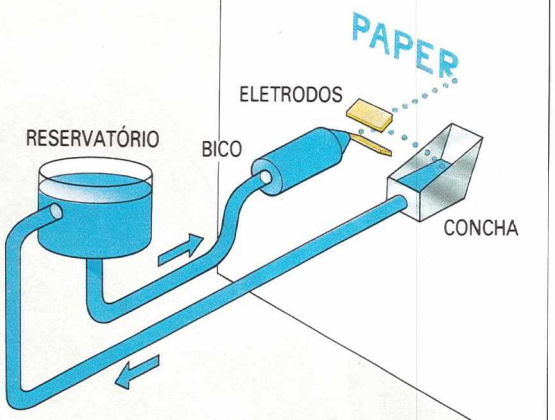
Quando deixam o bico, as gotículas ficam suspensas por um campo elétrico, que também as impele contra o papel. A folha de papel, por sua vez, está sobre uma lâmina de metal, que tem carga elétrica oposta à da tinta, o que ajuda a puxá-la para o papel.

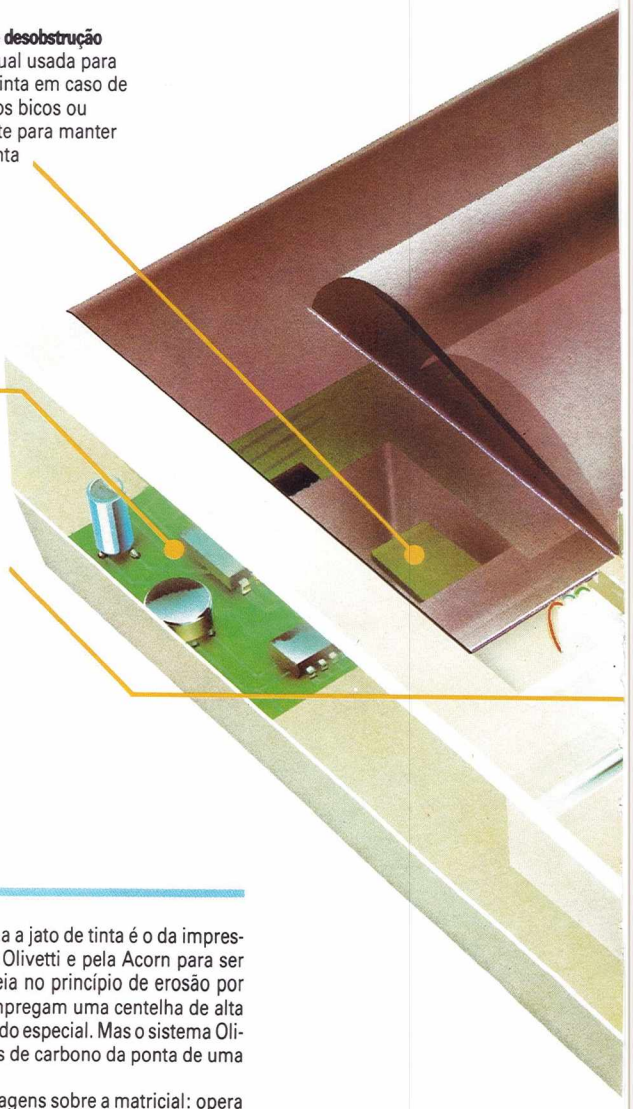
Essa técnica pode parecer pouco confiável, mas não apresenta problemas. O pior que pode acontecer é o bico entupir ou as gotas de tinta ficarem grandes demais.

Em princípio, uma impressora a jato de tinta funciona como qualquer matricial de um só martelete. A cadeia de caracteres ASCII que chega é armazenada num buffer até que este esteja cheio ou receba um comando CR (Newline). Então a impressora reconhece os caracteres um por um e procura os padrões correspondentes a eles na ROM. Em geral,

### Tiros certeiros

As primeiras impressoras a jato de tinta usavam um sistema sofisticado e muito caro. Dentro do bico, um dispositivo piezelétrico emitia um fluxo constante de gotículas de tinta eletricamente carregadas. Dois eletrodos direcionavam o fluxo, enquanto a cabeça de impressão percorria o papel. Nos lugares em que nada devia ser impresso, as gotículas eram desviadas para uma espécie de concha, de onde retornavam para o reservatório principal.



**Dispositivo de desobstrução**
Bomba manual usada para empurrar a tinta em caso de obstrução dos bicos ou simplesmente para manter o fluxo da tinta

**Placa do circuito**
Esta impressora tem seu próprio microprocessador 6809, ROM e RAM. Guarda toda a informação recebida nos buffers, pois o mecanismo só imprime uma linha de pontos a cada passagem sobre o papel

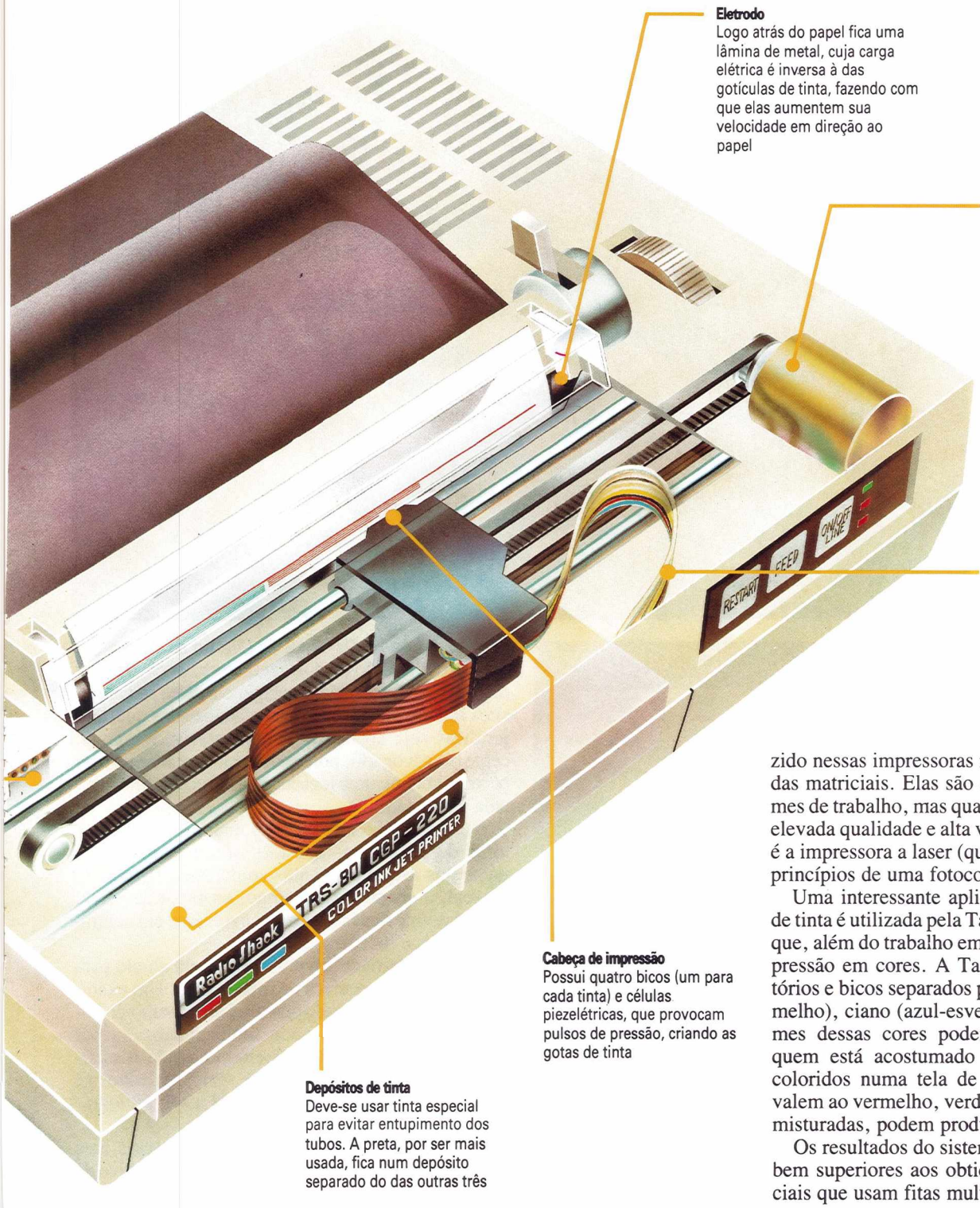
**Trava da cabeça de impressão**
Mecanismo que emite um jatc de tinta, bem mais delicado que outros dispositivos de impressão. A cabeça deve ficar travada na posição de repouso quando não está em uso. O procedimento correto em seguida ao acionamento da máquina não é complicado. Mas, se não for observado, poderá danificá-la

**Impressão a fogo**
Uma variação interessante do sistema a jato de tinta é o da impressora de "tinta seca". Fabricada pela Olivetti e pela Acorn para ser usada com o Micro BBC, ela se baseia no princípio de erosão por faísca. As impressoras desse tipo empregam uma centelha de alta tensão para perfurar um papel prateado especial. Mas o sistema Olivetti usa a faísca para levar partículas de carbono da ponta de uma haste substituível para o papel.

Esta impressora tem algumas vantagens sobre a matricial: opera silenciosamente, a cabeça de impressão é bem leve, e funciona com quase todos os tipos de papel. Mas apresentará também inconvenientes: o processo de impressão é lento, a cabeça imprime apenas uma linha de pontos a cada passagem e a tinta tende a manchar.

cada caractere é formado por vários pontos dispostos numa matriz de 8 × 8, e a impressora reproduz esse padrão no papel. Portanto, a cabeça de impressão precisa passar oito vezes sobre o papel para criar uma única linha de caracteres; contudo, a operação não é demorada, pois a impressora opera nos dois sentidos. Enquanto os caracteres contidos num buffer estão sendo impressos, o buffer seguinte vai recebendo aqueles que serão impressos, e estará cheio assim que o primeiro esvaziar. A rigor, a única diferença entre o jato de tinta e o martelo da impressora matricial é que a primeira lança gotas de tinta eletricamente carregadas sobre o papel e a segunda bate com uma agulha sobre uma fita coberta de tinta.

Em sua versão comercial, as impressoras a jato de tinta podem imprimir uma folha de papel em poucos segundos. Mas a qualidade do trabalho depende muito do tipo de papel: quanto mais absorvente for, mais tinta penetrará nele, prejudicando a nitidez. De modo geral, porém, a qualidade do material produzido nessas impressoras mostra-se muito superior à das matriciais. Elas são adequadas a grandes volumes de trabalho, mas quando se requer impressão de elevada qualidade e alta velocidade, a única solução é a impressora a laser (que trabalha com os mesmos princípios de uma fotocopiadora).




**Eletrodo**
Logo atrás do papel fica uma lâmina de metal, cuja carga elétrica é inversa à das gotículas de tinta, fazendo com que elas aumentem sua velocidade em direção ao papel

**Motores**
O movimento da cabeça de impressão é produzido por um motor convencional, enquanto o do papel é feito com um motor de avanço gradual

**Ligações**
A maioria das impressoras tem cabos flexíveis entre o circuito e a cabeça de impressão

**Cabeça de impressão**
Possui quatro bicos (um para cada tinta) e células piezelétricas, que provocam pulsos de pressão, criando as gotas de tinta

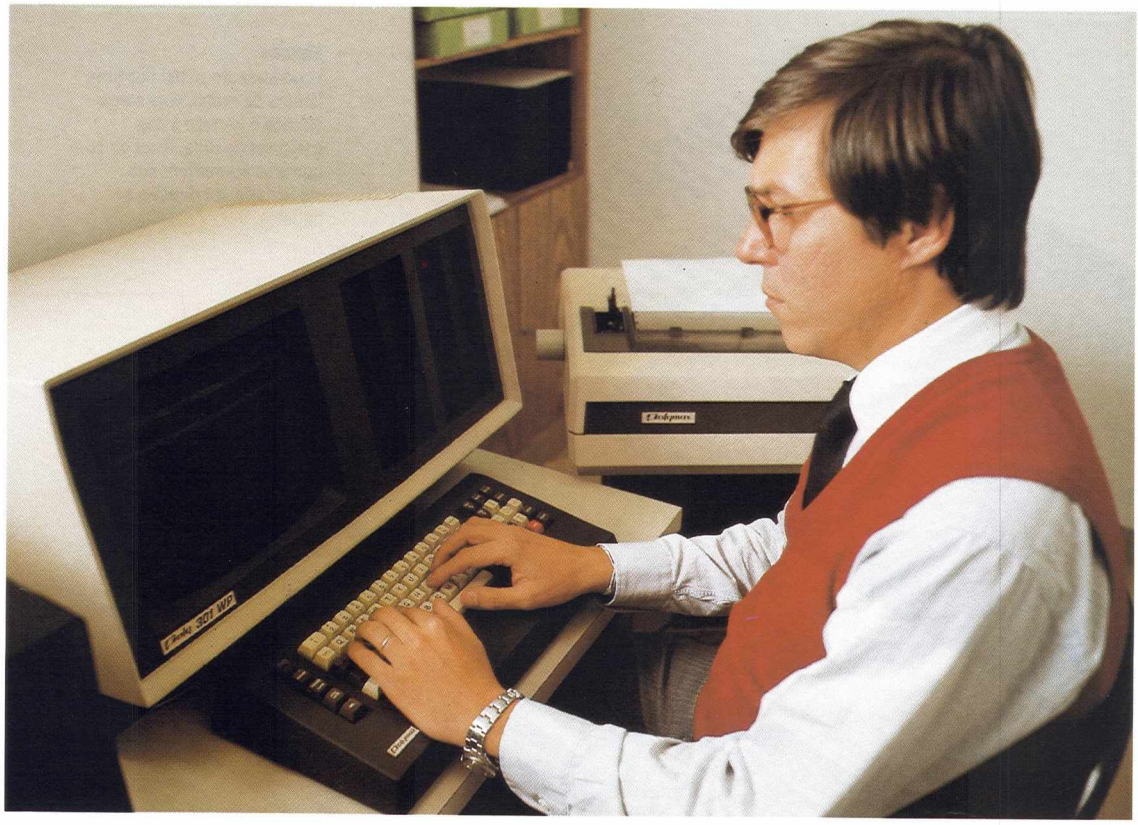
**Depósitos de tinta**
Deve-se usar tinta especial para evitar entupimento dos tubos. A preta, por ser mais usada, fica num depósito separado do das outras três

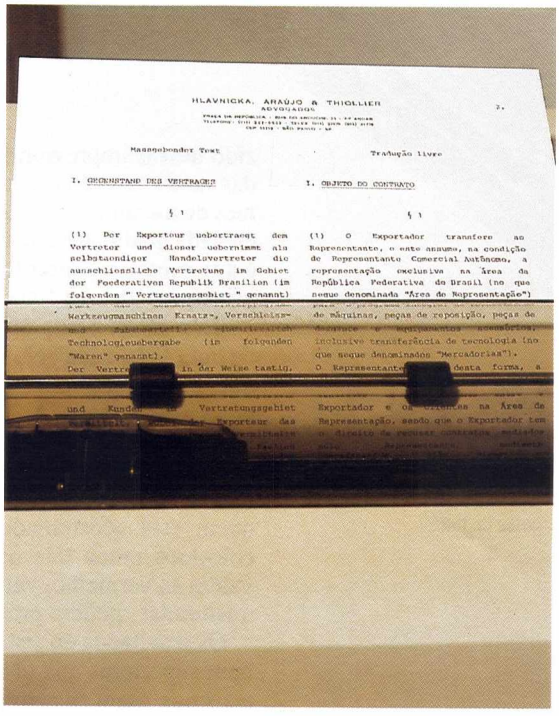
Uma interessante aplicação do princípio de jato de tinta é utilizada pela Tandy na máquina CGP 220, que, além do trabalho em preto, produz uma fiel impressão em cores. A Tandy CGP 220 tem reservatórios e bicos separados para as tintas magenta (vermelho), ciano (azul-esverdeado) e amarela. Os nomes dessas cores podem parecer estranhos para quem está acostumado a trabalhar com gráficos coloridos numa tela de televisão, mas elas equivalem ao vermelho, verde e azul do vídeo e, quando misturadas, podem produzir todas as outras.

Os resultados do sistema adotado pela Tandy são bem superiores aos obtidos em impressoras matriciais que usam fitas multicoloridas.

# Micros na advocacia

**Ao que tudo indica, num futuro bem próximo a computação poderá ajudar os profissionais de Direito até mesmo na elaboração de uma jurisprudência.**





HLAVNICKA, ARAÚJO & THIOLLIER
ADVOGADOS

Maßgebender Text — Tradução livre

I. GEGENSTAND DES VERTRAGES — I. OBJETO DO CONTRATO

§ 1 — § 1

(1) Der Exporteur übertraegt dem Vertretor und dieser übernimmt als selbstaendiger Handelsvertretor die ausschliessliche Vertretung im Gebiet der Foederativen Republik Brasilien (im folgenden " Vertretungsgebiet " genannt) Werkzeugmaschinen, Ersatz-, Verschleiss- [illegible] Technologieuebergabe (im folgenden "Waren" genannt).

(1) O Exportador transfere ao Representante, e este assume, na condição de Representante Comercial Autônomo, a representação exclusiva na Área da República Federativa do Brasil (no que segue denominada "Área de Representação") de máquinas, peças de reposição, peças de [illegible] inclusive transferência de tecnologia (no que segue denominados "Mercadorias").

Der Vertreter [illegible] in der Weise taetig, [illegible] und Kunden [illegible] Vertretungsgebiet [illegible] Exporteur das

O Representante [illegible] desta forma, a Exportador e os clientes na Área de Representação, sendo que o Exportador tem o direito de recusar contratos [illegible]

**Contrato internacional**
Em geral um contrato bilíngüe divide-se em duas colunas, uma para cada idioma. Em ambas, os parágrafos devem ter o mesmo número de linhas. Com o processador de palavras, a elaboração de um desses documentos é mais rápida e apresenta qualidade substancialmente melhor do que a conseguida com máquinas de escrever.

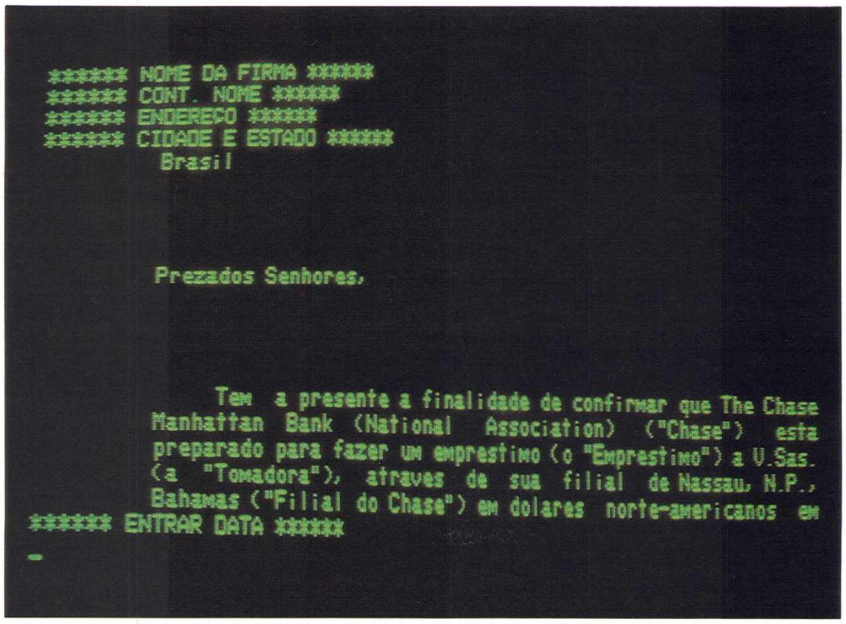
O trabalho do profissional de Direito se exerce por meio da utilização de dois recursos básicos: dados e linguagem. A precisão e a elegância desta são requisitos indispensáveis para que a análise das questões jurídicas seja ao mesmo tempo precisa e convincente. O advogado, porém, não raciocina sem se reportar diretamente à legislação e a fatos delimitados pela veracidade.

A este trabalho substantivo somam-se, num escritório de advocacia, questões administrativas comuns nos estabelecimentos comerciais, ligadas ao pessoal, à economia de tempo, ao fluxo de caixa, além das vinculadas a valores envolvidos em contratos, causas e indenizações.

Tudo isso caracteriza um campo bem amplo para a aplicação dos recursos da informática. Um exemplo é a datilografia, indispensável ao trabalho do advogado. Esse serviço pode ser racionalizado por processadores de texto, sobretudo quando o escritório reúne diversos advogados, cujos trabalhos se refletem na quantidade de material a ser datilografado.

A introdução de microcomputadores nessa área traz outra vantagem: a criação de um bom arquivo de minutas de contratos. O banco de textos do escri-






**Racionalização do trabalho**
Com o micro, pode-se comparar o custo do trabalho de cada advogado de um escritório e os honorários devidos pelo cliente. Um profissional que registra mais horas ociosas que produtivas está causando prejuízos. Convém, então, redistribuir as tarefas. Com a padronização de documentos, alterações que anteriormente exigiam muito tempo agora podem ser realizadas em minutos. É o caso de empréstimos internacionais, cujo modelo aparece no monitor, à direita.

tório Machado, Meyer, Sandacz e Ópice, de São Paulo, possuía, em julho de 1984, quarenta minutas de contratos e documentos-padrão gravados em disquetes de 5 1/4 pol., cada um deles armazenando cerca de quatro minutas de vinte folhas. Utiliza-se para isso o processador Poli 101 HP da Polimax, operado com software fornecido pelo fabricante.

Em outro escritório de advocacia paulistano, o Hlavnicka, Araújo e Thiollier, desenvolve-se um projeto mais ambicioso: em julho de 1984 a empresa estava abastecida com cem disquetes de 8 pol. armazenando minutas de contratos de compra e venda e de exportação e importação, atas de assembléias, procurações, petições padronizadas, contratos de distribuição de mercadorias, de leasing e de empréstimos etc.

A necessidade de possuir um índice organizado foi resolvida somando-se, ao índice geral, um por disquete. Assim, cada documento é codificado com a especificação de sua natureza, o número do cliente e o nome do arquivo.

A pequena variação de conteúdo presente na elaboração de contratos referentes à mesma área do Direito facilita sua preparação e a alteração de cláusulas por processamento de palavras. Anteriormente, uma pequena alteração poderia implicar um gasto de tempo útil suficiente para, incluindo a datilografia, ocupar toda uma tarde. Hoje, com a padronização de documentos, a mesma tarefa pode ser realizada em minutos.

A utilização da informática pode representar não apenas economia de tempo mas também melhora de qualidade. O emprego do processador de palavras para contratos em dois idiomas fornece um exemplo disso. Em geral, o texto de tais contratos divide-se em duas colunas, uma para cada idioma, e cada parágrafo deve ocupar o mesmo número de linhas de seu correspondente na outra língua. Contando apenas com a máquina de escrever, a execução dessa tarefa tornava-se bem difícil. Com o processador e um programa desenvolvido pela Polimax, valendo-se de um arquivo para cada coluna, é possível fazê-la com precisão e rapidez.

Para um escritório pequeno sugere-se o uso de impressora de agulhas, pois o custo da impressora margarida é bem mais alto. (Embora os tipos sejam bonitos, o gasto nem sempre compensa.) Outra providência econômica é rebobinar e reutilizar as fitas.

Na área administrativa, um recurso pode beneficiar sobretudo o escritório de grande porte, mesmo operando em outras áreas. Além de se incumbir de diversas tarefas, como o controle das contas correntes, das aplicações financeiras, das contas a pagar e a receber, e do próprio balanço, o microcomputador presta-se ao controle das horas efetivamente trabalhadas.

Compreende-se a dificuldade da aferição desse tempo nos escritórios em que trabalham vários advogados e estagiários. O programa para essa cronometragem baseia-se num relatório que cada profissional preenche diariamente, fornecendo os seguintes dados: número do advogado, dados sobre o cliente, serviços prestados a esse cliente (ida a repartições ou ao fórum, elaboração de cartas e minutas, telefonemas, presença em reuniões etc.). Registra-se ainda se os serviços são cobráveis e qual o tipo de cobrança.

A classificação dos dados é feita por cliente; por categoria profissional (de estagiário a advogado); por setor (trabalhista, fiscal, comercial); por sócio responsável; ou ainda por atividade.

De posse dos dados, o micro pode fornecer um relatório que, além de apresentar o total das horas trabalhadas em cada caso e o custo total do serviço prestado, permite o levantamento exato da situação do escritório, inclusive o número de casos que cada advogado tratou e a eventual sobrecarga de trabalho de alguns deles.

O arquivo de contratos oferece outras vantagens, entre elas a elaboração de cadastros básicos de clientes ou de advogados. Ainda em termos de arquivo, o micro propicia um índice de notícias com recuperação por palavra-chave, indicando o local em que cada uma delas está arquivada.

Mas, entre todas as comodidades que a informática oferece aos advogados, a mais importante é, sem dúvida, a criação de um banco de dados que permita o fácil acesso ao conjunto de soluções dadas pelos tribunais às questões de Direito. Se, por um lado, o alto custo desse projeto o torna inviável para um escritório, nada impede a formação de um pool com essa finalidade.

# Valores fictícios

## Para utilizar um arquivo é necessário criar primeiro uma estrutura e depois preenchê-la com dados.

No fascículo anterior, deixamos ao leitor um dilema para ser resolvido: como fazer o programa, que é processado pela primeira vez, consultar um arquivo que não existe (em fita ou disco)? A atividade inicial que provavelmente tentaremos fazê-lo executar será a consulta ao arquivo de dados e a atribuição desses dados a matrizes ou a variáveis. Ainda assim, se insistirmos nesse procedimento, sempre que o programa for processado, teremos de ser muito cuidadosos com a programação, para evitar a perda de todos os dados arquivados. Como vimos da última vez, a tentativa de abrir um arquivo não-existente simplesmente não funcionará ou então fará com que o programa "engasgue" (pare de funcionar).

Mas existe uma solução bastante simples para o dilema: muitos dos pacotes de software à venda incluem um programa de "instalação" ou de "preparação" que deve ser processado antes que o programa propriamente dito seja usado; e será esta a abordagem que adotaremos. Tais programas, em geral, permitem ao usuário algumas pequenas "adaptações" (como decidir se a impressora a ser usada deverá ser do tipo Epson ou Brother, paralela ou serial etc.), mas também criam arquivos de dados que posteriormente serão utilizados pelo programa principal. Lembre-se de que, ao contrário de arquivos de programas, os de dados podem ser acessados por qualquer programa (ver p. 316).

Para resolver nosso problema e permitir a execução da sub-rotina *LERARQ* (que lê o arquivo e atribui dados às matrizes), podemos escrever um programa de preparação que simplesmente abre um arquivo e nele registra valores fictícios. Escolhemos um valor que pode ser depois identificado pelo programa efetivo como não sendo um registro válido da agenda de endereços. Um valor conveniente seria o conjunto de caracteres @PRIMEIRO, pois não há possibilidade de que algum nome ou endereço comece assim, por mais estranha que seja sua origem. A sub-rotina *LERARQ* deverá ser um pouco alterada, de modo que, quando for aberta e a leitura executada, ela verifique esse valor antes de prosseguir. Se seu computador não dispuser do símbolo @, você deverá substituí-lo por um sinal "!" ou algum outro caractere que componha um nome mas não tenha possibilidade de vir a constar de sua agenda de endereços. A seguir, apresentamos um programa de preparação.

```
10 REM ESTE PROGRAMA CRIA UM ARQUIVO DE
   DADOS
20 REM PARA USO PELO PROGRAMA DA AGENDA
   DE ENDERECOS
30 REM REGISTRA UM ARQUIVO FICTICIO QUE
   PODE
40 REM SER USADO PELA SUB-ROTINA *LERARQ*
50 REM
60 REM
70 OPEN "O", #1, "AGEN.DAD"
80 PRINT #1, "@PRIMEIRO"
90 CLOSE #1
100 END
```

Conforme já mencionamos neste curso, os pormenores de leitura e desenvolvimento de arquivos diferem muito de uma versão para outra da linguagem BASIC, mas o princípio é quase sempre o mesmo. Primeiro, o arquivo deve ser declarado aberto antes de poder ser utilizado. A seguir, é estabelecida a direção do fluxo de dados, podendo ser de entrada (IN) ou de saída (OUT). Depois, atribui-se um número de "canal" ao arquivo. Isso permite a abertura e utilização de mais de um arquivo ao mesmo tempo (por enquanto, utilizaremos um único arquivo). Por fim, deve ser declarado o nome do arquivo escolhido.

A linha 70 do programa anterior está no BASIC da Microsoft e se assemelha, em princípio, às instruções OPEN utilizadas pela maioria das versões dessa linguagem. OPEN informa que vai ser aberto um arquivo e o dígito "O" indica que dados serão fornecidos a partir dele. O #1, por sua vez, mostra que o número 1 está sendo atribuído ao arquivo para essa operação; uma forma diferente de numeração de arquivo poderá ser empregada depois, se necessário. O nome dado ao arquivo é "AGEN.DAD".

A linha 80 simplesmente desenvolve um registro individual para o arquivo. A sintaxe do registro de dados no arquivo é (na maioria das linguagens BASIC) a mesma utilizada para impressão (PRINT), exceto que a instrução PRINT deve ser seguida pelo número de arquivo (#1, neste caso).

A linha 90 encerra o arquivo. Este poderia ficar aberto durante o tempo necessário no programa. Mas isso não é recomendável. Arquivos "abertos" são muito vulneráveis e devem ser fechados logo que possível no interior do programa, para proteger os dados neles contidos. Se, digamos, você desligar acidentalmente o computador enquanto o arquivo estiver aberto, talvez, quando vier a lê-lo, constate que os dados se perderam.

Há uma pequena confusão sobre o modo como os termos registro e arquivo são empregados em computadores, e essa confusão se acentua quando falamos de banco de dados e de arquivo de dados. Nos bancos, o arquivo consiste em um conjunto completo de dados relacionados. Por analogia com os tradicionais arquivos de escritório, o arquivo pode ser comparado a uma gaveta com a denominação PESSOAL. Ele abrange, por exemplo, um registro (um cartão em uma pasta) para cada pessoa da empresa. Cada um desses registros conterá uma série de campos idênticos, com as informações de NOME, SEXO, IDADE, SALÁRIO, TEMPO DE SERVIÇO etc.

Se o arquivo PESSOAL for computadorizado, todos os dados serão estruturados da mesma forma — um arquivo incluindo vários registros, cada registro com vários campos —, exatamente como nossa agenda de endereços computadorizada.

Entretanto, o arquivo seqüencial em fita cassete ou em disco não leva em conta o modo como os dados nele contidos são utilizados ou organizados pelo programa. Os arquivos de dados contêm apenas uma série de itens. E cada qual é um registro. Ou seja, o registro isolado no arquivo de dados normalmente não corresponde a um registro no sentido do banco de dados.

Cabe ao programa ler os registros do arquivo e atribuí-los às variáveis ou às matrizes, e estas devem ser organizadas de modo que formem um registro conceitual com um conjunto limitado de dados relacionados. Não há relação um-a-um, ou seja, correspondência entre os registros no arquivo de dados e os registros de um banco de dados.

Depois de processado, o programa de preparação não será mais necessário. Se for novamente rodado, destruirá todos os dados "válidos" que você houver fornecido ao banco de dados da agenda de endereços. Veremos por que isso acontece quando examinarmos o programa *LERARQ* já modificado.

O programa, durante a operação, não "sabe" se há ou não dados válidos no arquivo. A primeira coisa que a sub-rotina *LERARQ* efetua é abrir o arquivo "AGEN.DAD" e ler o primeiro registro (ou item de dados). Este não é lido em um elemento da matriz, como se poderia esperar, mas, sim, em uma variável alfanumérica especial a que chamamos TEST$. Antes da leitura de qualquer outro registro, esta variável é testada para verificar se contém o conjunto @PRIMEIRO. Se contiver, o programa sabe que não há dados válidos no arquivo e, desse modo, não mais precisa tentar lê-los e atribuí-los às matrizes. Conseqüentemente, o arquivo pode ser fechado e o resto do programa continua. Como não há dados válidos no arquivo, o usuário nada pode fazer até que dê entrada a um registro e o valor da variável TEST$ também possa ser usado, para permitir que o programa alcance a sub-rotina *ACRREG* e assim ao menos um registro válido seja acrescentado antes que se passe a outro procedimento.

Se, no entanto, o valor da variável TEST$ não for @PRIMEIRO, o programa pode presumir que há dados válidos no arquivo e começar a atribuí-los às matrizes correspondentes. Eis a sub-rotina *LERARQ* modificada:

```
1400 REM SUB-ROTINA "LERARQ"
1410 OPEN "I", #1, "AGEN.DAD"
1420 INPUT #1 , TEST$
1430 IF TEST$ = "@ PRIMEIRO" THEN GOTO 1540: REM CLOSE E RETURN
1440 LET CAMPNOMS(1) = TEST$
1450 INPUT #1 , CAMPMODS(1) , CAMPRUAS(1) , CAMPCIDS(1) ,
     CAMPESTS(1) , CAMPTEL$(1)
1460 INPUT #1 , CAMPINDS(1)
1470 LET TAMA = 2
1480 FOR L = 2 TO 50
1490 INPUT #1 , CAMPNOMS(L) , CAMPMODS(L) ,
     CAMPRUAS(L) , CAMPCIDS(L) , CAMPESTS(L)
1500 INPUT #1 , CAMPTELS(L) , CAMPINDS(L)
1510 REM ESPACO PARA CHAMAR SUB-ROTINA "TAMARQ"
1520 REM
1530 NEXT L
1540 CLOSE #1
1550 RETURN
```

A linha 1420 atribui um único registro proveniente do arquivo "AGEN.DAD" à variável TEST$. A linha seguinte verifica se seu valor é @ PRIMEIRO. Se for, é utilizada uma instrução GOSUB para saltar até a linha que encerra o arquivo (linha 1540) e, a seguir, a sub-rotina retorna ao programa que a chamou. Não são feitas outras tentativas de leitura. Supondo que não haja dados válidos no arquivo, o controle do programa retornará à sub-rotina *INICIL*, que então chama a sub-rotina *ESTFLG*. Esta, no momento, apenas atribui o valor 1 à variável TAMA se a variável TEST$ = @ PRIMEIRO. As instruções para a sub-rotina *ESTFLG* são dadas abaixo. Observe que há várias instruções REM que criam espaço para a inclusão de outras flags, se o quisermos depois.

```
1600 REM SUB-ROTINA "ESTFLG"
1610 REM ESTABELECE FLAGS APOS "LERARQ"
1620 REM
1630 REM
1640 IF TEST$ = "@ PRIMEIRO" THEN LET TAMA = 1
1650 REM
1660 REM
1670 REM
1680 REM
1690 RETURN
```

A sub-rotina *ESTFLG* retorna à sub-rotina *INICIL*, que, por sua vez, volta ao programa principal. A rotina *PRGPRN* chama a seguir a sub-rotina *SAUDAR*, que apresenta a mensagem de saudação na versão já publicada deste programa.

A rotina chamada a seguir pelo programa principal é *ESCLHA*. Uma pequena modificação da sub-rotina *ESCLHA* na página 357 determinará o modo de levar o usuário a acrescentar um registro, se o programa estiver sendo processado pela primeira vez.

```
3500 REM SUB-ROTINA "ESCLHA"
3510 REM
3520 IF TEST$ = "@ PRIMEIRO" THEN GOSUB 3860
3530 IF TEST$ = "@ PRIMEIRO" THEN RETURN
3540 REM "MENESC"
3550 PRINT CHR$(12)
3560 PRINT "ESCOLHER UM DOS SEGUINTES: "
3570 PRINT
3580 PRINT
3590 PRINT
3600 PRINT "1. ENCONTRAR UM REGISTRO (PELO NOME) "
3610 PRINT "2. ENCONTRAR NOMES (POR TRECHO DE UM NOME) "
3620 PRINT "3. ENCONTRAR REGISTROS (DE UMA CIDADE) "
3630 PRINT "4. ENCONTRAR REGISTROS (DE UMA INICIAL) "
3640 PRINT "5. LISTAR TODOS OS REGISTROS"
3650 PRINT "6. ACRESCENTAR UM REGISTRO"
3660 PRINT "7. MODIFICAR UM REGISTRO"
3670 PRINT "8. ELIMINAR UM REGISTRO"
3680 PRINT "9. SAIR E GRAVAR"
3690 PRINT
3700 PRINT
3710 REM "ATRESC"
3720 REM
3730 LET L = 0
3740 LET I = 0
3750 FOR L = 1 TO 1
3760 PRINT "ESCOLHER DE 1 A 9 "
3770 FOR I = 1 TO 1
3780 LET A$ = INKEY$
3790 IF A$ = "" THEN I = 0
3800 NEXT I
3810 LET ESCL = VAL(A$)
3820 IF ESCL < 1 THEN L = 0
3830 IF ESCL > 9 THEN L = 0
3840 NEXT L
3850 RETURN
```

Duas linhas foram acrescentadas. A primeira testa a variável TEST$, que ainda contém o valor lido na rotina *LERARQ*. Se for @PRIMEIRO, saberemos que não há dados válidos no arquivo e, desse modo, a única alternativa adequada é a rotina *ACRREG*, de número 6. Se o teste passar, o controle é transferido à rotina *PRIMEI*, que apresenta uma mensagem adequada e atribui o valor 6 à variável ESCL. Quando a sub-rotina retorna à linha 3530, a variável TEST$ é

novamente testada (ela está destinada a passar) e a sub-rotina retorna ao programa principal, saltando o resto da sub-rotina *ESCLHA*, uma vez que isto é inadequado.

Você talvez tenha estranhado a necessidade de TEST$ ser testada duas vezes. Isso ocorre para evitar que a sub-rotina retorne ao ponto errado do programa. Sem a linha 3530, o programa prosseguiria pelo resto da rotina *ESCLHA*, apresentando o menu de escolha, ainda que desnecessário. Ela também evita o emprego de instruções GOTO, embora a instrução IF TEST$ = "@PRIMEIRO" THEN GOTO 3850 funcionasse igualmente bem. As instruções GOTO tornam o programa confuso e difícil de acompanhar (os programas que fazem uso excessivo de instruções GOTO são apelidados de "codificação espaguete").

Antes de examinar a sub-rotina *PRIMEI*, remeteremos o leitor à rotina *LERARQ* e à instrução GOTO na linha 1430. Já que advertimos incisivamente o leitor contra o emprego de instruções GOTO, por que usamos uma delas aqui? Teria sido fácil encerrar o arquivo e retornar através do teste do valor da variável TEST$ em duas linhas separadas. Na verdade, utilizamos GOTO aqui para exemplificar uma das poucas circunstâncias em que seu emprego é admissível: no interior de segmentos de programas muito curtos e identificáveis. Sua função é evidente (e se torna ainda mais pelo comentário REM). As instruções GOTO não devem nunca ser usadas para sair de um loop (isto pode deixar o valor das variáveis em estado imprevisível), muito menos de uma sub-rotina (isto tornaria confusa a instrução RETURN, a menos que uma correspondência de retorno seja empregada na rotina), nem utilizada para saltar até pontos distantes do programa (isto faz com que o programa se torne impossível de ser acompanhado).

A sub-rotina *PRIMEI* é simples e direta: a tela é limpa e uma mensagem informa ao usuário que ele deve fornecer um registro. A linha 3870 atribui o valor 6 à variável ESCL, de modo que, quando o controle retornar à sub-rotina *EXECUT*, a rotina *ACRREG* será executada automaticamente. A codificação para a sub-rotina *PRIMVZ* é a seguinte:

```
3860 REM SUB-ROTINA "PRIMVZ" (DA MENSAGEM)
3870 LET ESCL = 6
3880 PRINT CHR$(12): REM LIMPA TELA
3890 PRINT
3900 PRINT TAB(8); "NAO EXISTEM REGISTROS NO"
3910 PRINT TAB(8); "ARQUIVO. VOCE DEVERA COMECAR"
3920 PRINT TAB(8); "ACRESCENTANDO UM REGISTRO"
3930 PRINT
3940 PRINT "(PRESSIONAR A BARRA DE ESPACO PARA CONTINUAR)"
3950 FOR B = 1 TO 1
3960 IF INKEY$ <> " " THEN B = 0
3970 NEXT B
3980 PRINT CHR$(12): REM LIMPA TELA
3990 RETURN
```

A sub-rotina *ACRREG*, apresentada na página 379, tem duas pequenas mas importantes alterações em relação à versão antes apresentada. Após fornecermos os campos como elementos das várias matrizes alfanuméricas, a variável TAMA é aumentada e atribui-se à variável TEST$ um conjunto nulo (ver as linhas 10120 e 10090). TAMA tem importância porque, utilizada em várias partes do programa, permite que este identifique os registros que estão sendo operados. Essa variável recebeu inicialmente o valor 0 como parte da sub-rotina *CRIMAT*. Mais adiante, na sub-rotina *ESTFLG* ela recebe o valor 1 se TEST$ = "@PRIMEIRO". Isso ocorre de tal modo que, quando a sub-rotina *ACRREG* é executada pela primeira vez, as instruções INPUT colocam os dados no primeiro elemento de cada matriz. Em outras palavras, INPUT "FORNECER O NOME"; CAMPNOM$(TAMA) corresponde a INPUT "FORNECER O NOME";CAMPNOM$(1).

A linha 10120 aumenta o valor da variável TAMA que passa agora a ser 2. Se a rotina *ACRREG* for executada novamente, os dados serão fornecidos no segundo elemento de cada matriz. Finalmente, a rotina *ACRREG* atribui " " à variável TEST$, na linha 10090. Faz-se isso porque agora foi introduzido um registro (embora ainda não armazenado em fita ou arquivo de dados em disco). Se a sub-rotina *ESCLHA* for novamente executada — e deve sê-lo para gravar os dados e sair do programa —, não vamos querer ser forçados a acrescentar outros registros. Se a variável TEST$ não for eliminada, o programa ficará bloqueado em um loop sem fim, e o único modo de sair será zerar ou desligar o computador, perdendo-se todos os dados.

A atribuição de uma série nula a TEST$ fará com que falhem os testes das linhas 3520 e 3530 de *ESCLHA*, permitindo a apresentação do menu de opções. O que acontecerá então à variável TAMA vai depender da rotina em execução. Até aqui apenas garantimos que TAMA=1, se não houver dados válidos no arquivo, e que isso sofre um acréscimo de uma unidade, toda vez que se adicionar um novo registro. Porém, o que acontecerá se houver uma série de registros válidos no arquivo? Para responder à questão teremos de examinar a sub-rotina *LERARQ* novamente.

A linha 1420 lê o primeiro item de dados na variável TEST$. Se não for @PRIMEIRO, fica suposto que o item de dados é válido. Os registros no arquivo sempre estão na mesma ordem, a saber: CAMPNOM, CAMPMOD, CAMPRUA, CAMPCID, CAMPEST, CAMPTEL, CAMPIND, CAMPNOM, CAMPMOD etc. Se o primeiro registro lido for um dado válido, deve pertencer ao primeiro elemento da matriz CAMPNOM$; assim, a linha 1440 transfere esse dado da variável TEST$ para a CAMPNOM$(1). As duas linhas seguintes completam os primeiros elementos nas outras cinco matrizes. Sabemos agora que temos pelo menos um registro (banco de dados) completo; assim, a variável TAMA recebe o valor 2, uma unidade maior que o número de registros válidos fornecidos às matrizes. Caso contrário, a sub-rotina *ACRREG* registraria os novos dados em elementos que já contêm dados válidos.

A seguir, um loop de 2 a 50 fornece os dados para todas as seis matrizes, aumentando o índice L em cada passagem. Já decidimos limitar nosso programa a operar com arquivos contendo cinqüenta nomes e endereços; as instruções DIM, na sub-rotina *CRIMAT*, reservaram espaço para isso. Entretanto, quando começar a usar o programa, é improvável que você disponha de arquivo completo com cinqüenta entradas. Assim, vamos precisar de uma rotina no programa que possa detectar esse fato, adequar a variável TAMA e interromper o loop de leitura.

Por esse motivo, incluímos a linha 1510, que pos-

sibilita chamar uma sub-rotina *TAMARQ*, com a qual trabalharemos mais adiante no curso. Há três modos de lidar com esse problema. Primeiramente, quando transferimos os dados para a fita, podemos fazer com que o primeiro registro inscrito seja a variável TAMA. A sub-rotina *LERARQ* pode então ser alterada para fornecer os dados, em primeiro lugar, a TAMA e, a seguir, criar um loop na forma FOR L = 1 TO TAMA, para colocar os dados no registro. O segundo (e preferível) método (uma vez que não contradiz nosso teste anterior para a flag @PRIMEIRO, na linha 1430) consiste em estabelecer um procedimento a ser executado após todos os registros terem sido transferidos — aí uma flag especial (na forma @ENCERRAR, por exemplo) pode ser incluída no final. Torna-se então viável a inserção de um teste na sub-rotina *LERARQ* para interromper o loop ao encontrar a flag @ENCERRAR.

O terceiro método consiste em usar a função EOF (END OF FILE, fim do arquivo) existente em alguns computadores, a qual na verdade é uma versão automatizada do segundo método. Esses computadores possuem uma flag EOF na maioria das vezes com o valor 0, isto é, FALSO, mas assumindo novamente valor (quase sempre valor 1, que representa VERDADEIRO), quando o fim do arquivo é atingido. Alguns equipamentos em BASIC permitem testar a flag EOF como uma variável dessa linguagem; nesse caso, o problema será resolvido através de uma construção, com a seguinte forma:

```
ENQUANTO (WHILE) NÃO EOF(N) (N é o número do arquivo)
  REALIZAR (DO)
    ENTRAR (INPUT #N, dados para ler)
ENCERRAR O LOOP WHILE-DO
```

Em outras máquinas, a flag EOF é representada como um único bit que deve ser acessado por meio da instrução PEEK. Para saber se seu micro possui a função EOF, você precisará consultar o manual de instruções. Não usaremos essa instrução em nosso programa porque difere muito de uma máquina para outra. Como exercício, porém, talvez o leitor queira experimentar uma alteração da sub-rotina *LERARQ* para os três métodos possíveis de lidar com arquivos de menos de cinqüenta itens.

Geralmente, é muito mais fácil desenvolver programas que lidam com arquivos de extensão fixa, mas resolver o problema dos de extensão variável e dinâmica nos ajudará a alterar depois o programa, ampliando-o para mais de cinqüenta itens.

```
4000 REM SUB-ROTINA *EXECUT*
4010 REM
4019 IF ESCL = 6 THEN GOSUB 10000: REM CONSULTAR NOTA DE RODAPE
4020 REM COMO USUAL "ON ESCL GOSUB etc." — CONSULTAR A NOTA DE RODAPE
4030 REM
4040 REM 1 = *ENCREG*
4050 REM 2 = *ENCNOM*
4060 REM 3 = *ENCCID*
4070 REM 4 = *ENCINI*
4080 REM 5 = *LISREG*
4090 REM 6 = *ACRREG*
4100 REM 7 = *MODREG*
4110 REM 8 = *ELMREG*
4120 REM 9 = *SAIPRG*
4130 REM
4140 RETURN
```

A rotina *EXECUT* não teria, em condições normais, a linha 4019 (daí atribuir-lhe um número ímpar) e a linha 4020 seria:

```
ON ESCL GOSUB número,número,número etc.
```

ou, então, uma série como a seguinte:

```
IF ESCL = 1 THEN GOSUB número
IF ESCL = 2 THEN GOSUB número etc.
```

A linha 4019 foi incluída para que o programa funcione mesmo sem a prévia codificação das demais sub-rotinas *EXECUT*.

```
10 REM "PRGPRN"
20 REM *INICIL*
30 GOSUB 1000
40 REM *SAUDAR*
50 GOSUB 3000
60 REM *ESCLHA*
70 GOSUB 3500
80 REM *EXECUT*
90 GOSUB 4000
100 END
1000 REM SUB-ROTINA *INICIL*
1010 GOSUB 1100: REM SUB-ROTINA *CRIMAT* (CRIA MATRIZES)
1020 GOSUB 1400: REM SUB-ROTINA *LERARQ* (LE ARQUIVOS)
1030 GOSUB 1600: REM SUB-ROTINA *ESTFLG* (ESTABELECE FLAGS)
1040 REM
1050 REM
1060 REM
1070 REM
1080 REM
1090 RETURN
1100 REM SUB-ROTINA *CRIMAT* (CRIA MATRIZES)
1110 DIM CAMPNOM$(50)
1120 DIM CAMPMOD$(50)
1130 DIM CAMPRUA$(50)
1140 DIM CAMPCID$(50)
1150 DIM CAMPEST$(50)
1160 DIM CAMPTEL$(50)
1170 DIM CAMPIND$(50)
1180 REM
1190 REM
1200 REM
1210 LET TAMA = 0
1220 LET RMOD = 0
1230 LET GRAV = 0
1240 LET CORR = 0
1250 REM
1260 REM
1270 REM
1280 REM
1290 REM
1300 RETURN
10000 REM SUB-ROTINA *ACRREG*
10010 PRINT CHR$(12): REM LIMPA TELA
10020 INPUT "FORNECER O NOME" ; CAMPNOM$(TAMA)
10030 INPUT "FORNECER A RUA" ; CAMPRUA$(TAMA)
10040 INPUT "FORNECER A CIDADE" ; CAMPCID$(TAMA)
10050 INPUT "FORNECER O ESTADO" ; CAMPEST$(TAMA)
10060 INPUT "FORNECER O NUMERO TELEFONICO"; CAMPTEL$(TAMA)
10070 LET RMOD = 1: REM FLAG DE REGISTRO MODIFICADO
10080 LET CAMPIND$(TAMA) = STR$(TAMA)
10090 LET TEST$ = " "
10100 REM INSERIR CHAMADA PARA *MODNOM* AQUI
10110 LET ESCL = 0
10120 LET TAMA = TAMA + 1
10130 REM
10140 REM
10150 RETURN
```

# Concorrência criativa

**Inventando e desenvolvendo máquinas tabuladoras, Hollerith e Powers abriram caminho no processamento da informação.**



**James Powers**
As máquinas de Powers destinavam-se a aplicações específicas. Exerceram forte concorrência às de Hollerith.



**Herman Hollerith**
Hollerith inventou a máquina eletromecânica leitora de cartões, que mais tarde deu origem aos "tabuladores".

As máquinas que Herman Hollerith inventou (ver p. 240) para apurar os resultados do censo de 1890 nos Estados Unidos deram origem a uma série de equipamentos para processamento de dados, conhecidos como "tabuladores". Antes do aparecimento dos primeiros computadores de uso comercial, na década de 50, os tabuladores foram essenciais para o crescimento da indústria e do comércio.

Em Pittsburgh (EUA), na década de 30, por exemplo, uma grande loja de departamentos usou em caráter experimental um sistema de crediário em que 250 terminais, distribuídos pela loja, ligavam-se a uma central de tabulação por meio de linhas telefônicas. O preço das mercadorias estava anotado em etiquetas perfuradas e a informação era enviada automaticamente aos tabuladores, que registravam a venda e preparavam a cobrança. Após verificar o limite do crédito do cliente, o tabulador mandava para uma máquina de escrever "on-line" do terminal a autorização de venda.

A concorrência no mundo dos negócios forneceu o estímulo inicial para o desenvolvimento dos tabuladores. O monopólio de Hollerith na produção dessas máquinas para o Departamento de Censo americano foi quebrado em 1910, quando o governo convidou também James Powers para fornecer equipamento alternativo.

o Powers montou um sistema de tabuladores totalmente mecânico que, assim, não feria as patentes dos aparelhos eletromecânicos de Hollerith. A rivalidade entre os dois — e mais tarde entre as companhias por eles criadas — teve como conseqüência o desenvolvimento das máquinas de processamento de dados.

Em 1902, Hollerith criou um painel de tomadas (parecido com o de uma mesa telefônica) que selecionava as colunas a serem somadas nos cartões perfurados, para o cálculo dos totais. Desse modo, sua máquina tinha uma capacidade de programação que não era igualada pelo concorrente — que sempre produziu máquinas para aplicações específicas.

Em 1924, Powers patenteou uma forma de representar letras e números em cartões perfurados: um único orifício em certa coluna representava determinado número; e a combinação de orifícios representava letras. Hollerith logo replicou com seu próprio sistema: o cartão de 80 colunas, que até hoje é usado como padrão. Cada coluna do cartão continha doze linhas de orifícios, que eram "lidos" por escovas de arame: onde havia orifício, fechava-se um circuito elétrico com a placa de metal por trás do cartão.

Os primeiros tabuladores apenas somavam ou acumulavam totais; mais tarde, porém, foram introduzidas funções matemáticas mais complexas para manipular os dados. Ao contrário dos computadores, inventados por cientistas com a finalidade de ajudar em cálculos matemáticos, o tabulador foi criado para ser um processador de informações. Mas os usuários logo descobriram outras aplicações para essas novas máquinas: tabuladores especiais foram adaptados para uso em mesas de computação, em análise de ondas e em astronomia — foi com sua ajuda que se descobriu Plutão, em 1930.

Os tabuladores acabaram se tornando sofisticados a ponto de combinar grandes quantidades de dados no processamento. A IBM chegou a patentear um aparelho que controlava a movimentação de 10.000 contas bancárias.

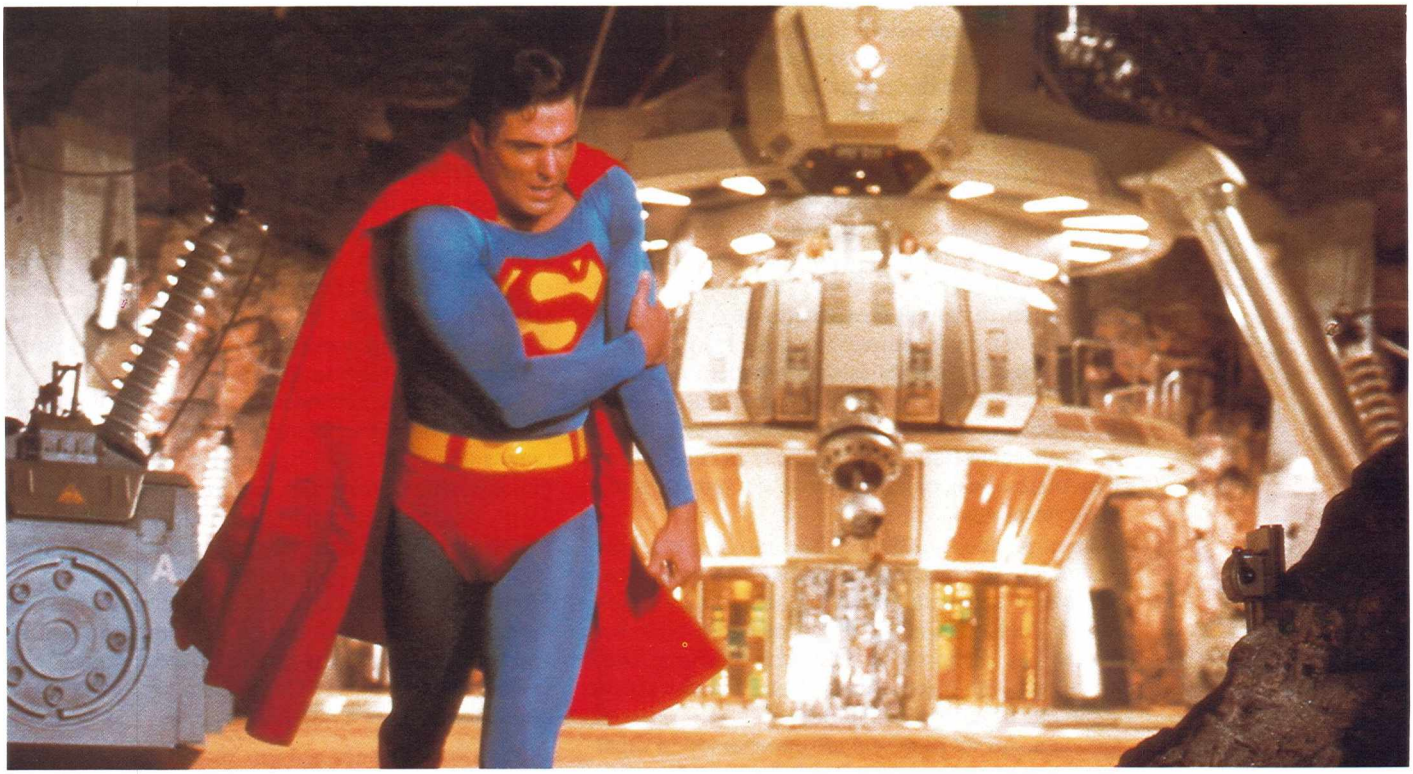
## Máquinas tabuladoras

No início dos anos 50, época de seu apogeu, o tabulador era um equipamento formado por oito unidades independentes. Havia uma perfuradora de cartões para registro dos dados — era capaz de processar duzentos por hora; outra máquina era a "unidade fiscal", que conferia o trabalho de perfuração; havia, ainda, uma unidade perfuradora de "reprodução", para copiar os cartões que estivessem velhos. Para imprimir explicações sobre os cartões, usava-se uma unidade chamada "interpretador". O tabulador propriamente dito registrava os totais de perfurações de cada coluna "lendo" cerca de 9.000 cartões por hora; estava quase sempre ligado a uma "perfuradora multiplicadora", que executava funções matemáticas mais complexas. Para comparar os dados de duas pilhas de cartões, existia um "compilador". Afinal, a unidade "seletora" dividia uma pilha de cartões em outras treze (uma para cada um dos doze furos e uma para a coluna vazia).

A operação do tabulador podia ser modificada pelos furos de controle (na 11.ª e 12.ª posições), feitos em cartões coloridos. Ao encontrar um, a máquina imprimia o subtotal e iniciava nova operação. Algumas técnicas de processamento de dados foram desenvolvidas a partir do tabulador, dando origem às primeiras linguagens de programação para os computadores.



# Ficção e realidade



**Os computadores estão quase sempre presentes na ficção científica. Muitos escritores previram tecnologias que agora são corriqueiras.**

Muitas conquistas técnicas e científicas foram imaginadas por escritores de ficção ou cineastas bem antes de se tornarem reais. Arthur C. Clarke, o autor de *2001 — Uma Odisséia no Espaço*, foi o primeiro a propor a idéia de satélites geoestacionários, em artigo publicado na revista *Wireless World (Mundo sem fios)*, na década de 50, quase vinte anos antes de seu desenvolvimento efetivo. O conto de Robert Heinlein, *Waldo*, descrevia manipuladores com controle remoto muito antes que os robôs viessem a ser usados. De fato, inúmeros inventores e cientistas inspiraram-se em idéias criativas dos escritores de ficção científica e roteiristas.

Entretanto, os computadores da literatura apresentam pouca semelhança com a realidade atual. No filme de futurologia *Rollerball*, por exemplo, um computador com reconhecimento de voz e sintetizador de som tem a forma cúbica de um tanque de líquido. Os computadores que se assemelham a modelos autênticos são pouco emocionantes e menos interessantes, embora sempre tenham sido vistos em filmes como parte do cenário. Sem dúvida, os filmes dos anos 60 e 70 que apresentavam computadores com forma bastante próxima à dos computadores atuais ajudaram a educar o público, mostrando a aparência real dessas máquinas novas e quase míticas.

As imaginações criativas começaram a formar idéias sobre computadores pouco depois de Charles Babbage (ver p. 220) ter iniciado seu trabalho pioneiro com a máquina analítica, na metade do século XIX. Em 1879, Edward Page Mitchell escreveu uma história intitulada *The Ablest Man in the World (O homem mais capaz da Terra)*, que descreve a implantação de uma máquina de calcular no cérebro de um deficiente mental, transformando-o em gênio. As idéias de Mitchell precederam os progressos científicos atuais em muitos aspectos. Ele teve a idéia da miniaturização — o aparelho de computação é ao mesmo tempo suficientemente pequeno para caber no crânio do deficiente e potente o bastante para dotá-lo de um intelecto de alta capacidade. Além disso, Mitchell antecipou a idéia de ligar o computador ao corpo humano. Hoje, mais de um século após a história ter sido escrita, as técnicas para conexão de dispositivos eletromecânicos — simples e controláveis — com o sistema nervoso central começam a ser aperfeiçoadas.

Em geral, poucos escritores têm conhecimento amplo da arquitetura do computador, embora alguns sejam engenheiros de alto nível e muitos utilizem o computador (como processador de palavras) em seu trabalho. Todavia, a maioria deles pode apresentar cenas convincentes de uma viagem intergaláctica, mesmo não sendo astrofísicos qualificados ou especialistas em foguetes. De modo semelhante, não há por que escritores não possam especular sobre as qualidades das gerações futuras de computadores sem amplo conhecimento das máquinas em uso.

**Superman III**
Fraude no computador é o tema central do terceiro filme do Super-homem. Richard Pryor representa um vilão que faz fortuna roubando meio cent de dólar em cada transação que passa pelo computador de um banco. Essa parte do enredo baseia-se em vários casos reais de fraude. O filme termina com a destruição do maior computador do mundo, construído para fins criminosos.

Essa especulação, no entanto, conduziu a um padrão de computador de ficção científica que parece ter — pelos conhecimentos atuais de computação — uma série de recursos impossíveis de ser obtidos. Para começar, esse computador padrão mantém armazenadas na memória todas as informações e idéias que poderiam ser pensadas e recupera de imediato quaisquer dados por processos análogos às faculdades da mente humana. O computador HAL de *2001 — Uma Odisséia no Espaço* é uma dessas máquinas inteligentes.

O supercomputador postulado pelos escritores de ficção científica é também onipresente, embora pareça a cada usuário que só ele tem acesso à máquina. A emissão de voz (que não sugere ser uma produção sintética de fonemas encadeados) e a identificação da voz (que sempre deixa de lado as características individuais da fala) constituem requisitos essenciais dessa supermáquina, enquanto o reconhecimento visual de objetos e a capacidade de sintetizar alimento (talvez a partir de seus constituintes elementares básicos) são outros de seus atributos mais constantes.

O computador que aqui delineamos em geral dispõe também de qualidades humanas, e essa caracterização o assemelha a um ser superdotado. Contudo, a "personalidade" do computador pode ser, às vezes, maligna ou degenerada. No filme *Dark Star*, uma bomba controlada por computador recebe as características de instabilidade de um assassino psicopata. Ao ser representado desse modo, o supercomputador faz parte do reino da fantasia. Por outro lado, podemos admitir que no moderno equipamento de computação está o possível ancestral de máquinas com alguns dos atributos que delineamos.

A alta capacidade de memória com curtíssimo tempo de acesso já se tornou possível. Foram criadas, no começo da década de 80, a memória gigabyte (1 bilhão de bytes) e as mais velozes máquinas comerciais que processam mais de 10 milhões de instruções por segundo. No campo da emissão de voz, estamos nos aproximando da perfeição apresentada nos filmes em que computadores conversam com seus operadores. A qualidade da voz depende apenas do espaço de memória disponível, da velocidade de processamento e do tempo de programação. O reconhecimento de voz, porém, é mais difícil de obter porque existe uma variação muito grande entre as características individuais da fala.

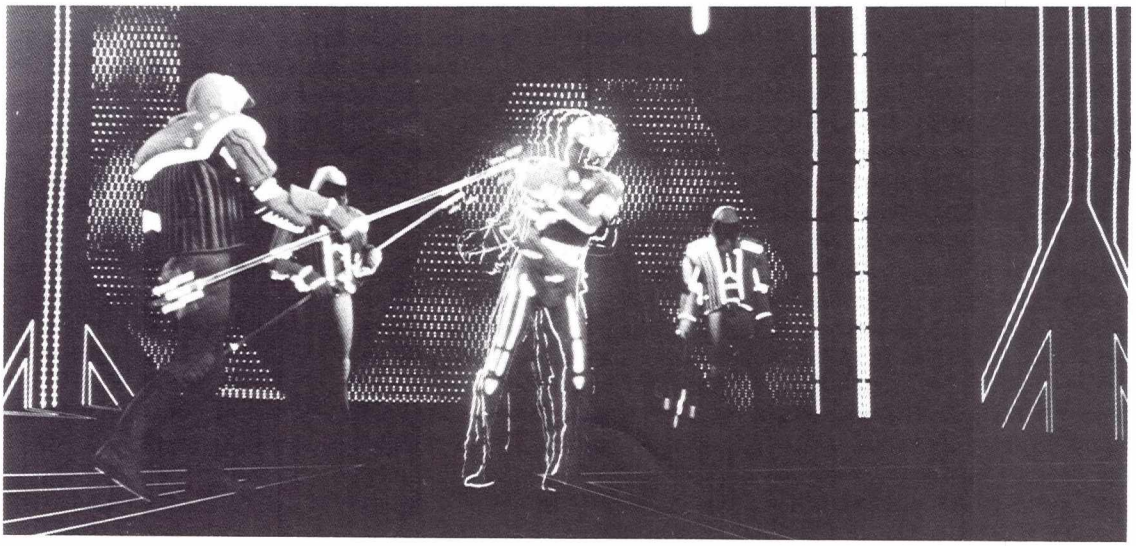
A identificação visual de objetos ainda está no início, mas sua tecnologia progride com rapidez. Quando examinamos robôs industriais (ver p. 281), observamos que grande parte do progresso se realizava na área de reconhecimento de objetos por meio de câmara de televisão e que o robô podia recolher determinada coisa entre várias outras misturadas. O reconhecimento visual significativo depende do tamanho do vocabulário visual, que também é função da capacidade de memória e da potência de processamento. Quanto à síntese de alimentos, talvez não seja possível fazer com que uma refeição se pareça com bife e batatas fritas, mas é possível fazê-la com sabor e cheiro semelhantes, embora os computadores não possam por enquanto criar comida a partir de seus elementos constituintes.

Nem todos os escritores chegam a esses limites de atribuição de capacidades assombrosas a suas máquinas de ficção. John Brunner, no romance de ficção científica *Stand On Zanzibar*, publicado em 1969, descreve o mundo no ano 2010, quando os problemas de superpopulação e fome teriam atingido nível crítico. O computador que ele descreve, o Shalmaneser, possui considerável capacidade de memória e velocidade de processamento (está ligado em linha direta com todos os aparelhos de televisão da Terra), mas sua linguagem para perguntas é muito semelhante à que hoje usamos:

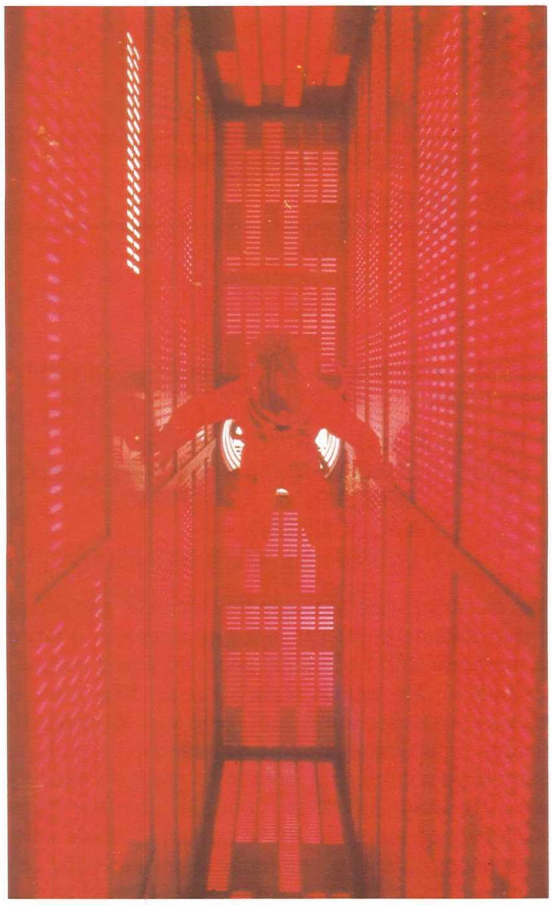
PROGRAMA REJEITADO
Q motivo da rejeição
ANOMALIAS NOS DADOS BÁSICOS
Q definir Q especificar
DADOS NAS CATEGORIAS A SEGUIR SÃO INADMISSÍVEIS. DAR HISTÓRICO DA CULTURA DE INTEGRAÇÃO SOCIAL E CULTURAL
Q admitir dados como fornecidos
QUESTÃO SEM SENTIDO E NÃO OPERACIONAL

Brunner deu-se a um longo trabalho quanto ao emprego da linguagem quase normal num contexto que o usuário do computador reconheceria como resposta de um sistema operacional. Outras de suas predições também são convincentes — o romance recebeu importantes prêmios de ficção científica.

Em filmes e livros mais recentes, os computado-



**Fantasia de Disney**
O filme *Tron*, da Walt Disney, passa-se tanto no mundo real quanto no interior do computador. O mundo externo tem como personagens engenheiros de software, analistas de sistemas e outros técnicos de computação; porém no interior do hardware as unidades individuais do programa e o sistema operacional tornam-se os personagens, e a arquitetura da máquina é o cenário em que decorre a ação.



**Onipotência no espaço**
O computador HAL — Heuristically programmed ALgorithmic, ou ALgoritmo Heuristicamente programado — do filme *2001 — Uma Odisséia no Espaço*, de Arthur C. Clarke, exemplifica a onipotente máquina de computação, muitas vezes encontrada em histórias de ficção científica.

res se tornaram mais do que parte da mobília ou personagens secundários. Integram, com freqüência, o próprio enredo. Um exemplo clássico é *Tron*, da produtora Walt Disney. Já nos referimos a esse filme excepcional como exemplo da animação computadorizada (ver p. 183). Seu nome é derivado de um sistema operacional mnemônico — TRace ON, ou rastreamento.

Há obras de ficção que não mencionam computadores, mas deixam ao leitor a certeza de que, sem equipamentos de computação de grande potência, a situação descrita nunca poderia existir. As mais conhecidas são: *1984*, de George Orwell, e *Admirável mundo novo*, de Aldous Huxley. Os dois livros têm como cenário um futuro que poderia ser vivido por seus autores, num mundo submetido a um pequeno grupo que reprime a população.

Torna-se difícil fazer uma análise exaustiva das caracterizações do computador na ficção, mas alguns trabalhos apresentam idéias criativas e originais. *Giles Goat-Boy*, de John Barth, é um bom exemplo. Trata-se de um extenso romance (812 páginas na edição em brochura) que parte do pressuposto de ter sido escrito por um supercomputador, o WESCAC, que relata um incidente havido com seu "autor".

Não é apenas na literatura de ficção que se encontram idéias criativas sobre computadores. E dos milhares de obras especializadas, não-ficcionais, uma sobressai pela qualidade da narrativa: *Soul Of A New Machine (Alma de uma nova máquina)*, de Tracy Kidder. Trata-se do relato do desenvolvimento do Eagle, da Data General, um microcomputador de 32 bits. Embora a história seja a dos engenheiros envolvidos no projeto, o personagem principal, de certo modo, é o próprio computador.



**Jogos de guerra**
Inadvertidamente, um jovem usuário de micro, tentando se comunicar com um amigo pela rede pública de telefones, viola o principal computador do sistema de defesa da OTAN. Supondo que se trata de um videogame, começa a jogar, sem perceber que deu início à Terceira Guerra Mundial.

# Autor original

## Pode-se desenvolver programas de computação que geram outros ou corrigem erros de codificação humana.

"Se os computadores são tão espertos, por que seres humanos devem programá-los?" Programadores experientes tendem a descartar questões como essa, formulada por um principiante cético. A dúvida, porém, não é tola como parece. Hoje se faz muita pesquisa de desenvolvimento de programas que podem dar origem a outros programas e de sistemas operacionais que objetivam corrigir erros em codificações escritas por seres humanos.

"ERRO SINTÁTICO?" é uma mensagem encontrada com freqüência por usuários de microcomputador. Pode ser irritante, porque fornece pouca informação. O compilador de um computador de grande porte fornecerá, em geral, muito mais informação quanto às características do erro encontrado. Por exemplo, a mensagem de erro pode se apresentar desta forma:

```
1090 LET A=(C*2+F$)*((FG-C)*TH+1))
ERROS: 1) CONCORDANCIA — VARIAVEL ALFANU-
          MERICA F$
          NAO ADMISSIVEL
       2) ULTIMO PARENTESE NAO ESPERADO
```

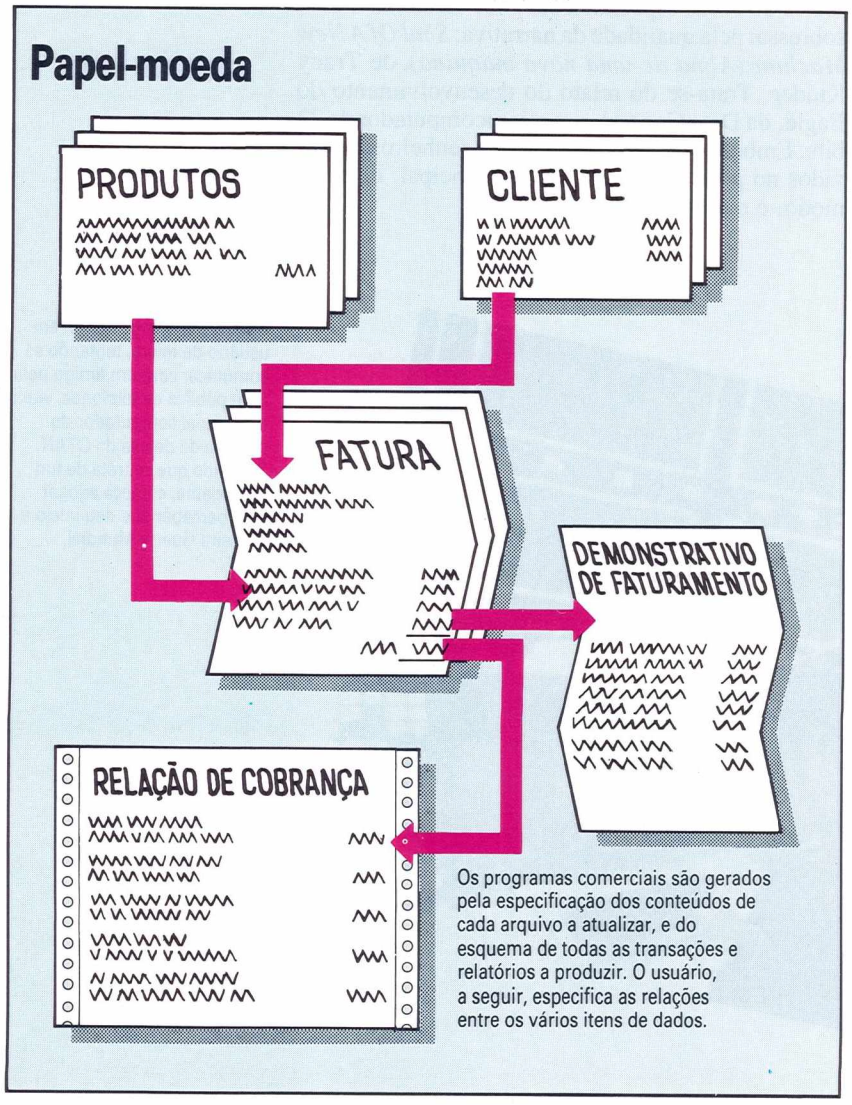
Não há razão séria para que tais técnicas não sejam empregadas num interpretador de micro — o custo da memória ROM adicional necessária para o armazenamento das rotinas seria mínimo. Mas poucos microcomputadores utilizam, pelo menos, procedimentos superficiais de supervisão de erro: a maioria nem chega a conferir a sintaxe do código à medida que é fornecido. Pode-se, contudo, comprar chips adicionais de memória ROM ou cartuchos de software acopláveis que ampliarão o alcance dos comandos disponíveis, sobretudo os relacionados com desenvolvimento e depuração de erros dos programas. Esses comandos BASIC abrangem:

HELP — Imprime a linha do programa e destaca a posição exata do caractere onde terminou a execução do programa. Isto, de modo geral, mas nem sempre, indicará a fonte do erro de sintaxe.

DUMP — Imprime uma lista de todos os nomes de variáveis, com seus conteúdos, que estiverem em uso no programa. É muito útil para se saber quanto o programa avançou em sua tarefa, antes da ocorrência do erro.

TRACE — Apresenta numa janela no canto da tela o número (ou números) da linha que está sendo executada, à medida que o programa se processa. Isso ajuda o usuário a traçar o fluxo do programa e assegura, entre outras coisas, que as sub-rotinas sejam executadas na ordem desejada.

O desenvolvimento de programas que permitam ao computador corrigir a codificação humana não é, em geral, tarefa simples. Porém, no caso de alguns erros, mostra-se muito fácil. Sabemos, por exemplo, que todas as linhas de programas devem começar por uma palavra-chave em BASIC (embora algumas máquinas admitam a não utilização da palavra LET). Assim, se uma linha se inicia pelo comando PRUNT ou PRONT, é fácil fazer com que diga PRINT. Neste curso de programação em BASIC, examinamos o conceito de combinação embaralhada (algoritmos para escolha da combinação mais próxima de qualquer frase) e isso também pode ser utilizado em palavras-chaves do programa. Da mesma forma, o interpretador pode apenas incluir uma lista de tipos de erros mais freqüentes de digitação com seus equivalentes corretos. Para maior segurança, seria con-



veniente que o computador conferisse com o operador quaisquer alterações feitas.

Mas, afora esses procedimentos elementares, a correção automática se torna bem mais difícil. No exemplo dado, F$ corresponde a um erro de impressão de F ou FS ou F4? Ou de algo inteiramente diferente? Se você mostrasse a listagem completa a outro programador competente, ele seria capaz de identificar as falhas e fazer as correções. Para tanto usaria dois critérios na tomada de decisão: o contexto no qual a linha do programa apareceu e sua própria experiência.

De modo bastante estranho, essa técnica tem sido mais utilizada na correção do texto em português ou inglês do que na verificação do código do programa. Um pacote de verificação de erros de grafia, por exemplo, trabalha o texto e destaca as palavras que não corresponderem às entradas em seu dicionário de cerca de 50.000 palavras, retidas em disco. A maioria desses pacotes dispõe de recursos para aprender novas palavras (como os nomes de empresas ou de pessoas) e as incluem em seus dicionários. Os pacotes mais elaborados até sugerem a maneira correta de soletrá-las, se uma correspondência aproximada for detectada. Processadores experimentais de palavras empregam os mesmos processos para correções gramaticais e de estilo literário indicando erros ou imprecisões de pontuação, exagerada repetição de palavras no mesmo parágrafo e adjetivos ou advérbios inadequados. Esses pacotes operam pelo exame do contexto de cada sentença e pela referência a uma biblioteca de exemplos.

Um empenho maior, contudo, foi dedicado ao desenvolvimento de sistemas que *criam* programas em vez de apenas *corrigir* os já existentes. Em 1981, surgiu no mercado o software habilmente denominado The Last One (O último). Pretendia ser um programa capaz de desenvolver qualquer outro programa que se desejasse. Em outras palavras, seria o último programa que se precisaria adquirir. Tal pretensão mostrou-se exagerada, embora The Last One fosse um auxiliar muito útil ao desenvolvimento de certos tipos de software, sobretudo para aplicações comerciais. Existem hoje vários produtos desse tipo no mercado, rodando tanto em micros comerciais quanto em alguns de uso doméstico — todos eles genericamente denominados "geradores de programas".

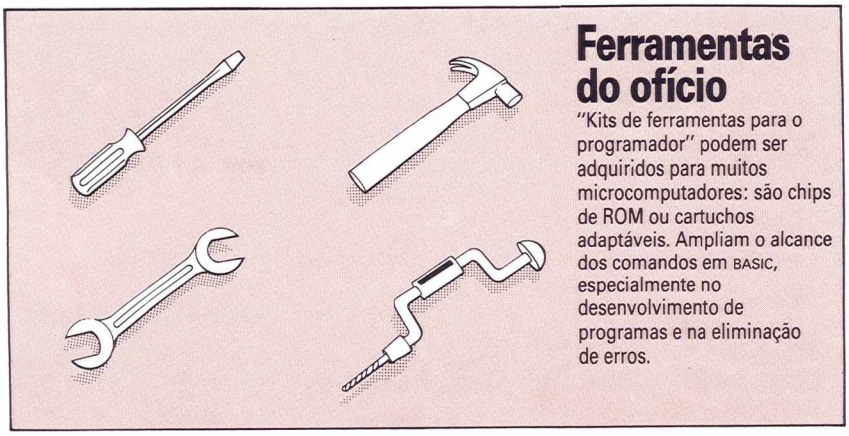
Examinemos agora o princípio básico que fundamenta um programa que pode desenvolver outros programas. Observe este exemplo bem comum:

```
10 PRINT "O QUE VOCE DESEJA QUE O
   PROGRAMA APRESENTE NA TELA?"
20 INPUT A$
30 PRINT "O PROGRAMA E:"
40 PRINT "10 PRINT ";CHR$(34);A$;CHR$(34)
```

Se responder OLÁ à pergunta, o programa (que deve rodar na maioria dos microcomputadores) imprimirá a seguinte linha:

```
O PROGRAMA E
10 PRINT "OLA"
```

Caso se utilize a mesma técnica nas fases de entrada de dados, de cálculo e de saída para a aplicação que se tem em mente, pode-se desenvolver um gerador bem simples de programas. Se todas as questões que ele solicita estiverem formuladas de modo claro, será fácil, mesmo para quem não tem experiência anterior, desenvolver um programa simples com o auxílio desse software.



## Ferramentas do ofício

"Kits de ferramentas para o programador" podem ser adquiridos para muitos microcomputadores: são chips de ROM ou cartuchos adaptáveis. Ampliam o alcance dos comandos em BASIC, especialmente no desenvolvimento de programas e na eliminação de erros.

Os geradores de programas disponíveis no mercado utilizam as mesmas técnicas. A maioria das aplicações comerciais consiste numa combinação de cinco processos distintos: entrada de dados, saída para a tela ou impressora, armazenamento em arquivo, recuperação e cálculo. O gerador precisa dispor de sub-rotinas padronizadas e muito flexíveis para cada um desses procedimentos. Ao lhe pedir que especifique a estrutura exata dos dados que você estará usando, os cálculos feitos com esses dados e os formatos de saída de que você necessitará na tela e na impressora, o gerador vai alterar os valores de certas variáveis de algumas sub-rotinas e depois fazer a conexão para criar um programa.

Embora estejam se tornando mais sofisticados, é improvável que os geradores de programas venham a substituir o programador humano em futuro próximo, pois apresentam sérias limitações. A técnica descrita é adequada a aplicações comerciais em bases transacionais, como contabilidade ou controle de estoque, mas em geral não pode ser empregada no desenvolvimento de processadores de palavras ou programas de jogos. Em segundo lugar, como o gerador de programa recorre a sub-rotinas padronizadas, a listagem resultante nem se aproxima da eficiência (tanto em velocidade quanto em memória utilizada) que teria se fosse desenvolvida por um programador. Além disso, os programas produzidos por geradores não são tão *user-friendly* (fáceis de usar) como os produzidos pelo programador, quando este é um ser humano. Poucas vezes utilizam, por exemplo, os recursos gráficos oferecidos pelas máquinas mais modernas.

Enfim, os geradores de programas disponíveis no mercado conseguem de fato substituir apenas o último estágio da programação — o desenvolvimento do código. O usuário ainda tem de elaborar a forma exata da saída e entrada de dados de que precisa. Em geral, os estágios que antecedem a programação são os mais difíceis e exigem habilidade específica, diferente daquela necessária à programação. A maior parte das grandes empresas emprega especialistas, os analistas de sistemas, para especificar os programas necessários; e programadores convertem essas especificações em códigos.

# Fora do espectro

**Alguns micros podem ser transformados em máquinas sofisticadas, mediante uma variedade de acessórios. É o que acontece com o Spectrum.**




Quando lançado, no começo de 1982, o Sinclair Spectrum foi considerado um aparelho revolucionário em matéria de preço e desempenho. Em 1983, o primeiro ano completo de produção, suas vendas (600.000 unidades) chegaram a mais da metade dos microcomputadores comercializados na Grã-Bretanha, o que surpreendeu os próprios fabricantes. O Spectrum representa um grande progresso em relação ao modelo anterior da Sinclair, o ZX81, com 16 ou 48 Kbytes de RAM como padrão. Dispõe de oito cores para margem, fundo e texto, e uma capacidade limitada de alta resolução gráfica; um teclado melhorado, mas ainda insatisfatório; e capacidade de gerar sons simples. Mas essa variedade de funções não impediu que fabricantes independentes produzissem inúmeros acessórios. A própria Sinclair entrou em ação, adicionando a seu produto armazenamento em larga escala, na forma de microdrives e também interfaces para cartuchos ROM encaixáveis e joysticks.




**Monitor de TV**
Como o Spectrum não tem funções de saída para um monitor, é difícil obter gráficos de alta qualidade. Este é o Profeel, da Sony, com o receptor separado do monitor

**Impressora serial**
Além da impressora ZX, é possível ligar uma do tipo matriz de pontos ou margarida ao Spectrum por meio de uma interface RS232 ou Centronics. A Interface 1 comporta dispositivos seriais

**Sintetizador de som**
O Trichord, da Petron: um bom exemplo de sintetizador para o Spectrum. É uma unidade de três vozes que oferece até 6.134 acordes de três notas quando usado com o Spectrum de 48 Kbytes

**Gravador cassete**
Mesmo com o microdrive acoplado, o Spectrum pode transmitir e receber dados de um gravador cassete

**Sintetizador de voz**
O Cheetah Sweet Talker é um sintetizador que utiliza sistema alofônico simplificado para produzir palavras mediante a combinação de suas sílabas. Dispõe de 63 alófomos e quatro intervalos de tamanhos variados

**Interface 2**
Muitas empresas produzem interfaces para controle de joysticks programáveis, mas o modelo da Sinclair tem uma tomada para ROM que facilita a instalação e o uso de jogos e linguagens alternativas

**Impressora ZX**
A fim de produzir uma impressora de baixo custo, a Sinclair produziu uma unidade de impressão por sensibilização de papel aluminizado

**Teclado**
Embora o teclado padrão do Spectrum constitua um grande melhoramento em relação ao do ZX80 e do ZX81, ainda não oferece "sensibilidade" ao toque

**Interface 1**
Projetada especificamente para servir o microdrive, a Interface 1 contém, ainda, circuitos que permitem a ligação de impressora, plotters e modems; além disso, permite que diversos Spectrum sejam ligados entre si, em rede local

# Fim específico

## Uncommitted Logic Arrays (ULAs) podem controlar as funções do microcomputador, independentemente de CPU, ROM e RAM.

Entre os muitos avanços da eletrônica, resultantes do rápido desenvolvimento do microcomputador, um dos mais importantes foi o aperfeiçoamento de um tipo de chip chamado Uncommitted Logic Array (matriz lógica sem funções determinadas) ou ULA. Embora distante do conhecimento público, essa revolução silenciosa vem acontecendo e está tão avançada que tornou possível a construção de computadores mais sofisticados e de outros dispositivos com apenas quatro componentes principais: a CPU, um pouco de RAM, um pouco de ROM e — para unir as três — uma ULA.

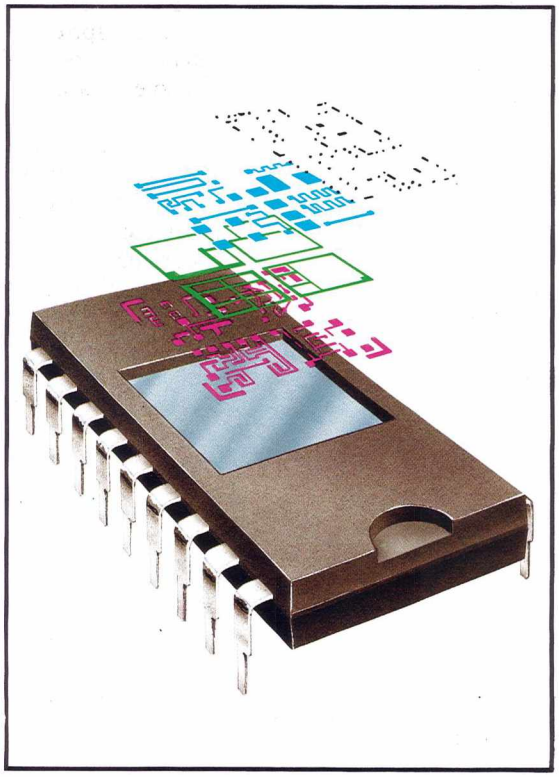
Como o próprio nome sugere, a ULA corresponde a um grande número (uma matriz) de portas lógicas que, de início, não têm funções específicas, mas podem ser modificadas para executar praticamente qualquer operação que o designer desejar. A ULA é considerada um desenvolvimento da ROM, uma vez que o conteúdo de ambas só se vê especificado pelo fabricante do chip, e não pelo usuário.

Antes de ser "programada", uma ROM ou ULA consiste apenas em grande número de circuitos ou células eletrônicas simples, que não estão ligadas e portanto não executam nenhuma ação. Todos os chips são constituídos de camadas sobrepostas de materiais semicondutores (ver p. 122). A camada final compõe-se de material condutor e forma conexões entre as diversas células. A ampla variedade de interligações possíveis dá flexibilidade à ULA; e, embora as células sejam bastante simples, consistindo num par de transistores, ou uma simples resistência, elas podem ser todas interligadas através da última camada, para montar circuitos complexos, tais como flip/flops (ver p. 305).

Constróem-se esses circuitos, chamados módulos, até com menos de meia dúzia de células, e já que uma ULA grande tem milhares delas, os módulos propriamente ditos podem se interligar para montar circuitos complexos, tais como máquinas registradoras, calculadoras e circuitos de tempo. As funções desempenhadas por esses circuitos são, com freqüência, executadas num microcomputador por uma coleção de chips lógicos de uso geral.

Uma ULA pode ser programada para desempenhar variedade muito diversificada de atividades. É possível fazer qualquer ULA sintetizar sons, controlar a exposição, o foco e o motor de uma câmara fotográfica ou fazer a maior parte do trabalho num termômetro digital. Além da ULA, quase nenhum circuito externo se faz necessário — apenas a bateria, o interruptor e alguns sensores ou botões de controle.

Os computadores são muitas vezes empregados para executar o projeto da camada que interliga as células da ULA. O minicomputador como um DEC PCP11/23, executando programas do tipo CAD — Computer Aided Design —, monta de início um diagrama codificado da lógica desejada. Então, o sistema traça e codifica um mapa do layout planejado. Tudo isso ocorre num terminal gráfico, e uma cópia impressa do projeto pode ser produzida num plotter, se houver conveniência.



**Plano mais elevado**
Todos os chips semicondutores são formados por camadas de materiais semicondutores, gravados um a um para criar os elementos do circuito. A camada final determina a conexão entre os elementos. A ULA consiste numa matriz de elementos lógicos combinados para formar um circuito lógico complexo.

## Máscara óptica

Concluído, o design é trasmitido para um computador maior, que verifica se o plano é aceitável, compara-o com o projeto lógico original e certifica-se de que não contém erros graves. Então, ele é submetido a outro programa que simula o circuito resultante, usando um programa de teste preparado pelo usuário. Quando o projeto estiver terminado, o computador produz o trabalho de acabamento para a máscara óptica usada na execução da última camada.

As possibilidades da ULA são variadas. A idéia de colocar grande número de circuitos simples em silício e deixar o usuário decidir como eles devem atuar uns sobre os outros mostra-se tão fascinante que poderia resultar na execução de um número maior de circuitos. Contudo, no atual nível da tecnologia, as ULAs só são econômicas quando há necessidade de no mínimo alguns milhares de circuitos elétricos idênticos. PROM (Programmable Read Only Memory), EPROM (Erasable PROM), EEPROM (Electrically Erasable PROM) e EAROM (Electrically Alterable ROM) são, todas elas, alternativas para a ROM, programável pelo usuário com equipamento adequado. Talvez, em breve, também apareçam alternativas para a ULA programáveis pelo usuário.

# Intérprete de papéis

**A simulação tem revelado ótimas aplicações pedagógicas para microcomputadores, e são vários os programas disponíveis.**

## Ventos

Este programa de simulação, distribuído pela Longmans inglesa, coloca você na posição de comandante de uma antiga caravela (como aquelas empregadas por Cabral). Na tela aparece um mapa-múndi, onde sua embarcação é representada por um ponto que navega de acordo com a bússola — N, L, SO etc. Você escolhe a data em que será lançada ao mar, o porto de partida e o de chegada. A velocidade da nau depende da direção dos ventos predominantes; esta informação aparece no rodapé da tela, junto com a posição da nau expressa em latitude e longitude, os ventos da região (vento oeste, por exemplo), a data, a distância percorrida até o momento e a duração total da viagem.

Tomemos a rota mais direta de Londres ao Rio de Janeiro, com partida no dia 1.º de janeiro. Rumando para o sul, fazemos bom progresso aproveitando o vento oeste do Atlântico, até chegarmos à linha do Equador. Aí ficamos retidos por três dias devido à calmaria. Afinal, sopra um vento sudoeste, mas surge um problema — como podemos navegar rumo sudoeste com um vento sudoeste? A solução é navegar em ziguezague (ou "bordejar") primeiro para o sul e depois para oeste, até alcançar o Rio de Janeiro, depois de 207 dias de viagem, percorrendo 9.620 milhas (equivalentes a 15.480 km).

Outras viagens implicam inúmeros perigos: furacões, gelo polar e naufrágio constituem alguns dos riscos a que você e sua nau estão sujeitos. Esse programa de simulação pode ser usado de várias formas. Na mais simples, como um jogo para ensinar às crianças os pontos cardeais e colaterais. Numa aula de geografia, além de desenvolver o conhecimento do globo terrestre, os alunos podem examinar as diversas zonas de ventos e ter uma idéia de otimização de rota.

## Simulação de vôo

Trata-se de uma versão para microcomputador de conhecido jogo de fliperama, onde o operador é o piloto de um pequeno avião. Você encontra na tela o que veria se estivesse na cabine de comando de um avião, inclusive o painel de instrumentos, onde aparecem mostradores, marcadores e luzes, que devem ser observados com atenção. Quatro teclas com setas representam o manche do avião; os demais controles (potência, trem de pouso etc.) são operados mediante outras teclas.

No jogo de fliperama seria impossível, mas nesse programa você tem a oportunidade de se familiarizar com os controles, começando com o avião em pleno ar. Para saber onde se encontra o avião, a tela mostra um mapa com a posição do aparelho, sinais de navegação e pistas de pouso e decolagem. A melhor forma de conduzir o aparelho é "fechar" num sinal de navegação e então inclinar o avião lateralmente em curva até que ele fique alinhado com o sinal. Isso é mostrado pelo ponto que pisca no "Relógio RDF" e que fica rodando até chegar ao topo do mostrador. Então você voa em linha reta até o sinal. Usando esse método, será fácil chegar ao aeroporto onde você tentará pousar.

A aterrissagem constitui a manobra mais difícil. Você precisa estar alinhado com a pista e aproximar-se na velocidade, na altura e no ângulo certos. E você ainda pode cometer um erro fatal — esquecer de baixar o trem de pouso!

## Simulação fisiológica

Você faz o papel do cérebro humano e sua função será manter o corpo vivo por apenas 50 minutos! O programa simula as diversas mudanças fisiológicas (temperatura do corpo, perda de água etc.) que ocorrem quando o corpo exerce alguma atividade. A primeira coisa a fazer é entrar com a idade e o sexo da pessoa e a atividade que você quer desempenhar. Dormir torna-se a atividade mais fácil de simular, pois o corpo despende pouca energia. Outras atividades consistem em andar, escalar um rochedo e — a mais exaustiva de todas — correr.

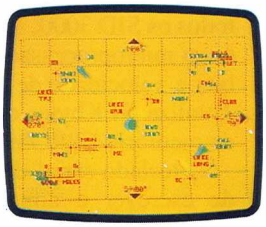
Os parâmetros que se controlam são os seguintes: o volume e a taxa da respiração e a taxa de transpiração. Você escolhe os valores iniciais — por exemplo, volume de respiração: 2,5 l; taxa de respiração: quinze inalações por minuto; taxa de transpiração: 3 g por minuto. E aí a simulação começa.

Na tela aparecem cinco gráficos representando as diferentes funções do corpo e um relógio. À medida que o tempo passa, os gráficos mostram como o corpo desempenha a atividade escolhida, e você deve evitar que qualquer um deles ultrapasse o nível de perigo. Se, por exemplo, a temperatura do corpo ficar muito alta, você suspende a atividade por um instante e aumenta a taxa de transpiração para combater a febre. Se você não tiver êxito e um dos gráficos cruzar a linha de perigo, pode receber o lacônico e definitivo diagnóstico: "A PESSOA ESTÁ MORTA". A Heinemann Educational Software produziu esse programa para o micro inglês BBC Model B.

Ventos




Simulação de vôo



Simulação fisiológica



# SID 3000

## Compactos e versáteis, os equipamentos do sistema SID 3000 permitem ao cliente escolher, entre quatro modelos, o que melhor atende a suas necessidades do momento.

SID 3000 é a designação de uma série de quatro microcomputadores de uso profissional e empresarial totalmente projetada e construída no Brasil pela SID — Sistema de Informação Distribuída S/A.

O sistema operacional da série é compatível com CP/M, o que permite o uso de uma enorme variedade de programas já desenvolvidos em linguagem COBOL, BASIC compilado, BASIC interpretado, Assembler e DBASE.

A Unidade Central de Processamento (CPU) de todos os SID 3000, localizada no gabinete de vídeo, é um microprocessador do tipo Intel 8085A, com 8 bits e velocidade de processamento de 2,76 MHz. Totaliza 64 Kbytes de memória RAM e 16 Kbytes de memória EPROM.

O teclado serial dos equipamentos tipo máquina de escrever é alfanumérico, numérico reduzido (com os algarismos de 0 a 9) e de funções. O bloco numérico foi disposto separadamente, para facilitar as operações matemáticas.

Um cabo espiralado de 1 m de comprimento, do tipo usado para telefones, liga o módulo do teclado ao monitor de vídeo, permitindo, assim, a operação do teclado a distância. O monitor de vídeo é monocromático e sua tela de fósforo verde tem 12 polegadas.

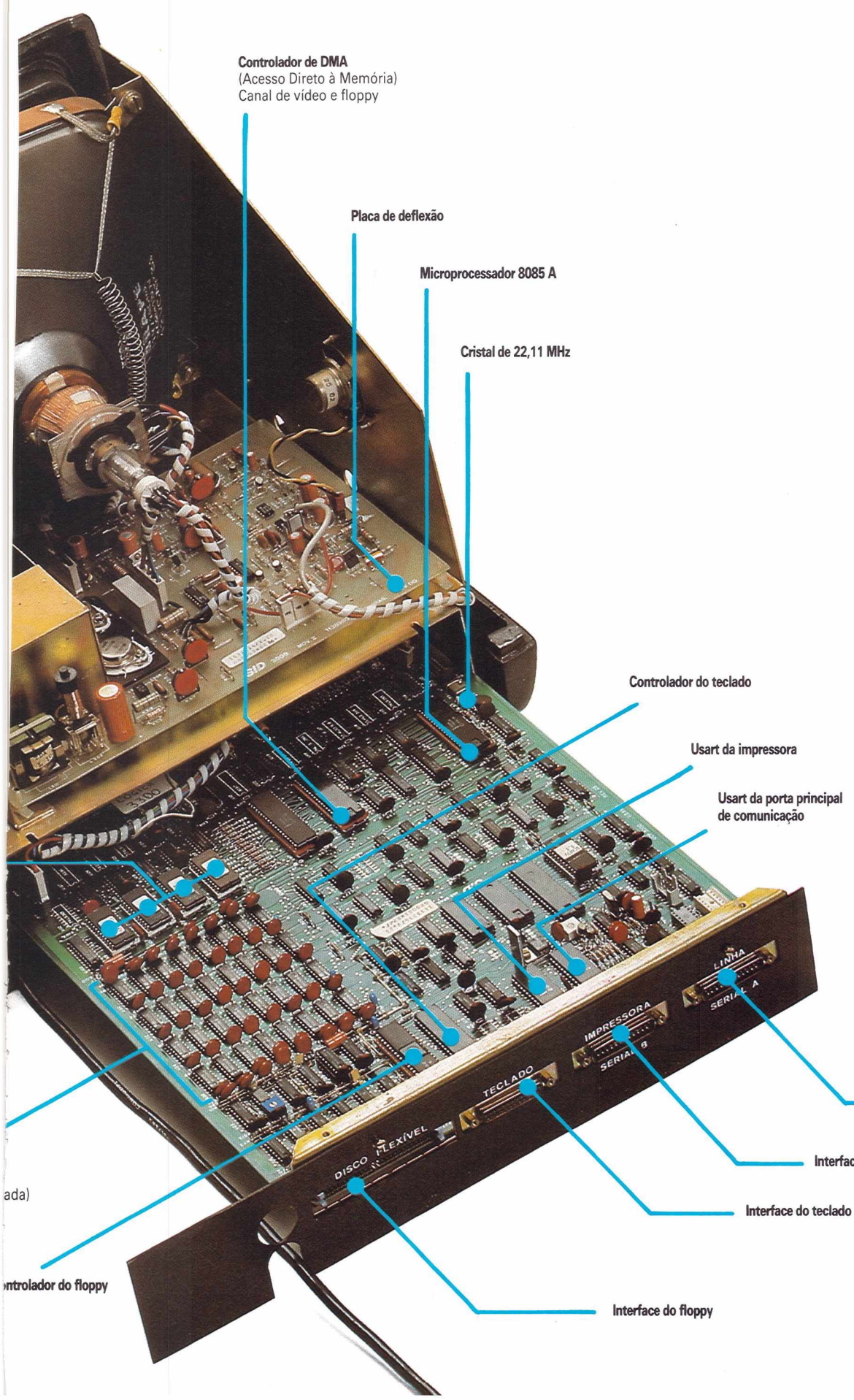
Esse monitor é controlado por uma interface gráfica que opera somente em modo alfanumérico (25 linhas × 80 colunas) e não possui capacidade gráfica ou semigráfica. Completando cada equipamento, há uma placa de funções múltiplas que inclui interface para impressora matricial ou linear, para discos flexíveis e, em um dos modelos, para discos rígidos.






Assim, o SID 3300 tem uma unidade DUAL de discos flexíveis de 5 1/4 polegadas e interface de comunicação RS232 C. O SID 3801 aceita até duas unidades DUAL de discos flexíveis face simples de 8 polegadas, com duas interfaces de comunicação RS232 C e RS422. O SID 3802, com as mesmas características de seus irmãos de série, tem ainda a vantagem de aceitar discos flexíveis face dupla de 8 polegadas.

O SID 3900, por fim, permite o uso de discos flexíveis face simples ou dupla de 8 polegadas e chega a aceitar até duas unidades de discos rígidos de 10Mb cada um. Com isso, aumentam ainda mais as possibilidades de escolha de programas pelo usuário e, portanto, a versatilidade do equipamento.




## SID 3000

**MICROPROCESSADOR**

8085A

**CLOCK**

2,76 MHz

**MEMÓRIA**

64 Kbytes de RAM de 16 Kbytes de EPROM.

**VÍDEO**

Monitor de 12", em fosforo verde, monocromático. Interface gráfica — modo alfanumérico (80 × 25).

**TECLADO**

Tipo máquina de escrever, com 99 teclas; teclado numérico separado.

**LINGUAGEM**

COBOL

**PERIFÉRICOS**

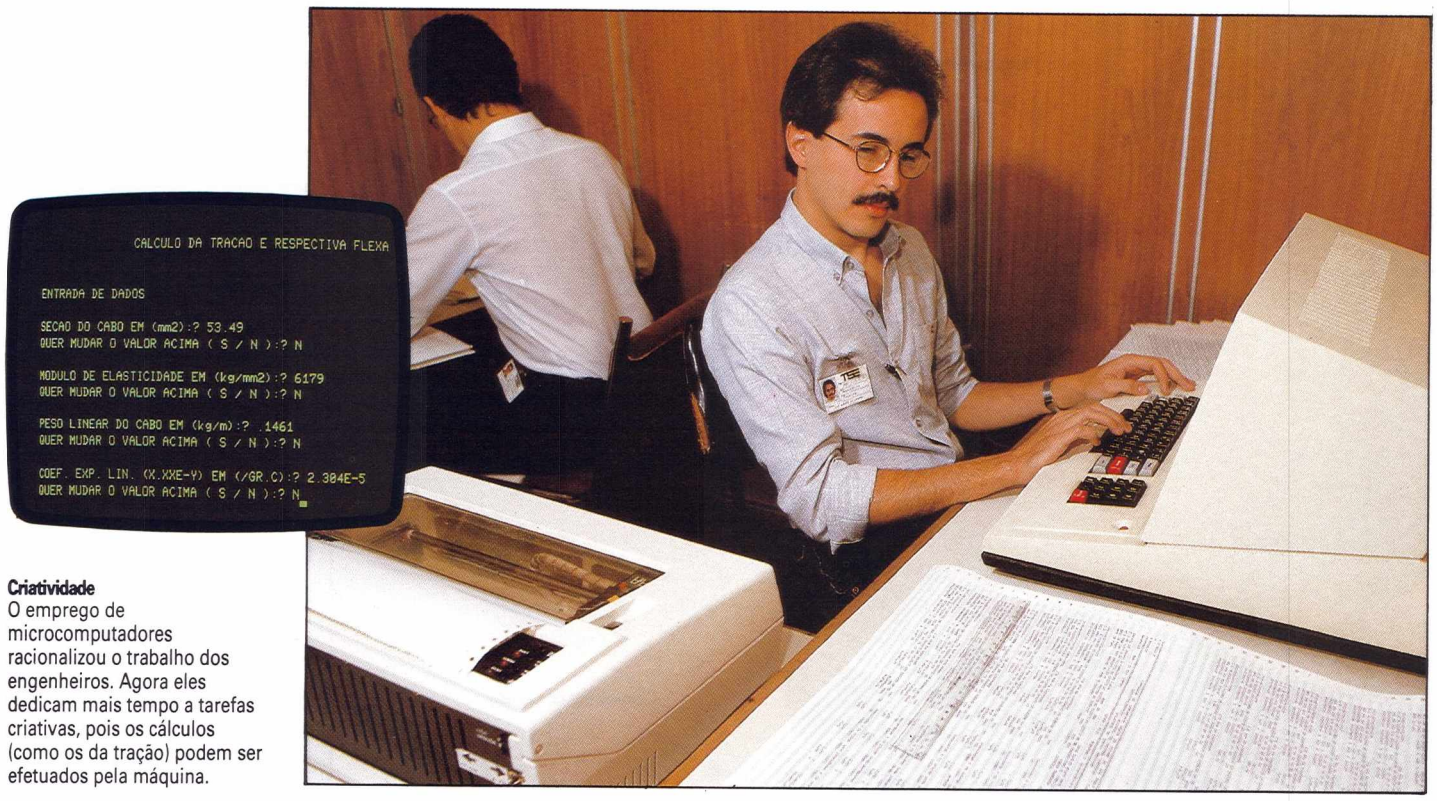
Unidades de disco (5 1/4" e 8", densidade simples e dupla; disco rígido), impressora, interface serial RS232 C ou RS422.

**DOCUMENTAÇÃO**

Todos os equipamentos da série SID 3000 dispõem de um completo manual de instalação e operação e vêm acompanhados de um catálogo com todos os fornecedores de software aplicativo.

# Mestre-de-obras

## O micro gerencia construtoras e substitui, na mesa do engenheiro, lápis, borracha e calculadora de bolso.



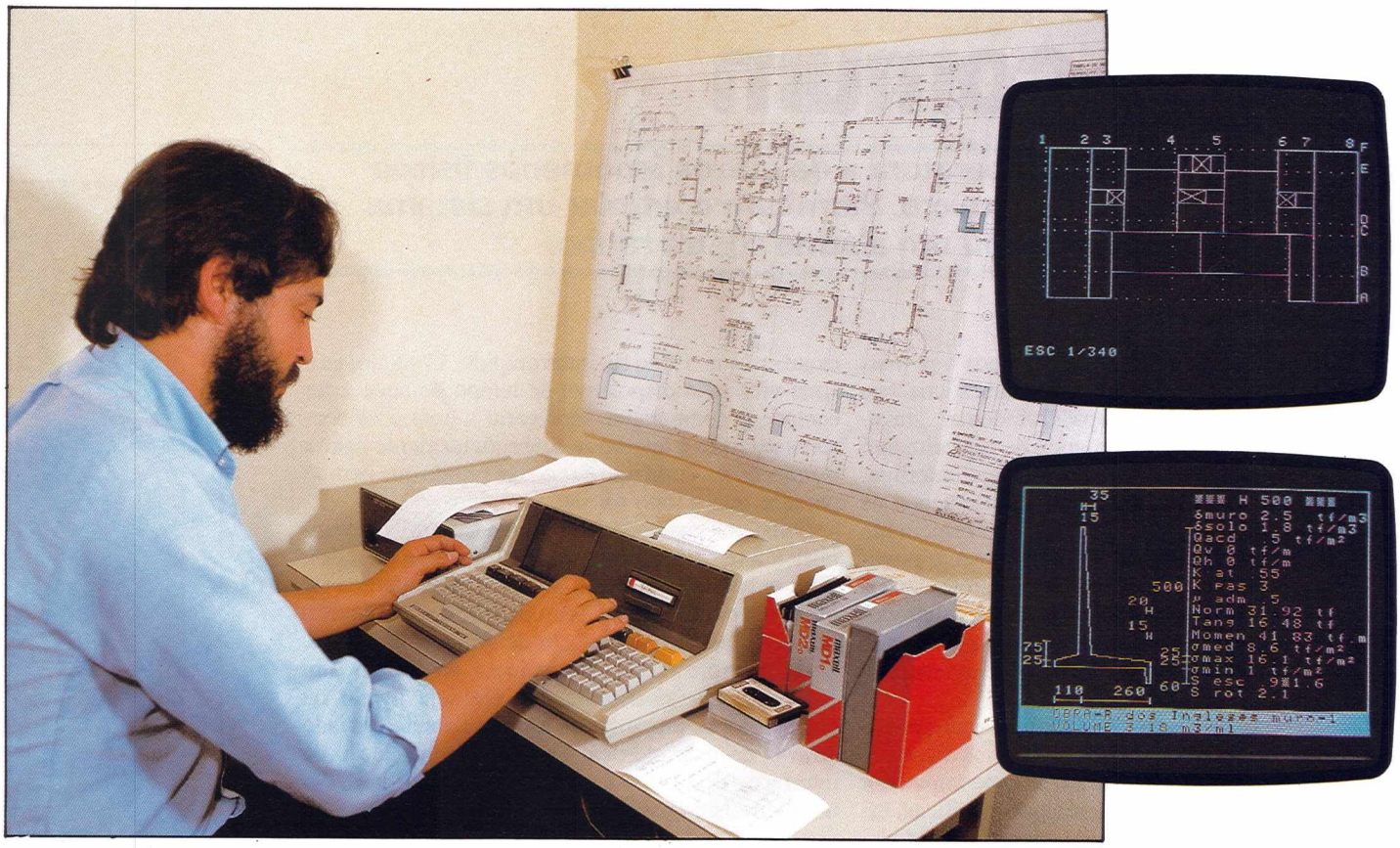

**Criatividade**
O emprego de microcomputadores racionalizou o trabalho dos engenheiros. Agora eles dedicam mais tempo a tarefas criativas, pois os cálculos (como os da tração) podem ser efetuados pela máquina.

Um microcomputador em cada canteiro de obra, controlando material, custos, cronograma e folha de pagamento. Experiências como essa deixaram de ser ficção ou profecia: acontecem com freqüência cada vez maior no dia-a-dia de muitos escritórios de engenharia e de construtoras. ''Entramos definitiva e irreversivelmente na era da informática.'' Essa frase do engenheiro Airton Mendes Rodrigues, assessor de sistemas de engenharia da TSE — Técnicas e Serviços de Engenharia, de São Paulo —, reflete uma tendência generalizada.

Quatro microcomputadores instalados no escritório elaboram cerca de 35% dos cálculos, o que engloba quase todos os cálculos de estruturas metálicas e de concreto. ''Na ponta do lápis mesmo são executados apenas os cálculos não padronizados, como os de hidráulica para diferentes estações de tratamento de água'', esclarece o engenheiro Mendes Rodrigues. A implantação do sistema de microcomputadores da TSE, em 1982, visava a atender apenas a área de estruturas; no entanto, acabou motivando a criação da ASE — Assessoria de Sistemas de Engenharia —, responsável pelo desenvolvimento de um plano de utilização dos micros dentro da empresa, que inclui a formação, por meio de cursos, de pessoal especializado e a elaboração de programas para as diversas áreas. Os investimentos, estimados em agosto de 1984 em mais de 55 milhões de cruzeiros, tiveram retorno positivo na forma de economia de tempo, custos e energia: ''Cálculos que levávamos quase quinze dias para fazer, hoje fazemos em cerca de uma hora'', ressalta Luís Alberto Villaverde, programador formado pelos cursos da ASE.

A importância da redução do tempo é menos óbvia do que se supõe, como explica Mendes Rodrigues: ''Às vezes, a economia de tempo possibilita a decisão rápida em momento crítico, quando não se pode perder nem um minuto''. Para ele, em síntese, a definição de engenharia é ''construir bem da maneira mais econômica possível''. Nesse contexto, a utilização do computador possibilita a análise de maior número de soluções possíveis para cada problema: ''Paramos de fazer contas e passamos a pensar mais''.

Com essa mesma intenção, ou seja, reduzir o trabalho repetitivo e ganhar tempo, o engenheiro José Martins Laginha Neto, do GTP — Grupo Técnico de Projetos —, instalou em 1981 um microcomputador em seu escritório. Como a maior parte dos profissionais do setor, ele começou com programas de cálculos e análises estruturais de pórticos e grelhas, solucionando o dimensionamento de vigas, lajes e pilares. Em 1983, o GTP passou a desenvolver programas de cálculo de pavimentos de edifícios; no ano seguinte, realizou estudos na área de desenho e pôs em prática um programa de elaboração de plantas de detalhamento.




Os estudos com desenhos representam uma sofisticação do processo, além de auxiliar por sua maior rapidez a detecção de erros. A entrada de dados faz-se de maneira gráfica, e não numérica. Ou seja, o usuário visualiza na tela um desenho, como se ele estivesse sendo elaborado a mão.

A experiência da CAVO — Companhia Auxiliar de Viação e Obras — confirma a tendência do setor em direção ao uso sistemático de computadores. Utilizando microcomputadores na área de orçamento e acompanhamento financeiro de obras desde o início de suas atividades, a CAVO criou um programa de cadastramento de quase todas as composições unitárias para serviços de construção civil, totalizando cerca de 2.000 itens. Eles englobam as áreas de trabalhos em terra, alvenaria, estruturas, instalações hidráulicas e elétricas. Outro cadastro armazena os preços unitários desses itens. As vantagens do pacote são evidentes. "Não é apenas questão de tempo", afirma o engenheiro Fernando Leite de Moraes, dessa empresa, "mas a possibilidade de montar vários orçamentos e refazê-los com diferentes alternativas. Isso, em geral, não se faz com lápis e papel." Sem falar, ainda, na redução do volume de papel empregado e do tempo (hoje a CAVO prepara um orçamento em cerca de 20 minutos, contra os três dias exigidos pelo método tradicional), o que é vital no momento de uma concorrência.

O engenheiro Airton Mendes Rodrigues, da TSE, diz que se verifica também uma mudança no relacionamento cliente/empresa, graças à simples presença de um computador: "Um serviço mais rápido e confiável melhora muito a imagem da empresa". Tese compartilhada por Laginha Neto, que aponta, no entanto, os riscos da má operação da máquina pelos profissionais: "É um mito achar que o computador resolve tudo. A participação de um profissional ainda é grande e, se ele não souber manipular o equipamento, ou não conhecer o programa, o resultado pode ser o contrário do esperado".

Por isso, não chega a ser tão simples, como muitas pessoas pensam, a implantação de um sistema de computadores no setor da construção. Antes, é preciso conhecer bem a rotina de cada escritório e o equipamento desejado. Os microcomputadores, por exemplo, são mais vantajosos em obras complexas, com muitos itens a ser orçados ou acompanhados. Em obras mais simples — uma rede de esgotos, por exemplo —, as vantagens de sua utilização são menores. A convivência com os computadores pode ser dificultada ainda pela necessidade de um programa mais específico. "Em geral", afirma Laginha Neto, "a maioria dos usuários utiliza apenas 20% da capacidade de seu equipamento, por falta de software." Outros apontam a falta de vivência da maioria dos programadores na área de engenharia. Às vezes, é preciso fazer tantas adaptações, as alterações nos programas são tão amplas, que eles se tornam inúteis. Neste caso, as empresas de construção desenvolvem elas próprias seus programas. A CAVO criou cadastros de fornecedores de materiais e de clientes, e tabelas de órgãos públicos. A TSE ampliou seus programas nas áreas de documentação, plantas, requisição de material etc. E ambas as empresas tinham como plano a instalação de um microcomputador em cada canteiro de obra, acompanhando o processamento de faturas e folha de pagamento, controlando material, custos e horas de serviço dos operários, entre outras atividades.

"A maior desvantagem de um micro", conclui jocosamente um dos engenheiros, "é não se poder convidá-lo para um chopinho."

**Bem e depressa**
Os principais elementos da planta são visualizados na tela, e um programa especialmente desenvolvido calcula até as estruturas empregadas na concretização do projeto. A computação tem auxiliado de forma crescente a tarefa da engenharia: construir bem e depressa.

# Senso comum

**Sensores de luz, de temperatura e de outros fenômenos físicos podem ser ligados a um micro. Permitem controlar um sistema de aquecimento central ou um alarme contra roubo.**

Os microprocessadores estão sendo cada vez mais usados em diversos aparelhos eletrodomésticos, tais como máquinas de lavar roupa, torradeiras, gravadores de videocassete e unidades de aquecimento central. Portanto, nada mais natural que interligar chips de controle para que os aparelhos existentes na casa troquem informações entre si ou informem um sistema de controle central. É perfeitamente possível projetar e montar sistemas de controle central para regular o funcionamento de aparelhos eletrodomésticos. Esses sistemas dividem-se em três categorias: sistemas dedicados, sistemas de interrupção e controladores de rede.

Os sistemas dedicados encontram-se no comércio, mas você pode projetar seus próprios dispositivos, que ficam acoplados a um microcomputador convencional. Através de interfaces específicas, esses dispositivos ligam-se diretamente a unidades elétricas ou eletromecânicas, como luzes e termostatos, controlando seu funcionamento. No entanto, para montar um sistema desse tipo você precisa ter amplo conhecimento do computador e de hardware elétrico e ser capaz de escrever seus próprios programas de controle. Os sistemas dedicados apresentam uma limitação importante: o programa deve funcionar de modo contínuo. Qualquer interrupção de energia elétrica deixa o dispositivo, na melhor das hipóteses, travado em posição constante e incapaz de executar os ajustes e processos do programa.

A segunda categoria de sistemas de controle usa "interruptores", ou seja, sinais eletrônicos gerados pelos dispositivos reguladores, ligados ao sistema de aquecimento central, alarme contra roubo ou detectores de fogo e fumaça. Quando algum desses dispositivos tem algo fora do comum para transmitir ao computador, envia um sinal de prioridade que interrompe o programa em andamento. Embora o sistema de interrupção precise funcionar continuamente, não é tão afetado por uma pane no sistema de energia elétrica, porque os dispositivos que ele controla já são, em parte, auto-regulados.

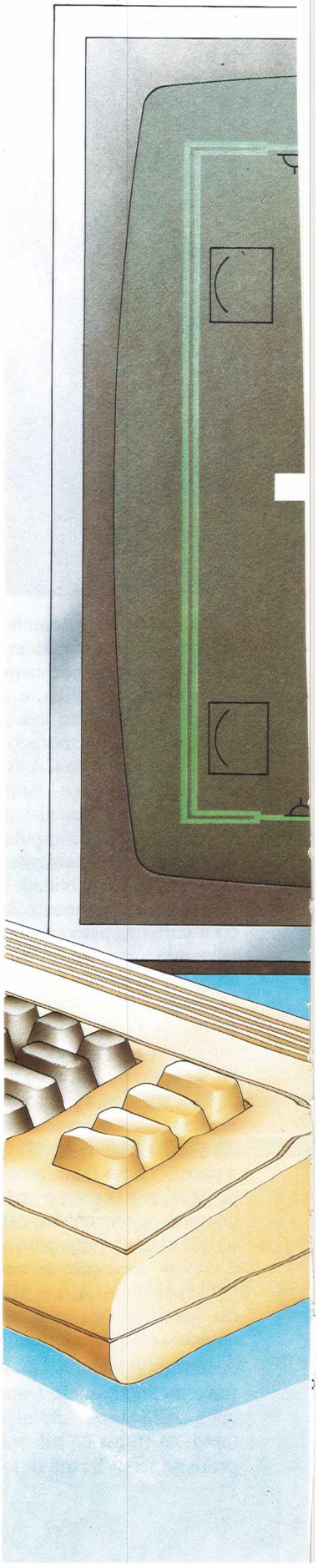
O computador mantém vários programas na memória: um para cada dispositivo ligado a ele, além do programa que o usuário possa estar usando. Digamos que você esteja entretido com um videogame quando o detector de fumaça dispara uma mensagem de interrupção. Em resposta a esse sinal, o computador pára o jogo (preservando toda a informação referente a ele) e começa a executar o programa do detector de fumaça. Pode aparecer uma mensagem na tela, alertando para a possibilidade de um incêndio; ou, se o computador não estiver sendo usado, pode soar um alarme. Uma vez descoberta e controlada a origem da fumaça, você volta ao ponto exato em que o jogo foi interrompido. Contudo, se o sinal viesse do dispositivo de tempo do aquecedor central, o computador verificaria a hora e os sensores externos e internos e conseqüentemente ligaria o aquecedor. Tudo isso é feito tão rápido que nem se percebe que o jogo foi interrompido!

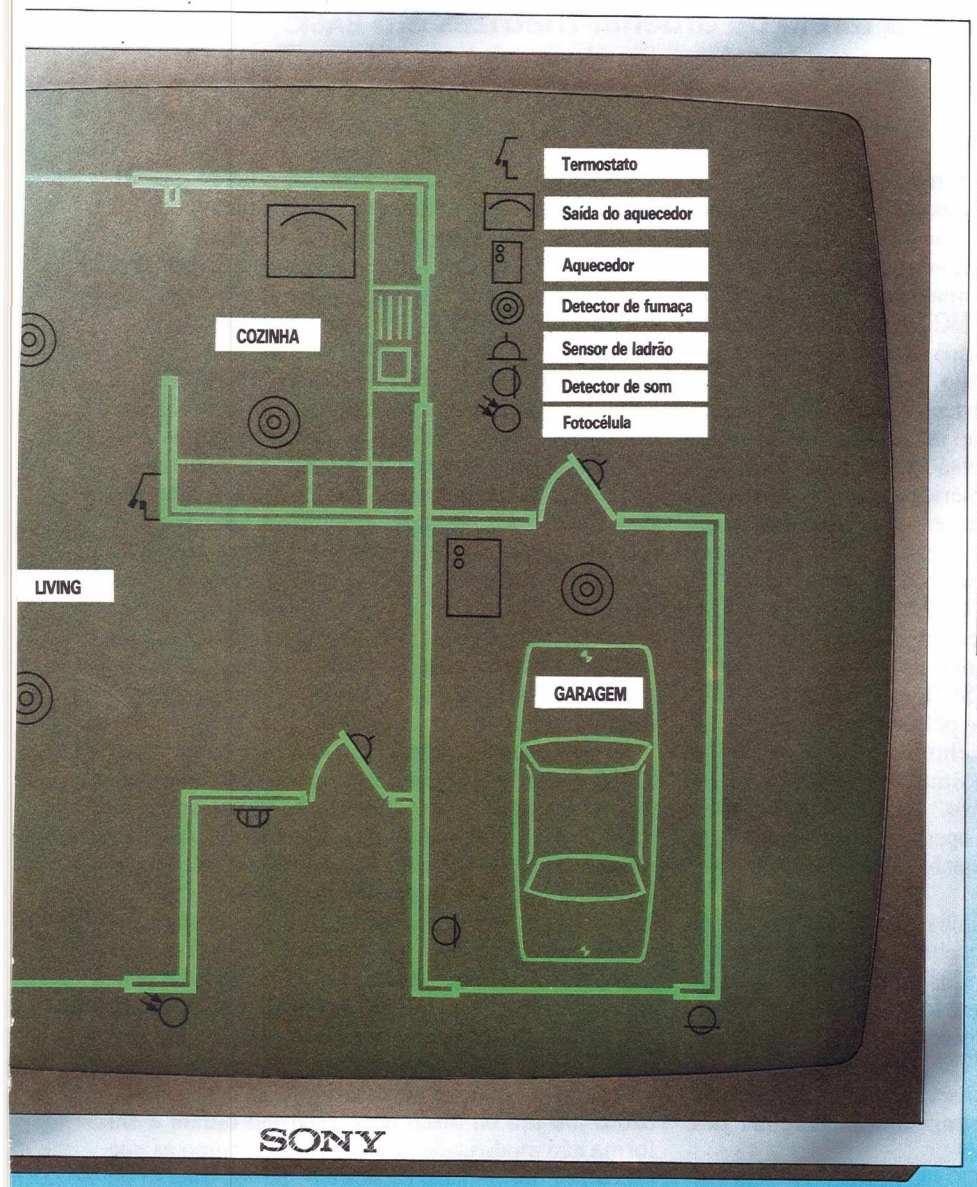
O sistema chamado BSR Home Controller exemplifica a terceira categoria (controladores de rede). Esse engenhoso dispositivo usa a rede elétrica de uma casa para controlar as unidades elétricas ligadas em qualquer tomada ou em determinado circuito. Cada controle recebe um número de código (um endereço) que possibilita que ele seja ligado ou desligado por um sinal de alta freqüência enviado através da rede elétrica. Contudo, a instalação de interfaces desse tipo é extremamente perigosa. Só um eletricista qualificado pode ligar os outputs de um computador à rede elétrica.

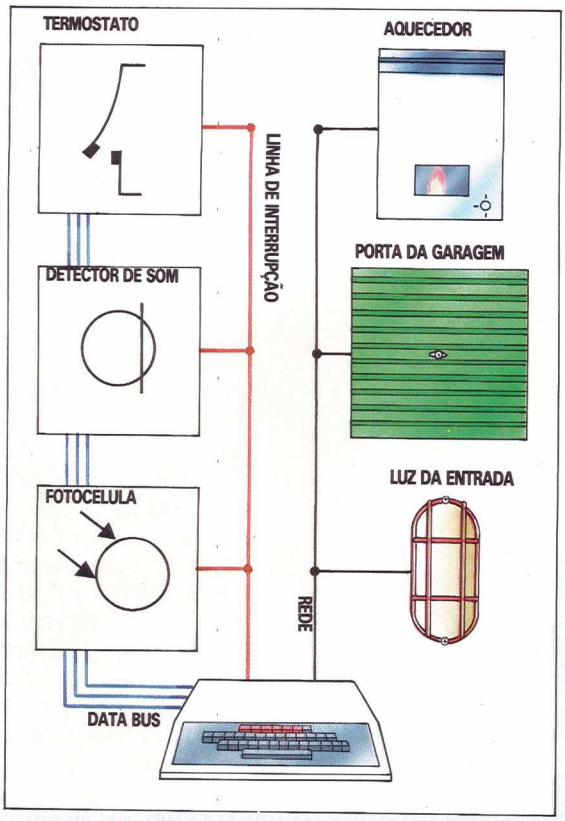
Instalado o sistema de controle, o próximo passo consiste em providenciar alguma forma de operação remota — de modo que certas situações deixem de ser problema —, como, por exemplo, estando a pessoa longe de casa, lembrar-se de que o ar condicionado ficou ligado ou o sistema de alarme desligado. Qualquer dispositivo comum de comunicação, como um modem, serve em todos esses sistemas, permitindo que o controle seja exercido num terminal remoto. Caso essa forma de serviço seja utilizada, torna-se necessário o uso de uma senha como meio de acesso.

Todos os sistemas aqui descritos existem no mercado internacional. Os sensores podem variar, desde microinterruptores do tipo usado em alarmes contra roubo até os mais complexos chips para termômetros digitais. Muitos computadores pessoais como o Apple II, o Commodore 64 e o BBC Model B, têm condições de agüentar esse nível de expansão, mas outras máquinas precisam passar por modificações consideráveis para atingir resultados semelhantes.

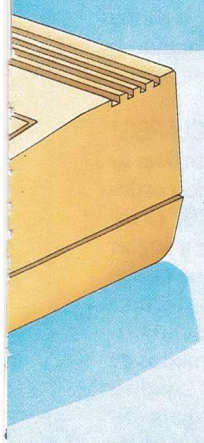
A despesa maior na instalação de um controle por computador para casa consiste na compra de hardware para ligar o computador à rede elétrica — são necessários diversos isoladores, relés e interruptores para fazer o trabalho com segurança e eficiência. É provável, porém, que a tarefa mais exaustiva para o usuário do microcomputador na instalação de um sistema desse tipo seja a de escrever o software. Como tais sistemas dependem da velocidade de resposta (não adianta soar o alarme contra fogo depois do incêndio da casa), os programas de controle devem ser escritos em código de máquina. Ainda não existem programas desse tipo no mercado, mas poderão ser disponíveis no futuro.






**Endereço residencial**
Este desenho mostra duas das técnicas pelas quais um microcomputador pode dirigir diversos aparelhos eletrodomésticos. Quando um dos três sensores à esquerda tem algo a informar, envia um impulso eletrônico pela linha de interrupção comum. Essa linha vai diretamente ao microprocessador, que suspende por algum tempo qualquer programa que esteja executando e pula para uma rotina especial que lê qualquer informação colocada pelo sensor no data bus.
Os dispositivos à direita estão ligados em rede; assim, o computador pode ativar qualquer um deles, enviando um conjunto de dados que contém, por exemplo, o número de código da porta da garagem e as instruções para abri-la.



# Centro de controle

A representação esquematizada de uma casa na tela do computador não é algo impossível. Sistemas de controle de instalações industriais e de segurança computadorizada empregam esse método. Operando o computador pelo método de "interrupção", não é necessária a exibição na tela, porque o software executa todos os controles por trás do programa principal, e não se nota a paralisação num videogame, por exemplo. Pode não estar longe o dia em que as casas serão construídas com redes internas de computação, da mesma forma como hoje são feitas suas instalações elétricas.

# Tempo e movimento

## Embora demorada, a tarefa de ordenar matrizes em BASIC permite ganhar tempo na localização de registros específicos.

Até aqui desenvolvemos a maioria dos códigos necessários para criar entradas no "banco de dados" da agenda de endereços. Não examinamos, porém, a programação para gravar essas entradas em fita ou disco. E falta-nos ainda uma rotina adequada para a criação do campo CAMPMOD$, conforme especificação anterior. O programa completo para tanto é apresentado neste fascículo.

Primeiramente, convertemos os caracteres nas linhas de 10250 a 10330 em maiúsculas. As linhas de 10350 a 10370 efetuam então a contagem dos caracteres da variável alfanumérica e examinam se cada um deles corresponde a um espaço. O último espaço encontrado atribui à variável S o valor correspondente a sua posição na variável alfanumérica.

As linhas de 10400 a 10420 transferem um a um os caracteres alfanuméricos maiúsculos para a variável CNOM$, até atingirmos o último espaço. Isso, se eles tiverem o valor ASCII superior a 64. Todo caractere que não passar pelo teste é ignorado; assim, esse procedimento elimina os sinais de ponto final (ASCII 46), de apóstrofo (ASCII 39), de espaço (ASCII 32) e outros. As linhas de 10450 a 10470 efetuam a mesma operação com os caracteres após o espaço final, transferindo-os para SNOM$ (sobrenome).

Se N$ contiver uma única palavra — por exemplo, TREVANIAN —, S receberá o valor 0 e todos os caracteres serão transferidos para SNOM$. A variável empregada no prenome foi denominada CNOM$, em vez de FNOM$. Isso porque as variáveis que começam com as letras "FN" causam confusão em muitos dos dialetos do BASIC, levando-os a identificar uma chamada com uma função definida pelo usuário (FN).

As linhas 10490 e 10500 são necessárias para atribuição de valores nulos a variáveis alfanuméricas nessa rotina, antes de serem usadas de novo. Esse é um aspecto a ser considerado sempre que as estruturas do tipo LET X$=X$+Y$ forem empregadas. Deixando de "limpar" as variáveis, isso resultará no acúmulo progressivo de caracteres indesejados em cada nova utilização. Observe que ESCL recebe o valor 0 na rotina ACRREG, porque só queremos garantir que o usuário acrescente um registro, se não houver nenhum no arquivo (na primeira vez em que o programa é processado).

Agora que temos um meio de incluir quantos registros quisermos no arquivo, necessitamos de um método para gravá-lo em fita ou disco. O sistema mais simples seria escrever todos os registros no arquivo de dados (AGEN.DAD, na versão deste programa), seguindo a ordem em que acaso se encontrem. O principal inconveniente desse tipo de procedimento surge quando temos de localizar um registro específico. Se não pudermos ter certeza de que todos estão ordenados de alguma forma, o único método viável será voltar ao começo, examinando cada registro por vez, para verificar se combina com a "chave" da busca. Se, por acaso, o registro procurado for o último fornecido ao banco de dados, todos os demais terão de ser checados antes. Se o último registro fornecido for Ricardo Bergamo (isto é, CAMPMOD$(TAMA−1) = "BERGAMO RICARDO"), a rotina de busca deve prever o registro como estando em algum ponto próximo ao início do arquivo — caso os registros estejam ordenados. Mas tanto a ordenação como a busca são procedimentos demorados. Precisa-se, portanto, determinar qual dos dois tem prioridade. Adotamos o princípio de que uma agenda de endereços é consultada com muito mais freqüência do que recebe novos dados (ou alguma forma de modificação). Assim, para perder menos tempo, preferimos ordenar os registros antes de armazená-los no arquivo de dados, após a utilização do programa.

Tendo isso em mente, uma variável denominada RMOD é criada para ser usada como flag. Ela pode ter um dos dois valores: 0 ou 1. De início, recebe o valor 0, que indica a não modificação de qualquer registro durante a presente execução do programa. Qualquer operação que altere o arquivo — como o acréscimo de novos registros — atribui o valor 1 a RMOD. As operações que "precisam saber" se o arquivo foi modificado examinarão o valor de RMOD antes de prosseguir. Por exemplo, SAIPRG, a rotina que grava o arquivo e desvia o programa, verifica RMOD na linha 11050. Se RMOD = 0, a gravação e o ordenamento não são necessários, pois o arquivo de dados em fita ou disco deve estar em ordem e em forma não modificada. Outras rotinas, como as que procuram um registro específico, também terão de checar RMOD. Se RMOD for 0, a busca (ou outra operação) pode prosseguir. Se RMOD for 1, a rotina de ordenação terá de ser chamada antes. Quando o arquivo inteiro estiver ordenado, atribui-se novamente o valor 0 a RMOD.

Nossa rotina de ordenação, chamada *CLAREG* na listagem do programa, recoloca RMOD em 0 na linha 11320, após todos os registros terem sido ordenados. Antes de passar ao exame de *SAIPRG* (a rotina que grava o arquivo em fita ou disco e a seguir encerra o programa), convém discorrer um pouco sobre a *CLAREG*. Trata-se de uma forma de técnica de ordenação denominada "bubble sort" (classificação bolha) (ver p. 286). Entre os muitos modos de ordenar dados, este é um dos mais simples e rápidos. Há bons motivos para a adoção de rotinas mais eficientes, mas ordenações muito elaboradas são de compreensão difícil. Portanto, só examinando a quantidade de itens a ser ordenada se pode tomar uma decisão a respeito. A "complexidade de

tempo'' da nossa bubble sort é $n^2$. Ou seja, o tempo exigido para a ordenação de dados aumenta na proporção do quadrado do número de itens que devem ser ordenados. Se dois itens levarem 4 milissegundos para ordenação, quatro levarão 16 milissegundos, cinqüenta precisarão de 2 segundos e meio, e mil vão requerer mais de 16 minutos. Uma demora de 2 ou 3 segundos é admissível no uso de programas como o nosso, mas não uma de 15 minutos.

A maneira pela qual desenvolvemos este programa possibilita um máximo de cinqüenta registros, o que não implica perda de tempo durante as ordenações. Contudo, mais adiante no curso, esboçaremos algumas técnicas empregadas para criar arquivos dinâmicos, que podem ser ampliados para praticamente qualquer tamanho. Se você tentar uma modificação desse tipo no programa, um dos primeiros problemas a enfrentar será o das rotinas mais elaboradas de ordenação.

Os itens de dados que estamos ordenando são as séries de caracteres em CAMPMOD$(L) e CAMPMOD$(L+1). Os registros só ocorrem se CAMPMOD$(L) for maior que CAMPMOD$(L+1), e o campo do índice (não usado no momento) é atualizado nas linhas 11490 e 11570. Toda vez que dois registros forem permutados, a variável S (para indicar que uma permuta ocorreu) recebe o valor 1. Ao atingir a linha 11290, a rotina de busca checa o valor de S e retrocede para comparar todos os registros outra vez. Quando eles estiverem ordenados, o valor de S será 0 e a rotina vai se encerrar após a recolocação do valor de RMOD em 0.

A rotina SAIPRG (referida como *SAIPRG* na listagem do programa) começa na linha 11000. Ela primeiro examina os registros para verificar se algum deles foi modificado durante a presente execução do programa (linha 11050 IF RMOD = 0 THEN RETURN). Se não houve alteração, não é preciso regravar o arquivo. Assim, a rotina retorna ao programa principal. Isso nos traz de volta à linha 100, que checa o valor de ESCL. Se ESCL tem o valor 9 (como teria se *SAIPRG* fosse executada), o programa principal segue para a instrução END na linha 110.

No caso de o programa verificar que RMOD vale 1, na linha 11050, isso quer dizer que pelo menos um registro foi modificado, e há a possibilidade de os registros não mais estarem em ordem. Sendo assim, a rotina *SAIPRG* chama a sub-rotina de ordenação (linha 11070) e a seguir, após todos os registros terem sido ordenados, grava-os em fita ou disco.

A rotina para gravar (*GRAREG*) é chamada na linha 11090 e começa na linha 12000. Na listagem principal, ela se desenvolve em BASIC Microsoft e, portanto, os detalhes para manipulação do arquivo variam de uma versão para outra dessa linguagem. A linha 12030 abre o arquivo de dados AGEN.DAD e atribui o número de canal #1 à operação. A linha 12050 determina os limites do loop que conta todos os registros do arquivo. O limite superior é TAMA−1, e não TAMA, porque esta última variável sempre tem um valor que é uma unidade maior do que o número de registros válidos no arquivo. Assim, ao ser acrescentado um novo registro, este não se sobrepõe a um já existente.

O formato das linhas 12060 e 12070 merece atenção. Cada campo é separado por uma ",", também enviada ao arquivo. Essa vírgula é necessária à maioria das versões do BASIC porque INPUT # e PRINT # operam do mesmo modo que as instruções normais INPUT e PRINT. Observe a instrução INPUT X,Y,Z. Ela aguardaria uma entrada do teclado como 10,12,15<CR>, que atribuiria 10, 12 e 15 a X, Y e Z, respectivamente. Sem as vírgulas, a instrução INPUT não poderia identificar onde cada item de dados termina e atribuiria todos os dados à primeira variável. De modo semelhante, a instrução INPUT# (na maioria das versões do BASIC) não poderia dizer onde cada registro do arquivo termina e tentaria preencher cada variável alfanumérica com tantos dados quantos coubessem. Uma vez que, na maioria das versões do BASIC, as variáveis alfanuméricas comportam até 255 caracteres, os dados seriam atribuídos muito antes que o loop FOR L = 1 TO TAMA−1 se encerrasse. Isso resultaria numa mensagem de erro INPUT PAST END (indicando que uma instrução INPUT foi emitida após ter sido atingido o limite da capacidade de dados), e as variáveis alfanuméricas (como NAMFLD$(x)) acabariam por conter muito mais dados do que sua capacidade.

Uma vez armazenados todos os registros no arquivo, de L = 1 TO TAMA−1, *GRAREG* retorna à linha 90 no programa principal. A linha 100 examina o valor de ESCL para checar se a última operação foi *SAIPRG* ou não. No caso de ter sido 9 (gravar e desviar), o programa segue para a instrução END na linha 110. Se ESCL tiver qualquer outro valor, o programa retorna a *ESCLHA* e possibilita outra opção ao usuário.

Como observação final, mencionamos a rotina *TAMARQ* que começa na linha 12500. Ela é oferecida como possível alternativa à instrução da linha 1520. Do modo como está apresentado, o programa depende da presença de uma função de encerramento do arquivo: IF EOF(1) = −1 THEN LET L = 50. Todos os BASIC têm alguma forma de indicar que o fim de um arquivo foi atingido, por meio de função especial, como EOF(x), ou pelo acesso a uma posição especial da memória, mediante o comando PEEK. A rotina *TAMARQ*, na linha 12500, é apresentada como sugestão, caso não exista a função EOF no equipamento e a linha 1520 tenha de ser substituída por GOSUB 12500.

## A propósito...

| Comando/Instrução/Função | Ação/Resultado | MS BASIC | Applesoft | TRS-80 | Sinclair |
|---|---|---|---|---|---|
| INT (x) | Fornece o maior inteiro que é menor ou igual a x | + | + | + | + |
| INVERSE | Passa o monitor para a forma de vídeo inverso | | + | | |
| KILL "arquivo" | Elimina um arquivo do disquete | + | | | |

| Comando/Instrução/Função | Ação/Resultado | MS BASIC | Applesoft | TRS-80 | Sinclair |
|---|---|---|---|---|---|
| LEFT$(x$,n) | Fornece o enésimo caractere mais à esquerda de x$ | + | + | + | |
| LEN (x$) | Fornece o número de caracteres de x$ | + | + | + | + |
| LET variável=expressão | Aloca o valor de expressão à variável | + | + | + | + |
| LINE INPUT"mensagem";var$ | Lê uma linha inteira a partir do teclado e aloca a var$ | + | | | |
| LINE INPUT#arquivo,var$ | Lê uma linha inteira de um arquivo e aloca a var$ | + | | | |
| LINE (x1,y1)-(x2,y2) | Desenha uma linha na tela entre os dois pontos | + | | | |
| LIST linha | Lista na tela a partir da linha especificada | | | | + |
| LIST linha1-linha2 | Lista as linhas especificadas no vídeo | + | + | + | |
| LLIST linha | Lista o programa na impressora a partir da linha | | | | + |
| LLIST linha1-linha2 | Lista as linhas especificadas na impressora | + | | + | |
| LN x | Fornece o logaritmo natural de x | | | | + |
| LOAD "arquivo" | Carrega um arquivo de programa | + | + | | + |
| LOC (a) | Fornece a posição atual no arquivo | + | | | |
| LOCATE linha,coluna | Posiciona o cursor no ponto indicado da tela | + | | | |
| LOCK "nomearquivo" | Protege um arquivo em disco | | + | | |
| LOG(x) | Fornece o logaritmo natural de x | + | + | + | |
| LOMEM | Gera limite inferior da memória para programa BASIC | | + | | |
| LPOS (n) | Fornece a posição atual da cabeça da impressora no buffer | + | | | |
| LPRINT lista de expressões | Lista um conjunto de variáveis na impressora | + | | + | + |
| MEM | Fornece o total de bytes disponíveis na memória | | | + | |
| MERGE "arquivo" | Intercala um programa gravado com o da memória | + | | | |
| MID$(v$,n,m)=y$ | Aloca na vary$ parte da varyv$, iniciando em n m carac | + | | + | |
| MID$(x$,n,m) | Fornece m caracteres da variável x$ a partir da posição n | + | + | | |
| NAME "arquivo" AS "nome" | Muda o nome do arquivo | + | | | |
| NEW | Elimina o programa atual e variáveis | + | + | + | + |
| NEXT variável | Encerra um loop FOR-NEXT | + | + | + | + |
| NO TRACE | Desliga o comando TRACE | | + | | |
| NORMAL | Desliga os modos FLASH e INVERSE de vídeo | | + | | |
| ON ERROR GOTO linha | Em caso de erro, pula para a linha indicada | + | | + | |
| ON expressão GOSUB linhas | Passa a execução para a linha correspondente a INT(expr) | + | + | + | |
| ON expressão GOTO linhas | Passa a execução para a linha correspondente a INT(expr) | + | + | + | |
| ONNER GOTO linha | Em caso de erro, pula para a linha indicada | | + | | |
| OPEN "nomearquivo" | Abre um arquivo de texto | | + | | |

```
10 REM "PRGPRN"
20 REM *INICIL*
30 GOSUB 1000
40 REM *SAUDAR*
50 GOSUB 3000
60 REM *ESCLHA*
70 GOSUB 3500
80 REM *EXECUT*
90 GOSUB 4000
100 IF ESCL <> 9 THEN 60
110 END
1000 REM SUB-ROTINA *INICIL*
1010 GOSUB 1100: REM SUB-ROTINA *CRIMAT* (CRIA MATRIZES)
1020 GOSUB 1400: REM SUB-ROTINA *LERARQ* (LE ARQUIVOS)
1030 GOSUB 1600: REM SUB-ROTINA *ESTFLG* (ESTABELECE FLAGS)
1040 REM
1050 REM
1060 REM
1070 REM
1080 REM
1090 RETURN
1100 REM SUB-ROTINA *CRIMAT* (CRIA MATRIZES)
1110 DIM CAMPNOM$(50)
1120 DIM CAMPMOD$(50)
1130 DIM CAMPRUA$(50)
1140 DIM CAMPCID$(50)
1150 DIM CAMPEST$(50)
1160 DIM CAMPTEL$(50)
1170 DIM CAMPIND$(50)
1180 REM
1190 REM
1200 REM
1210 LET TAMA=0
1220 LET RMOD=0
1230 LET GRAV=0
1240 LET CORR=0
1250 REM
1260 REM
1270 REM
1280 REM
1290 REM
1300 RETURN
1400 REM SUB-ROTINA *LERARQ*
1410 OPEN "I", #1, "AGEN.DAD"
1420 INPUT #1, TEST$
1430 IF TEST$="@ PRIMEIRO" THEN GOTO 1540: REM CLOSE E
     RETURN
1440 LET CAMPNOM$(1)=TEST$
```

```
1450 INPUT #1,
     CAMPMOD$(1) ,CAMPRUA$(1) ,CAMPCID$(1) ,CAMPEST$(1) ,
     CAMPTEL$(1)
1460 INPUT #1, CAMPIND$(1)
1470 LET TAMA=2
1480 FOR L=2 TO 50
1490 INPUT #1,
     CAMPNOM$(L) ,CAMPMOD$(L) ,CAMPRUA$(L) ,CAMPCID$(L) ,
     CAMPEST$(L)
1500 INPUT #1, CAMPTEL$(L) ,CAMPIND$(L)
1510 LET TAMA=TAMA+1
1520 IF EOF(1)=-1 THEN LET L=50
1530 NEXT L
1540 CLOSE #1
1550 RETURN
1600 REM SUB-ROTINA *ESTFLG*
1610 REM ESTABELECE FLAGS APOS *LERARQ*
1620 REM
1630 REM
1640 IF TEST$="@PRIMEIRO" THEN LET TAMA=1
1650 REM
1660 REM
1670 REM
1680 REM
1690 RETURN
3000 REM SUB-ROTINA *SAUDAR*
3010 PRINT CHR$(12): REM LIMPA TELA
3020 PRINT
3030 PRINT
3040 PRINT
3050 PRINT
3060 PRINT TAB(14); "*BENVINDO AO*"
3070 PRINT TAB(4); "*MICROCOMPUTADOR -- CURSO BASICO*"
3080 PRINT TAB(1); "*AGENDA DE ENDERECOS COMPUTADORIZADA*"
3090 PRINT
3100 PRINT "(PRESSIONAR A BARRA DE ESPACO PARA CONTINUAR)"
3110 FOR L=1 TO 1
3120 IF INKEY$ <> " " THEN L=0
3130 NEXT L
3140 PRINT CHR$(12)
3150 RETURN
3500 REM SUB-ROTINA *ESCLHA*
3510 REM
3520 IF TEST$="@PRIMEIRO" THEN GOSUB 3860: REM SUB-ROTINA
     *PRIMEI*
3530 IF TEST$="@PRIMEIRO" THEN RETURN
3540 REM "MENESC"
3550 PRINT CHR$(12)
3560 PRINT "ESCOLHER UM DOS SEGUINTES:"
3570 PRINT
3580 PRINT
3590 PRINT
3600 PRINT "1. ENCONTRAR UM REGISTRO (PELO NOME)"
3610 PRINT "2. ENCONTRAR NOMES (POR TRECHO DE UM NOME)"
3620 PRINT "3. ENCONTRAR REGISTROS (DE UMA CIDADE)"
3630 PRINT "4. ENCONTRAR REGISTROS (DE UMA INICIAL)"
3640 PRINT "5. LISTAR TODOS OS REGISTROS"
3650 PRINT "6. ACRESCENTAR UM REGISTRO"
3660 PRINT "7. MODIFICAR UM REGISTRO"
3670 PRINT "8. ELIMINAR UM REGISTRO"
3680 PRINT "9. SAIR E GRAVAR"
3690 PRINT
3700 PRINT
3710 REM "ATRESC"
3720 REM
3730 LET L=0
3740 LET I=0
3750 FOR L=1 TO 1
3760 PRINT "ESCOLHER DE 1 A 9"
3770 FOR I=1 TO 1
3780 LET A$=INKEY$
3790 IF A$="" THEN I=0
3800 NEXT I
3810 LET ESCL=VAL(A$)
3820 IF ESCL <1 THEN L=0
3830 IF ESCL >9 THEN L=0
3840 NEXT L
3850 RETURN
3860 REM SUB-ROTINA *PRIMEI* (DA MENSAGEM)
3870 LET ESCL=6
3880 PRINT CHR$(12): REM LIMPA TELA
3890 PRINT
3900 PRINT TAB(8);"NAO EXISTEM REGISTROS NO"
3910 PRINT TAB(8);"ARQUIVO. VOCE DEVERA COMECAR"
3920 PRINT TAB(8);"ACRESCENTANDO UM REGISTRO"
3930 PRINT
3940 PRINT "(PRESSIONAR A BARRA DE ESPACO PARA CONTINUAR)"
3950 FOR B=1 TO 1
3960 IF INKEY$ <> "" THEN B=0
3970 NEXT B
3980 PRINT CHR$(12): REM LIMPA TELA
3990 RETURN
4000 REM SUB-ROTINA *EXECUT*
4010 REM
4020 IF ESCL=6 THEN GOSUB 10000
4030 REM
4040 REM 1= *ENCREG*
4050 REM 2 = *ENCNOM*
4060 REM 3 = *ENCCID*
4070 REM 4 = *ENCINI*
4080 REM 5 = *LISREG*
4090 IF ESCL=6 THEN GOSUB 10000
4100 REM 7 = *MODREG*
4110 REM 8 =6 *ELMREG*
4120 IF ESCL=9 THEN GOSUB 11000
4130 REM
4140 RETURN
10000 REM SUB-ROTINA *ACRREG*
10010 PRINT CHR$(12): REM LIMPA TELA
10020 INPUT "FORNECER O NOME";CAMPNOM$(TAMA)
10030 INPUT "FORNECER A RUA";CAMPRUA$(TAMA)
10040 INPUT "FORNECER A CIDADE";CAMPCID$(TAMA)
10050 INPUT "FORNECER O ESTADO";CAMPEST$(TAMA)
10060 INPUT "FORNECER O NUMERO TELEFONICO";
      CAMPTEL$(TAMA)
10070 LET RMOD=1: REM FLAG DE REGISTRO MODIFICADO
10080 LET CAMPIND$(TAMA)=STR$(TAMA)
10090 LET TEST$=""
10100 GOSUB 10200: REM *MODNOM*
10110 LET ESCL=0
10120 LET TAMA=TAMA+1
10130 REM
10140 REM
10150 RETURN
10200 REM ROTINA *MODNOM*
10210 REM CONVERTE O CONTEUDO DE CAMPNOM$ EM MAIUSCULAS,
10220 REM REMOVE SINAIS E ARQUIVA NA ORDEM:
10230 REM SOBRENOME+ESPACO+PRIMEIRONOME EM CAMPMOD$
10240 REM
10250 LET N$=CAMPNOM$(TAMA)
10260 FOR L=1 TO LEN(N$)
10270 LET TEMP$=MID$(N$,L,1)
10280 LET T=ASC(TEMP$)
10290 IF T >|= 97 THEN T=T-32
10300 LET TEMP$=CHR$(T)
10310 LET P$=P$+TEMP$
10320 NEXT L
10330 LET N$=P$
10340 REM ENCONTRAR ULTIMO ESPACO
10350 FOR L=1 TO LEN(N$)
10360 IF MID$(N$,L,1)=" " THEN S=L
10370 NEXT L
10380 REM ELIMINAR SINAIS E ARQUIVAR PRIMEIRONOME
10390 REM EM CNOM$
10400 FOR L=1 TO S-1
10410 IF ASC(MID$(N$,L,1)) > 64 THEN
      CNOM$=CNOM$+MID$(N$,L,1)
10420 NEXT L
10430 REM ELIMINA SINAIS E ARQUIVA SOBRENOME
10440 REM EM SNOM$
10450 FOR L=S+1 TO LEN(N$)
10460 IF ASC(MID$(N$,L,1)) > 64 THEN
      SNOM$=SNOM$+MID$(N$,L,1)
10470 NEXT L
10480 LET CAMPMOD$(TAMA)=SNOM$+" "+CNOM$
10490 LET P$="": LET N$="": LET SNOM$="": LET CNOM$="":
      LET S=0
10500 RETURN
11000 REM SUB-ROTINA *SAIPRG*
11010 REM CLASSIFICA E GRAVA O ARQUIVO
11020 REM SE ALGUM REGISTRO FOI
11030 REM MODIFICADO(RMOD=1)
11040 REM
11050 IF RMOD=0 THEN RETURN
11060 REM
11070 GOSUB 11200: REM *CLAREG*
11080 REM
11090 GOSUB 12000: REM *GRAREG*
11100 RETURN
11200 REM SUB-ROTINA *CLAREG*
11210 REM CLASSIFICA TODOS OS REGISTROS POR ORDEM
11220 REM ALFABETICA SEGUNDO CAMPMOD$ E ATUALIZA
11230 REM CAMPIND$
11240 REM
11250 LET S=0
11260 FOR L=1 TO TAMA-2
11270 IF CAMPMOD$(L) > CAMPMOD$(L+1) THEN GOSUB 11350
11280 NEXT L
11290 IF S=1 THEN 11250
11300 REM
11310 REM
11320 LET RMOD=0: REM ZERA A FLAG DE REGISTRO MODIFICADO
11330 REM
11340 RETURN
11350 REM SUB-ROTINA *INVREG*
11360 LET TCAMNOM$=CAMPNOM$(L)
11370 LET TCAMMOD$=CAMPMOD$(L)
11380 LET TCAMRUA$=CAMPRUA$(L)
11390 LET TCAMCID$=CAMPCID$(L)
11400 LET TCAMEST$=CAMPEST$(L)
11410 LET TCAMTEL$=CAMPTEL$(L)
11420 REM
11430 LET CAMPNOM$(L)=CAMPNOM$(L+1)
11440 LET CAMPMOD$(L)=CAMPMOD$(L+1)
11450 LET CAMPRUA$(L)=CAMPRUA$(L+1)
11460 LET CAMPCID$(L)=CAMPCID$(L+1)
11470 LET CAMPEST$(L)=CAMPEST$(L+1)
11480 LET CAMPTEL$(L)=CAMPTEL$(L+1)
11490 LET CAMPIND$(L)=STR$(L)
11500 REM
11510 LET CAMPNOM$(L+1)=TCAMNOM$
11520 LET CAMPMOD$(L+1)=TCAMMOD$
11530 LET CAMPRUA$(L+1)=TCAMRUA$
11540 LET CAMPCID$(L+1)=TCAMCID$
11550 LET CAMPEST$(L+1)=TCAMEST$
11560 LET CAMPTEL$(L+1)=TCAMTEL$
11570 LET CAMPIND$(L+1)=STR$(L+1)
11580 LET S=1
11590 REM
11600 RETURN
12000 REM SUB-ROTINA *GRAREG*
12010 REM
12020 REM
12030 OPEN "O", #1, "AGEN.DAD"
12040 REM
12050 FOR L=1 TO TAMA-1
12060 PRINT #1,CAMPNOM$(L);",";
      CAMPMOD$(L);",";CAMPRUA$(L);","CAMPCID$(L)
12070 PRINT #1,CAMPEST$(L);",";CAMPTEL$(L);","CAMPIND$(L)
12080 NEXT L
12090 REM
12100 REM
12110 REM
12120 REM
12130 CLOSE #1
12140 REM
12150 RETURN
12500 REM SUB-ROTINA *TAMARQ*
12510 IF CAMPNOM$(L)="  " THEN LET L=50
12520 IF CAMPNOM$(L)="  " THEN RETURN
12530 LET TAMA=TAMA+1
12540 REM
12550 REM
12560 RETURN
```

# Vannevar Bush



**O analisador diferencial**
Esta máquina foi projetada para a solução de importante classe de funções matemáticas: as equações diferenciais de segundo grau. O método fora sugerido por Lord Kelvin, e consistia em introduzir o output de um "integrador" (instrumento que calcula com eficiência a área dentro de uma curva) no input de outro. No entanto, a força do output era geralmente muito fraca para atuar como um input e o método só pôde ser aplicado depois do invento do amplificador.

A máquina que Bush criou em 1931 era uma complexa estrutura de engrenagens, eixos e motores elétricos. O input e o output viam-se expressos em forma de rotações do eixo. Resolveu-se o problema do feedback pelo emprego de um amplificador de "toque".

Na década de 40, construiu-se um analisador diferencial mais avançado, usando componentes elétricos, mas a máquina pesava mais de 100 toneladas. O output das cinco registradoras era em forma de impresso digital.

**O analisador diferencial, projetado por Vannevar Bush, era uma calculadora eletromecânica que resolvia equações diferenciais.**

Muitos afirmam que o americano Vannevar Bush é o pai do computador. Sua mais importante contribuição para o desenvolvimento da informática remonta a 1931, quando criou um analisador diferencial mecânico, fator de estímulo das pesquisas que levaram ao desenvolvimento do computador digital.

Bush nasceu perto de Boston, Massachusetts, em 11 de março de 1890 e, seguindo os passos do pai, fez o curso de engenharia. Após graduar-se, em 1913, trabalhou algum tempo na General Electric e em seguida foi professor-assistente na faculdade onde se formara. Fez estudos de pós-graduação na Universidade de Harvard e no Instituto de Tecnologia de Massachusetts (MIT). Durante a Primeira Guerra Mundial (1914-18), Bush trabalhou na fabricação de detectores de submarinos utilizados pela Marinha dos Estados Unidos.

Quando estudante, fez sua primeira invenção: um dispositivo para levantamento topográfico de terras. O mecanismo, suspenso entre as rodas de uma bicicleta, calculava a altura do solo em que passava e emitia o resultado na forma de gráfico representativo do perfil da área. Como parte do aparelho, havia um dispositivo chamado integrador, uma vez que a determinação da altura em qualquer posição exigia o conhecimento de todos os valores anteriores do percurso.

No MIT, Bush ensinou transmissão de energia elétrica e passou a pesquisar um dos maiores problemas relacionados com o fornecimento de eletricidade: como evitar *black-outs* resultantes de aumentos repentinos e imprevistos da demanda. As equações matemáticas envolvidas em tal situação tinham sido descobertas no século XIX pelo cientista escocês James Clerk Maxwell (1831-1879). Contudo, havia tantas equações simultâneas que o problema não podia ser facilmente resolvido com lápis e papel, e Bush tratou de inventar uma máquina para efetuar os cálculos. Ele se inspirou no trabalho de Lord Kelvin (1824-1907), físico inglês que havia proposto uma máquina para resolver equações envolvidas na previsão das marés.

No começo dos anos 20, Bush construiu sua primeira calculadora, que capacitava os operadores a traçar gráficos do percurso das ondas (usando um potenciômetro, instrumento que transforma a medida de uma posição em voltagem). Eles passavam esses sinais elétricos num medidor de watts especialmente adaptado — o disco giratório encontrado em qualquer medidor de força doméstico, que registra a quantidade de energia consumida, integrando os valores oscilantes da corrente e da voltagem para fornecer o consumo.

O sucesso alcançado por essa máquina na solução de um conjunto de equações simultâneas sugeriu a possibilidade de se construir um dispositivo que resolveria equações diferenciais de segundo grau ainda mais difíceis. Uma pesquisa mais aprofundada levou Bush à criação do primeiro analisador diferencial, em 1931. A máquina fez tanto sucesso que foi lançada também na Inglaterra e em outros países da Europa.

Nos Estados Unidos, a Escola Moore de Engenharia Elétrica, da Universidade da Pensilvânia, que mais tarde construiu o computador ENIAC (ver p. 88), utilizou com sucesso um desses analisadores. Onde a calculadora de Bush conseguira reduzir a margem de erro a 2%, o analisador diferencial fornecia resultados com precisão de 99,95%. Mas o custo do aumento da precisão desse dispositivo mecânico elevou-se a dez vezes em cada decimal extra alcançada. Com o desenvolvimento do computador digital, no entanto, o custo de uma máquina apenas dobrava quando sua precisão aumentava da mesma forma.

Bush tornou-se reitor da Escola de Engenharia e vice-presidente do Instituto Carnegie, em 1939, e a competência por ele demonstrada na administração desse patrimônio milionário para pesquisa científica resultou em sua nomeação para presidente do Departamento de Pesquisa da Defesa Nacional americana, no ano seguinte. Nesse cargo ele respondeu pela pesquisa militar dos Estados Unidos durante a Segunda Guerra Mundial (1939-45) e, em particular, influiu na autorização para o projeto Manhattan, que resultou na construção da primeira bomba atômica. Vannevar Bush aposentou-se em 1955 para dedicar-se a seus *hobbies*, entre os quais se incluíam iatismo, criação de perus e invenções. Faleceu quase vinte anos depois, em 1974.



**Mentes jovens**
As crianças são mais receptivas à nova tecnologia do que alguns adultos, que reagem contra a idéia de ter de aprender novos conceitos.

# Coisa de criança?

## Os mais modernos brinquedos educativos têm capacidade de processamento equivalente à de microcomputadores e usam programação similar.

Os microprocessadores, que formam o "coração" dos microcomputadores, tornaram-se componentes quase obrigatórios de muitos aparelhos eletrodomésticos, como máquinas de costura, máquinas de lavar e até fechaduras de portas. Fabricantes de brinquedos também experimentaram a utilização de microprocessadores, sobretudo em carrinhos de corrida e trens elétricos.

Pelo menos uma empresa fabricante de computadores, a Texas Instruments, descobriu que a produção de brinquedos educativos providos de microprocessadores mostrava-se muito compensadora. Seu primeiro lançamento nesse ramo foi a unidade do tipo calculadora que propunha problemas elementares de matemática: o Little Professor, que se tornou muito popular, inclusive no Brasil, onde foi comercializado com o nome de Professor Corujinha.

Speak & Maths foi o segundo brinquedo educativo da Texas Instruments a fazer uso do chip sintetizador de voz. Antes (em 1978) a empresa lançara o Speak & Spell, com vocabulário de algumas centenas de palavras. O teclado da unidade, feito de membrana flexível, assemelha-se ao TK82, e inclui todo o alfabeto além de teclas adicionais. Pressionando-se a tecla GO, o usuário ouve a palavra que ele deverá escrever usando o teclado. Em cada toque, a letra correspondente aparece no visor, por meio de diodos emissores de luz, até que a palavra esteja completa. Então, o Speak & Spell diz ao usuário (em voz alta) se a palavra foi escrita de modo correto ou não. Há, ainda, outros jogos com palavras.

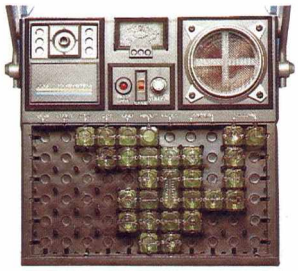
Speak & Maths funciona de maneira semelhante, mas propõe problemas aritméticos. Esses dois produtos altamente inovadores conseguiram ocupar larga faixa do mercado de brinquedos educativos e levaram a Texas Instruments para um campo bem diferente do mercado tradicional da eletrônica.

O terceiro brinquedo falante da Texas, o Touch & Tell, é mais recreativo que educativo. Apresenta uma série de capas plásticas, cada uma com gravura ou desenho diferentes, que são individualmente identificados por um código magnético. Quando a criança toca um ponto da gravura, o aparelho diz, de forma audível, qual o objeto escolhido.

A sintetização de voz constitui a técnica de computação mais sofisticada à disposição dos fabricantes de jogos e brinquedos. Sua aplicação mais popular ocorre nas versões reduzidas de alguns jogos de fliperama.

Outra área onde o microcomputador provocou impacto no mercado de brinquedos é a de carros que andam sozinhos. Um dos mais conhecidos é o Big Trak, programado por meio de um teclado, onde devem ser digitadas as instruções. O brinquedo parece uma tartaruga (ver p. 176) e pode ser controlado por microcomputador, usando-se uma entrada paralela.

Entre outros brinquedos à base de microprocessador, incluem-se: o Simon (lançado no Brasil, pela Estrela, com o nome de Genius), que desafia a criança a repetir uma seqüência de notas musicais e luzes coloridas; o Maximus, da Playskool, que faz um treinamento aritmético semelhante ao do Little Professor; e uma variedade de robôs. Entre os brinquedos para crianças maiores (e adultos) encontram-se o Electroni-Kit, o Mykit Systems e o Radionics. Como os próprios nomes sugerem, são kits eletrônicos. Usam componentes encapsulados e podem ser fixados em placas que servem de base para formar uma variedade de dispositivos simples.



**Electroni-Kit**
Como o próprio nome indica, este aparelho é um kit de montagem usado para criar uma variedade de dispositivos eletrônicos, como radiotransistores, amplificadores etc. Seus componentes, encapsulados em plástico transparente, são montados sobre uma placa que servirá de base para o dispositivo desejado (seguindo-se os diagramas dos esquemas). O kit mais sofisticado da série é o de componentes de um microcomputador rudimentar, próprio para ensinar operação muito simples com o código da máquina.



**Maximus**
É um produto mais avançado, produzido pela MB Electronics. Embora pareça uma calculadora, é um jogo de combinações. Possui uma função para operar em diversos modos, o que permite combinar notas, gravuras etc.







**Simon**
A unidade gera uma seqüência de notas musicais e luzes. Os quatro quadrantes coloridos, na parte superior, atuam como interruptores tanto para os sons quanto para as luzes. O objetivo do jogo é repetir exatamente a seqüência.



**Texas Instruments**
Essa empresa entrou no mercado educacional na metade da década de 70, com o Little Professor, que propunha problemas aritméticos. Antes dos anos 80 já comercializava um aparelho que ensinava ortografia, utilizando um chip sintetizador de voz. Depois essas técnicas foram empregadas em jogos aritméticos elementares e outros dispositivos.






**Robo-1**
Consiste num braço de robô de desenho convencional, controlado por dois joysticks. Ele não opera sob controle de programas. Mas com um pouco de engenhosidade, pode ser ligado a microcomputador, por meio de interface. O braço utiliza feedback visual e controle por parte do usuário, em vez de usar motores graduais de precisão.




**Big Trak**
Embora pareça apenas um resistente carrinho de brinquedo, é na realidade um robô do tipo que anda no chão. Para programá-lo, digitam-se os códigos de direção e distância num teclado. Com alguma habilidade, liga-se um microcomputador convencional ao Big Trak, por uma porta paralela ou serial. Neste caso, um programa controla o veículo, o que possibilita a passagem para um subprograma diferente, caso ele se defronte com determinado conjunto de condições.



# Revisão eletrônica

## Programas verificadores de ortografia, gramática e estilo já podem ser usados com muitos processadores de palavras.

Os engenheiros de computadores ainda estão longe de criar máquinas com capacidade de gerar e manipular linguagens naturais, como o português. Mas uma das aplicações planejadas para os computadores é a tradução por máquina de idiomas — do português para o japonês, por exemplo. Existem funções para tradução por máquinas, mas apenas de textos simples, como atas e relatórios. Além de sofrer essa limitação, qualquer tradução produzida pelo computador principal precisa ser corrigida a mão. As histórias de erros são muitas. Dizem que a frase evangélica "The spirit is willing, but the flesh is weak" ("O espírito está pronto, mas a carne é fraca") foi traduzida do inglês para o russo e deste para o inglês, por dois programas diferentes, com o seguinte resultado final: "The wine is agreeable, but the meat is spoiled" ("O vinho está agradável, mas a carne, estragada"). Histórias desse tipo ilustram um ponto muito importante — as dificuldades encontradas quando um computador processa informação sem "entender" o que ela significa. Um problema proposto com freqüência a estudantes de computação é determinar como um computador poderia distinguir os significados de frases como estas:

A AMA GOSTA DE MOEDA CUNHADA.
MINHA CUNHADA AMA SEU MARIDO

A construção das duas frases parece idêntica, mas, no primeiro exemplo, AMA é sujeito, enquanto, no segundo é predicado. CUNHADA também tem um sentido em cada frase. A experiência constitui o único meio de diferenciá-lo. Havendo memória suficiente, pode-se simular experiência no computador, mas isso já foge para o campo da inteligência artificial, área em que a pesquisa ainda não avançou até esse ponto. O que se evidencia é a diferença entre sintaxe e semântica. A primeira, que é o conjunto de regras referentes aos processos de construção usados numa língua, constitui área de fácil domínio para os computadores. Um programador que já tenha encontrado mensagens do tipo SYNTAX ERROR? sabe disso. A semântica, no entanto, diz respeito ao significado que aquelas frases e construções transmitem.

Na década de 50, o lingüista americano Noam Chomsky desenvolveu a base da teoria contemporânea sobre linguagens humanas e regras gramaticais. Embora ele não estivesse diretamente envolvido com computação, suas teorias são pertinentes tanto à tradução por máquina quanto ao trabalho de intérpretes e compiladores de linguagens de programação.

Um dos resultados de sua pesquisa foi a criação de vários instrumentos de software utilizados para escrita de textos. Além de pacotes para processamento de palavras utilizados na criação, edição e impressão de textos, existem programas que verificam textos à procura de possíveis erros de ortografia e datilografia, e até para verificar a correção gramatical e o estilo. Embora nenhum dos produtos atuais apresente qualquer coisa de notável no que diz respeito à inteligência artificial, convém examinar seu modo de operar, a forma em que se apresentam ao usuário e sua programação interna.

Todos os programas verificadores da correção das palavras usam um dicionário ortográfico gravado em disco que armazena até 50.000 palavras. Muitos pacotes permitem o acréscimo ao dicionário de outros itens, como termos de jargão profissional ou nomes de empresas e produtos que você possa querer que sejam verificados.

No entanto, encontrar espaço adequado na memória para um dicionário completo constitui sério problema. Um byte, que é formado por 8 bits, contém um único caractere alfanumérico, quando se usa o código ASCII. Assim, mesmo considerando uma média otimista de apenas cinco caracteres por palavra, um dicionário de 30.000 palavras exige 150 Kbytes de memória. Isso supera a capacidade normal das unidades de disco para microcomputadores. Recorre-se então a duas técnicas para comprimir essa espécie de informação.

Suponhamos que nosso dicionário precise conter apenas letras minúsculas (uma rotina no programa fará as conversões), e que, não sendo necessários símbolos numéricos e alguns de pontuação, eles sejam removidos pelo programa. Assim, podemos construir todo o dicionário usando um máximo de 32 caracteres diferentes, em lugar dos 128 do conjunto ASCII completo (ou 256, se incluirmos símbolos gráficos). Portanto, pode-se reduzir o limite de armazenamento de cada caractere de 8 para 5 bits. A palavra "computar", por exemplo, pode ficar armazenada num total de 40 bits (ou 5 bytes). Os primeiros 5 bits do primeiro byte especificam a letra "c", e os 3 próximos bits mais os 2 primeiros do segundo byte especificam a letra "o", e assim por diante.

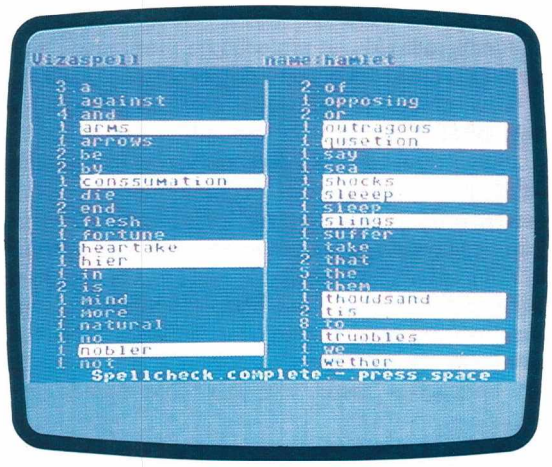
Outra técnica empregada nos verificadores de ortografia é a "tokenising" (simbolização). Ela funciona sob a premissa de que certas combinações de caracteres aparecem com tanta freqüência que podem ser representadas por um único byte, que seria "sinalizado" de maneira a indicar que se trata do símbolo de um grupo de caracteres, e não de um único caractere. Seu microcomputador, com certeza, usa simbolização em BASIC — cada palavra-chave, como PRINT ou NEXT, fica armazenada na RAM como um único byte para poupar espaço.

No dicionário de um verificador de ortografia,

usa-se a simbolização na frente das palavras. Consideremos, por exemplo, o grande número de termos (em inglês) que começam com auto-, non-, dis-, ou con-. O pacote VizaSpell, próprio para o processador de palavras VizaWrite do Commodore 64, emprega compressão e simbolização para condensar um dicionário de 30.000 palavras em apenas 65 Kbytes armazenados num disco.

Procurar todas as palavras de um documento em seu dicionário constitui o trabalho mais difícil de um verificador de ortografia. Uma busca binária pode ser empregada mas, no caso de documento com 1.000 palavras, isso levaria horas. O ideal seria que o processador de palavras conferisse palavra por palavra à medida que elas fossem digitadas. Como isso é impraticável em programação, o documento deve ser verificado como um todo, seja em disco ou em RAM (quando se utilizam máquinas maiores). O programa processa o documento e compila, em ordem alfabética, uma lista das palavras. Em geral, mais de 50% de um relatório é constituído por apenas cem palavras diferentes.

A maioria dos programas verificadores de ortografia recorre a esse processo para fornecer um relatório adicional sobre o emprego de palavras no documento, o que pode ajudar o usuário a evitar repeti-



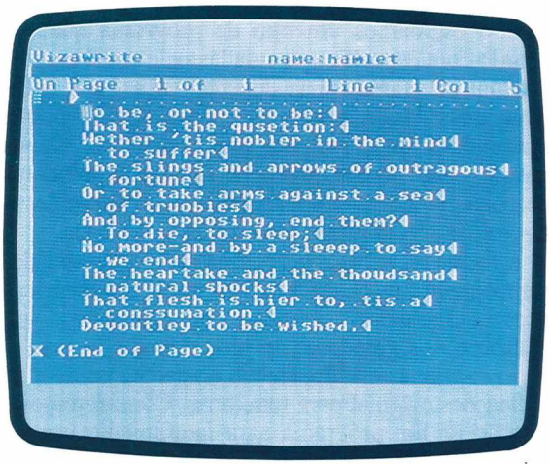

ções desnecessárias. Assim, um simples algoritmo passa por essa lista e pela lista do dicionário, ao mesmo tempo, para que se formem os pares. O tempo utilizado para completar a busca será bastante reduzido e constante — exatamente 4 minutos no caso do VizaSpell, não importando o tamanho do documento.

As palavras não encontradas no dicionário serão impressas em forma de lista ou destacadas no documento original. Para cada palavra salientada, o usuário recebe três opções:

1) A palavra foi escrita ou digitada de forma incorreta e deve ser corrigida.
2) A palavra está correta e deve ser acrescentada ao dicionário do programa.
3) A palavra está correta, mas é improvável que seja usada novamente; não deve, portanto, ser acrescentada ao dicionário. (Isso pode ocorrer sobretudo quando se está trabalhando com endereços.)

Verificadores de gramática e estilo trabalham de maneira semelhante. Os primeiros recorrem a um número limitado de regras (por exemplo, procurar uma letra maiúscula no começo de cada sentença); em conseqüência, muitos erros gramaticais passam despercebidos. Quase todos os pacotes disponíveis de verificadores de estilo empregam um grande dicionário de exemplos a fim de identificar erros de sintaxe e expressão. Têm elementos, portanto, para sugerir construções alternativas às frases desajeitadas que encontram. Eles também costumam chamar a atenção para o uso excessivo de determinada ex-






**Ser ou não ser**
Esse famoso monólogo de Hamlet ilustra o funcionamento da revisão eletrônica. Primeiro, digita-se o texto no computador, usando um processador de palavras. Em seguida, o verificador de ortografia é solicitado por um par de comandos simples, o que cria uma lista em ordem alfabética de todas as palavras usadas, indicando também sua freqüência. A lista é conferida com o dicionário do disco e as palavras desconhecidas ou incorretas são evidenciadas. O programa também destaca palavras aparentemente comuns, mas que estão sendo usadas pela primeira vez e que convém acrescentar ao dicionário para uso futuro.

pressão dentro de um parágrafo, ou o uso de frases longas e deselegantes.

Escrever em linguagem BASIC uma forma simples de programa verificador de ortografia, gramática ou estilo constitui exercício muito interessante, mesmo para programadores inexperientes — embora você precise de um bom conhecimento das funções que manipulam as variáveis alfanuméricas em sua máquina. Com uma sofisticação maior do software, podemos esperar que, em breve, os pacotes para processar palavras já venham com essas funções.

# Gerador de aplicações

**Semelhante ao gerador automático de programas, ele tem aplicações lúdicas, além das comerciais.**



**Pinball**
Este pacote é um tipo de gerador de aplicações para jogos. O usuário desenha o layout e estabelece a lógica para o jogo usando um menu de objetos e diversos instrumentos graficamente representados.

Em artigo anterior do MICROCOMPUTADOR — CURSO BÁSICO estudamos um tipo de programa que, de acordo com especificações dadas pelo usuário, produz um programa capaz de executar as aplicações planejadas. Usam-se geradores de programas com a maioria dos microcomputadores comerciais; alguns pacotes são adequados para microcomputadores pessoais, embora o tipo de aplicação a que se destinam exija, no mínimo, uma unidade de disco.

A forma mais comum de gerar programas que desempenhem funções específicas implica o recurso a pacotes chamados '' geradores de aplicações''. Ao contrário dos geradores de programas, eles produzem programas que dependem do pacote gerador de aplicações original para funcionar. Consideremos a criação de um programa para faturamento usando esses dois tipos de gerador. Assim poderemos destacar as diferenças entre eles.

Se usarmos um gerador de programa, em primeiro lugar o software é carregado do disco para o computador. Quando o usuário tiver respondido a todas as perguntas referentes a arquivos, registros, campos, relações matemáticas, layouts da tela e relatórios impressos que deverão ser feitos (isto é, tiver especificado o programa aplicativo necessário), o gerador pede que se insira um disco virgem na unidade de disco. Neste segundo disco, ele grava o programa que acabou de ser gerado. Repetindo-se o processo, fazem-se cópias do programa de faturamento para todas as filiais da empresa.

Por contraste, um gerador de aplicações parece, de início, menos satisfatório. Quando você tiver completado o estágio de especificações, as rotinas necessárias e o gerador serão gravados no mesmo disco. Como alternativa, o gerador de aplicações pode gravar o programa em disco separado, mas de tal forma que o disco gerador original continuará necessário para se executar a aplicação. Embora se possa empregar uma única cópia do pacote original para produzir ilimitadas aplicações diferentes, todas elas devem ser usadas no mesmo espaço físico do disco gerador. Se você colocar sua aplicação à venda, seus clientes precisarão comprar também uma cópia do gerador. (Esses geradores empregam diversos métodos de proteção de programas, a fim de dificultar a execução de cópias sem autorização.)

Um gerador de aplicações é só um programa de uso geral. Quando você especifica sua aplicação, apenas atribui valores a diversas variáveis importantes do gerador, chamadas ''parâmetros''. Estes controlam o fluxo do programa, a estrutura da informação e os layouts para tela e impressora. Quando você grava a aplicação num disco, está armazenando uma lista de parâmetros. Essa lista — também chamada de ''módulo de aplicação'' — equivale, portanto, a um conjunto de instruções que indicam ao gerador de aplicações como executar determinada aplicação.

Alguns pacotes passam para outra etapa e permitem que você especifique sua aplicação na forma de uma linguagem de nível muito alto (semelhante à pseudolinguagem que usamos de início para desenvolver uma nova rotina no curso de programação em BASIC). O gerador interpreta essa listagem, operação que pode ser repetida pelo interpretador BASIC, caso o programa gerador esteja escrito nessa linguagem. (Isso cria um caso interessante de hierarquia em software.)

Eventualmente, empresas que não respondem pela autoria dos geradores originais criam e comercializam os módulos de aplicações. Por exemplo: considera-se o dBase II (o mais popular dos sofisticados pacotes de bancos de dados usados com microcomputadores) um gerador de aplicações com módulos que consistem em conjuntos de comandos de bancos de dados de alto nível. Módulos para aplicações mais particulares — tais como um sistema de contabilidade destinado a corretores da bolsa — podem ser construídos sem que se escreva o programa desde o início. Tendo em vista a limitação do mercado, um pacote para corretores da bolsa executado sob o comando do dBase II mostra-se mais funcional

que um escrito em BASIC. No primeiro caso, o autor do programa concentrou todos os seus esforços na *operação* do programa e não no ato de escrever o código. As partes do programa mais suscetíveis a violações (por exemplo, o manuseio do arquivo) são escritas pelos autores do gerador e testadas em diferentes aplicações por milhares de usuários.

A principal diferença, porém, entre um gerador de programas e um gerador de aplicações reside na capacidade de agradar ao usuário. O programa final criado pelo primeiro tipo de pacote consiste num código artificial, quase sempre escrito numa linguagem como o BASIC. Esse código ainda é inferior, tanto em eficiência quanto em estilo, àqueles criados pelo homem. Com o gerador de aplicações, até 99% do programa final consiste em código escrito pela software house, provavelmente em linguagem de máquina. Esse é o caso do Silicon Office, um dos geradores de aplicações mais sofisticados e fáceis de se usar em microcomputadores comerciais. O programa resultante mostra-se rápido e eficiente, tem procedimentos incorporados para detectar erros do operador e produz displays de tela nítidos, com menu.

Além disso, os geradores de aplicações não se restringem a programas comerciais. No Pinball Construction Set (ver p. 241), um ótimo exemplo de pacote não-comercial, o módulo de aplicações vem bem especificado, dispondo na tela os elementos necessários para a mesa desse jogo eletrônico.

Existe muita coincidência entre as programações orientadas para o assunto e para o objetivo (ver p. 242), que se resume em incentivar o programador a executar suas aplicações pela simples especificação dos objetivos exigidos do programa. Até um simples programa para folha eletrônica, próprio para microcomputadores pessoais, é considerado um gerador de aplicações — se você especificar as relações entre os diversos campos, o pacote faz todo o trabalho de rotina.

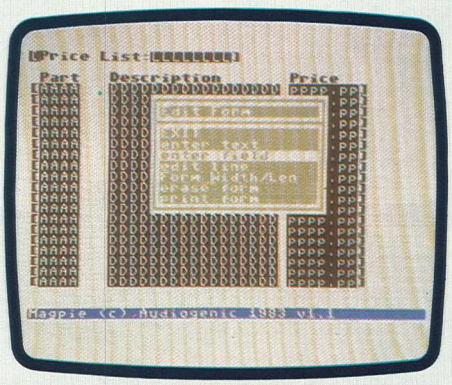
O Magpie — produzido pela Audiogenic para o Commodore 64 com uma unidade de disco — é um gerador de aplicações montado para uso comercial. Também esse pacote emprega bem a programação orientada para objetivos visuais: especifica as relações dos campos de informação entre diferentes registros, quando se desenha o layout dos arquivos.

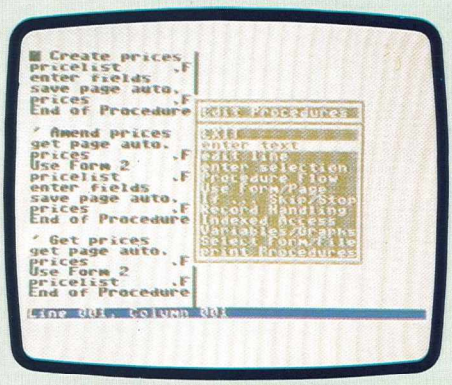


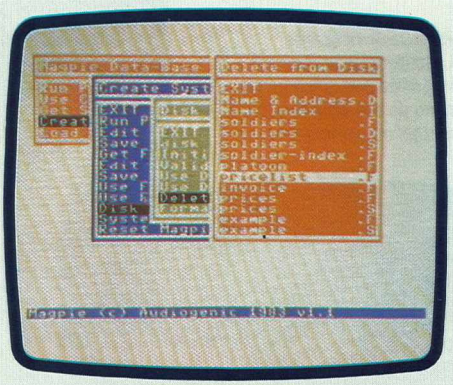

**Jogos complexos**
Os geradores de programas e os geradores de aplicações vêm propiciando o desenvolvimento de atividades lúdicas (como os fliperamas) cada vez mais complexas.

Embora, a rigor, não sejam considerados geradores de aplicações, um número crescente de pacotes vêm incorporando esses princípios. Quando executados pela primeira vez, os programas à base de parâmetros fazem uma série de perguntas ao usuário e gravam as respostas em disco ao lado dos programas. Essas informações determinam alguns detalhes da operação do programa. Um programa de faturamento, por exemplo, faz perguntas relacionadas com informações que a empresa quer que constem das faturas e sobre os prazos de pagamentos por ela determinados. Um jogo de fliperama pode perguntar com quantos alienígenas e foguetes o usuário gostaria de começar, ou até dar-lhe a oportunidade de desenhar os invasores.

O software vem sendo projetado de forma que o usuário não tenha de aprender programação e, ao mesmo tempo, proporciona alto grau de flexibilidade na operação. Numa situação ideal, o software se ajusta às exigências do usuário (em vez de o usuário ajustar-se ao software).




## Magpie no Commodore 64

A primeira etapa na criação de uma aplicação consiste em especificar o layout de todos os formulários, transações e relações, tal como uma lista de preços.

Em seguida, ele especifica todos os cálculos e processos, na forma de uma lista de instruções. Aqui são representadas as rotinas para alterar os preços e trazê-los (GET) do disco para a tela.

O Magpie funciona com menu. Aqui a tela mostra o resultado após selecionar CREATE, depois DISK, DELETE e o arquivo a ser removido, que no caso é a PRICE LIST (lista de preços).

# Texto e computação

**Quem trabalha com a palavra escrita persegue a informação e/ou a qualidade do texto. Em ambos os casos, os micros vêm se revelando de grande valia, inclusive no Brasil.**

À medida que seu uso se difunde e se tornam mais conhecidas suas possibilidades, os microcomputadores passam a ser mais usados pelos profissionais da palavra escrita. Entre eles estão os jornalistas e os escritores, para os quais os micros oferecem vários recursos.

No entanto, convém verificar uma distinção importante. Para isso, temos dois exemplos. "Certo deputado paulista, numa dessas tardes, talvez haja sussurrado qualquer coisa ao ouvido de um colega mineiro" é afirmação que nunca seria feita por jornalista competente. Já um escritor do porte de Jão Guimarães Rosa termina sua obra-prima *Grande sertão: veredas* da seguinte maneira: "Nonada. O diabo não há! É o que eu digo, se for... Existe é homem humano. Travessia".

No trabalho do jornalista o que deve predominar é a informação, sua quantidade e exatidão, ao passo que no trabalho do escritor a informação interessa menos em sua quantidade e mais em seu tratamento, que é o fundamental no texto literário. O instrumento de trabalho de cada um deve atender a essas exigências.

Embora em alguns países o uso de micros por escritores seja comum, sobretudo para certos experimentos de ficção científica, no Brasil ele ainda se mostra muito raro. Um dos poucos exemplos da utilização da informática para a criação de um texto literário entre nós seria o conto *Otávio e Marília*, do paulista Renato Pompeu, em que a utilização do videotexto permite a participação ativa do leitor no andamento e na solução do enredo.

Já entre os jornalistas brasileiros, o uso do microcomputador mostra-se mais disseminado.

Carlos Lovizzaro, repórter da revista *Dados & Idéias* (especializada em informática), revela-se um entusiasta das vantagens oferecidas pelos micros no desenvolvimento de seu trabalho. "Em primeiro lugar, a velocidade do microcomputador é muito maior, fator que se torna importante na dinâmica



**Literatura de vanguarda**
A escritora Lilian Prist e o programa para a leitura em micro de sua obra *O impacto pessoal do computador.*

```
1200  SPEED= 20
1210  HOME
1220  PRINT "MAIS CONTEXTO,MAIS C
   ONTATO"
1222  PRINT "..................
      ..."
1224  PRINT "SAO MEIO DE COLOCAR
   VOCE
1225  PRINT "EM CONTATO COM PALAV
   RAS,
1226  PRINT "EXPRESSOES E IMAGENS
    DIRETAMENTE
1228  PRINT "LIGADAS AO MICRO,DEN
   TRO DE"
1229  PRINT "SITUACOES TOTALMENTE
    DIFERENTES"
1232  PRINT "DAS QUE VOCE ESTA AC
   OSTUMADO A "
1233  PRINT "A ENCONTRAR."
1234  PRINT "E O MICRO ENTRANDO N
   O SEU "
1235  PRINT "MUNDO.
1236  PRINT "..................
      ......"
```

jornalística. Da máquina de escrever ao terminal, muita coisa passará por grandes revoluções tecnológicas a partir da informática, sobretudo no processamento de textos: a velocidade da digitação, os maiores recursos para a correção ortográfica e a abolição total da tesoura. Com a informática ficam eliminadas as manipulações com papel e o resultado final é um trabalho mais limpo e preciso.''

Na opinião de Lovizzaro — que desde maio de 1984 opera um CP 500 com dois drives e uma impressora P-500 —, o computador começa a aumentar a produtividade do jornalista depois de três ou quatro meses de prática. Primeiro é preciso desvendar a máquina e saber, por exemplo, para que serve um drive ou um disquete. ''A gente começa a fazer isso quase brincando. Depois entra nos programas propriamente ditos.''

Para ele, um dos problemas que surgem logo de início está nos manuais que acompanham o computador. ''O manual não é escrito tendo em vista a pessoa que vai processar um texto. Ele apenas apresenta para o usuário o potencial do programa, em termos gerais. Por isso é preciso traduzir a linguagem em si. Só depois começa a fase de acumulação e armazenamento de informação.''

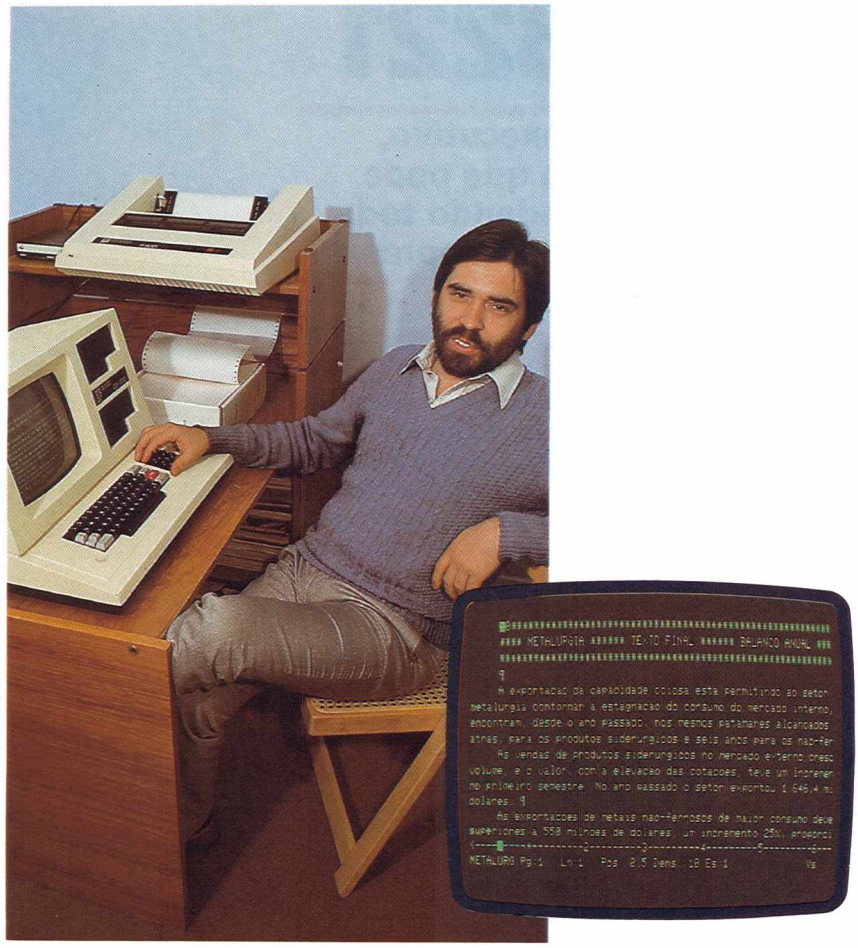
Após esse tempo de adaptação, o usuário ultrapassa a produtividade da máquina de escrever e as vantagens começam a se mostrar. O editor de textos, ao mesmo tempo que edita, grava suas matérias e pode utilizar a memória da máquina para guardá-las, tendo acesso imediato a elas no momento em que forem necessárias.

Lovizzaro explica que, estabelecendo-se uma hierarquia, o banco de dados viria em segundo lugar, por depender de informações. Mas ''para quem trabalha com mais de cem fichas, em teses e pesquisas, já convém partir para o banco de dados''. Ele ressalta que, com essa aplicação, é possível a manipulação de informações (enquanto o editor de textos manipula idéias).

Segundo Lovizzaro, o jornalista especializado não pode permanecer como mero observador da informática: deve se colocar na posição de usuário, pois esta é a única forma de aprendê-la e vivenciá-la. ''E isso é extensivo a todos os outros jornalistas, considerada a profunda transformação que vai ocorrer na imprensa.''

O editor-responsável pela seção ''Dinheiro Vivo'' da *Folha de S. Paulo* e da Abril Vídeo, Luís Nassif, também utiliza um CP 500 para elaborar suas complexas tabelas de indicadores econômicos, que abrangem as variações de ORTN, UPC, INPC, dólar, ouro etc. ''Não dá mais para imaginar como seria meu trabalho sem o computador'', afirma Nassif, que não é um iniciante no ramo. ''Meu contato com micros aconteceu em 1981, quando comecei a preparar os primeiros programas na área financeira, num Dismac 80. Só mais recentemente é que a gente começou a processar texto, mas os softwares são caríssimos e deixam muito a desejar.''

Mesmo assim, enquanto demonstra alguns recursos de seu CP 500 (facilidade na diagramação, na correção ortográfica e no cálculo de variáveis econômicas), Nassif ressalta que está satisfeito com o equipamento, bastante avançado em relação a seu antigo Dismac que lhe ensinou o BASIC, mas não tinha a capacidade de memória de que ele necessita. Ao mesmo tempo, há quem utilize o micro em trabalhos na área de literatura. Para a escritora brasileira Lilian Prist — autora de *O impacto pessoal do computador*, que descreve sua trajetória na área da informática —, um dos fatores mais importantes para quem utiliza o micro é a possibilidade de programação dos espaços. Operando um Unitron AP acoplado a uma IBM elétrica, ela constatou que ''com a máquina de escrever acontece certa limitação quando o autor deseja modificar o texto. Parece que, quando uma palavra é escrita no papel, ocorre uma fixação e, conseqüentemente, um aprisionamento. Na tela, pelo contrário, pode-se realizar inserções tanto entre letras quanto entre linhas. O acréscimo ou a eliminação de palavras, frases ou parágrafos torna-se possível sem a reescrita do texto''.




**Um salto incrível**
O jornalista Carlos Lovizzaro considera um salto à frente a computação aplicada a sua profissão. "Ainda teremos muito trabalho para aproveitar todo o potencial que essas máquinas podem nos oferecer", frisa ele.

As etapas que considera mais mecanizadas na linguagem (e por isso mesmo irrelevantes do ponto de vista do conteúdo) são deixadas para o computador. Aumentam, assim, as possibilidades de utilização da capacidade criativa de quem escreve.

Como todo programa de processamento precisa ser conhecido antes de o autor começar a manipular o texto, Lilian acha que o computador exige do indivíduo uma interação e um esforço no qual cada um irá despender um tempo diferente para assimilar o processo. Mas para ela o fundamental — a ser feito pelos meios de comunicação — é colocar as pessoas em contato com a informática, para que se compreenda que essa interação é tão ampla quanto as possibilidades do ser humano.

# Labo 8221

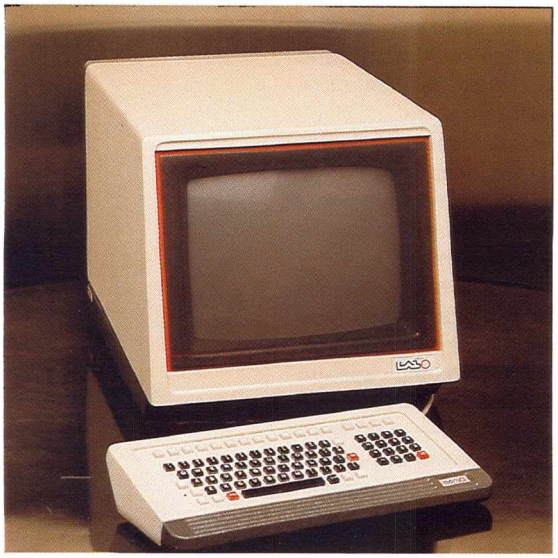
## Para a mesa do executivo, um equipamento que pode se expandir, atendendo toda a administração da empresa.

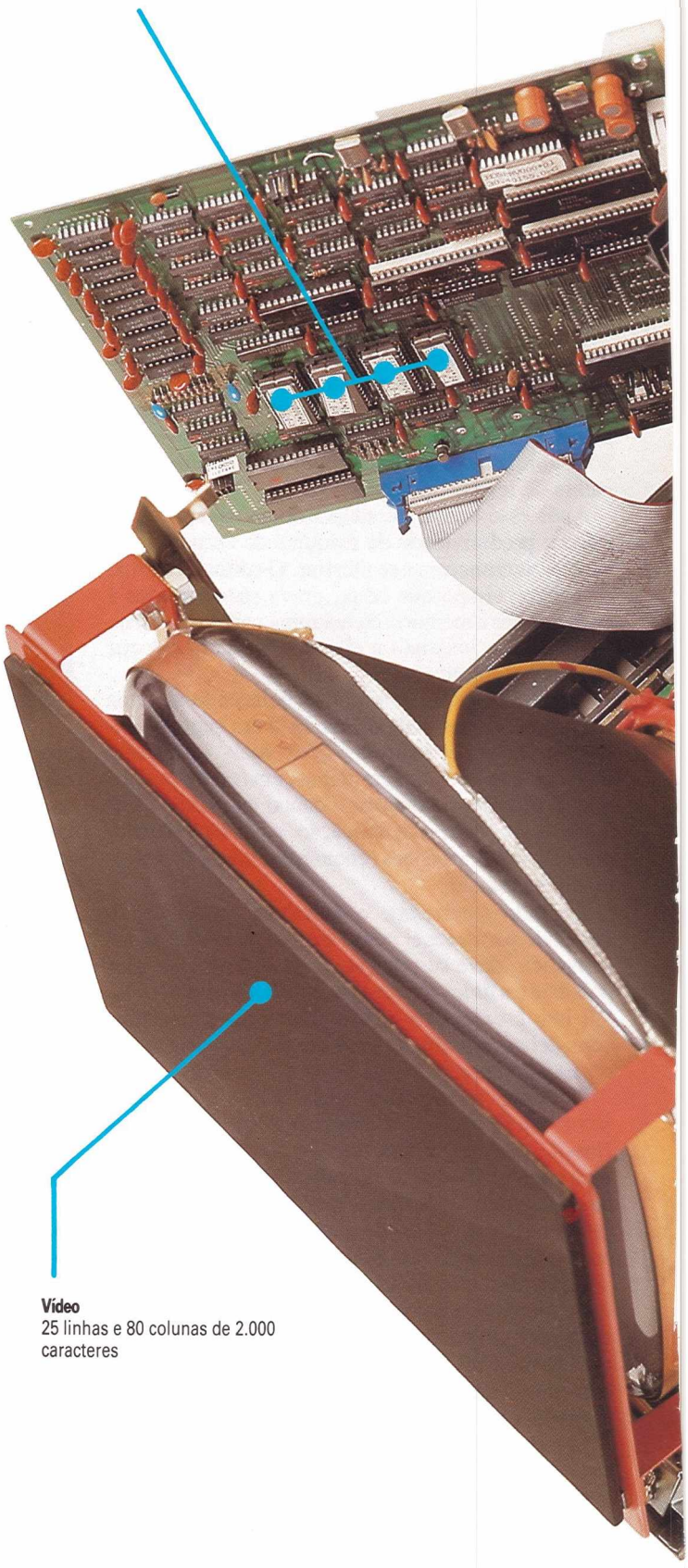
Os microcomputadores Labo foram concebidos tanto para atender ao uso pessoal/profissional quanto ao geral de pequenas, médias e grandes empresas. Os equipamentos, por suas características, funcionam em ambiente multiusuário que requeira operações simultâneas. Ágeis, robustos e de fácil expansibilidade no próprio local de instalação, oferecem a solução global para os problemas da administração empresarial (serviços, produção, comercialização), por causa de sua compatibilidade com as diversas linguagens disponíveis no mercado: CP/M, COBOL, dBase, FORTRAN.

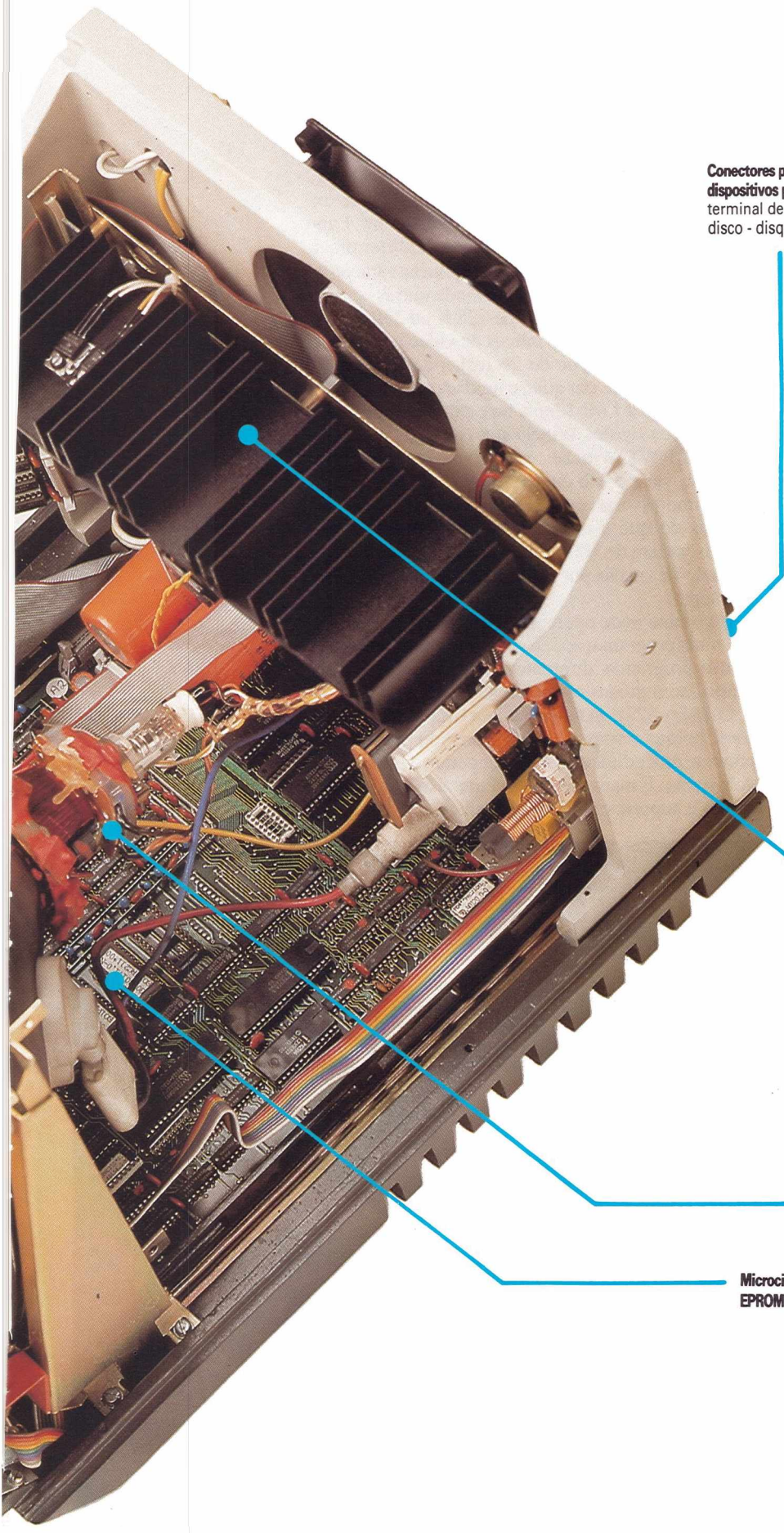
O modelo mais simples da linha, o Labo 8221 XC-Executivo, pode evoluir para o Labo 8221 e o Labo 8221 WT-Winchester, destinados a aplicações gerais, como controle de estoque, faturamento, contabilidade, contas a pagar/receber, folha de pagamento etc.

Na versão mais simples e barata, para uso pessoal do executivo, o micro da Labo, sem perder as características de um processador autônomo, pode atuar como um terminal interativo 3270 dos equipamentos IBM. A troca de um pelo outro não exige grandes investimentos nem requer mudança na estrutura dos sistemas IBM já implantados na empresa.

Outra vantagem do Labo 8221 XC-Executivo é a versatilidade. Como tem sistema operacional compatível com o CP/M, dispõe de todas as facilidades de uso dos softwares mais difundidos no mercado, como o processador de palavras Wordstar e a folha eletrônica SuperCalc. Opera com eficiência consagrados produtos para executivos, como os gerencia-







**Conectores para interface de dispositivos periféricos:**
terminal de vídeo unidade de disco - disquete impressora

**Fonte de alimentação e deflexão**

**Microcircuito de RAM**
128 K

**Microcircuito de memória EPROM de linguagem e monitor**

## Labo 8221 XC-Executivo

**MICROPROCESSADOR**

Z80A

**CLOCK**

4 MHz

**MEMÓRIA**

128 K

**VÍDEO**

25 x 80 caracteres

**TECLADO**

Com todas as letras maiúsculas e minúsculas, caracteres especiais e numéricos. Teclado auxiliar numérico programável e dezessete teclas de funções programáveis.

**LINGUAGENS**

BASIC, COBOL, FORTRAN, dBase, Assembler.

**PERIFÉRICOS**

Duas unidades de discos flexíveis de 5 1/4". Impressora de 80 cps.

**DOCUMENTAÇÃO**

Guia do usuário 8221 XC.

dores de bancos de dados dBase II e Friday ou Financial Planner, o programa mais poderoso para modelos financeiros.

A versão Executivo consegue ainda comunicar-se — seja por linha de transmissão de dados, seja por uso de suporte comum — com os demais micros da série 8200 ou com os mínis da série 8000 da Labo, além de ligar-se com sistemas de grande porte. Assim, é possível receber ou transmitir arquivos ao mesmo tempo em que o micro está rodando outro programa — por exemplo, um simulador financeiro.

A capacidade de multiprogramação deve-se ao sofisticado Sistema Operacional Labo (SOL). Graças a ele, o equipamento é capaz de executar dois programas, mais o "spooler" automático: um em modo "desenvolvimento" ou em modo "operação" e outro em modo "lote". Em termos mais simples, isto significa que ele pode efetuar mais de uma tarefa simultaneamente. Por exemplo: calcula a folha de pagamento ou coleta dados sobre a movimentação de estoque, enquanto na impressora sai a relação dos clientes — um desempenho equivalente ao de dois micros comuns.

Sempre que necessário, o Executivo permite a ligação direta (ou mediante adaptadores) a máquinas de escrever Remington, IBM e Olivetti. Dessa maneira, um diretor pode, por exemplo, recorrer à boa qualidade de impressão da máquina de escrever eletrônica de sua secretária, quando quiser enviar correspondência. Em sua configuração mais simples, esse equipamento vem com uma Unidade Central de Processamento (CPU) com memória de 128 K; terminal de vídeo de 25 linhas x 80 caracteres (as três últimas telas mantidas têm memória); teclado livre do terminal, contendo todas as letras maiúsculas e minúsculas, caracteres especiais e numéricos, além de um teclado auxiliar numérico programável e dezessete teclas de funções programáveis pelo usuário para suas atividades mais freqüentes. As duas unidades de disco comportam apenas disquetes de 5 1/4 polegadas (face simples e densidade dupla), num total de 360 K. E a impressora é matricial de 80 cps.

Para o uso geral da empresa, esse micro se expande modularmente, transformando-se no Labo 8221, que opera em configuração multiterminal. Esta máquina mais poderosa incorpora as características do Executivo e apresenta ainda outras.

A CPU do 8221 pode ser de 128 K ou 256 K. Admite a ligação de três terminais de vídeo e seu teclado profissional dispõe de 26 teclas de funções. A memória externa, em quatro unidades de discos flexíveis, chega a 1 Mb (dupla face e dupla densidade). Com discos rígidos Winchester (ou seja, na versão labo 8221 WT-Winchester), a capacidade do equipamento vai a 5 Mb ou 10 Mb.

As impressoras do 8221 também são mais poderosas do que a utilizada no Executivo. As matriciais variam de 100 a 200 cps. Mas existe a possibilidade de opção por uma impressora de linha de 300 lpm.

O sistema pode operar com um usuário em modo de multiprogramação, rodando simultaneamente quatro programas diferentes. Ou no modo multiusuário, com quatro operadores.

O Labo 8221 garante a segurança das informações, porque o acesso a ele é protegido por um conjunto de procedimentos que asseguram a integridade do sistema. E mais: o equipamento dispõe de atendimento e suporte em todo o Brasil.



# Código de ordenação

**Quando se trabalha com matrizes extensas, a classificação Shell é mais eficiente que as classificações bolha ou por inserção. A Shell opera pela divisão dos dados em séries.**

Examinamos, na página 286, dois métodos de classificação de matrizes — as classificações bolha e por inserção. Em geral, a bolha revela-se mais fácil de ser desenvolvida, mas a classificação por inserção opera com maior rapidez. A experiência com ambos mostra que boa parte do tempo se despende com a troca de cartas a curtas distâncias.

```
7999 REM**********************
8000 REM* SHELL *
8001 REM
8025 PRINT "CLASSIFICACAO SHELL - INICIAR ! ! ! ! ! "
8050 LET LK=LT
8100 FOR Z=0 TO I STEP 0
8150 LET LK=INT (LK/II)
8200 FOR LB=I TO LK
8250 LET LL=LB+LK
8300 FOR P=LL TO STEP LK
8350 LET D=R(P)
8400 FOR Q=P TO LL STEP-LK
8450 LET R(Q)=R(Q-LK)
8500 IF D<=R(Q) THEN LET R(Q)=D:LET Q=LL
8550 NEXT Q
8600 IF D>R(LB) THEN LET R(LB)=D
8650 NEXT P
8700 NEXT LB
8750 IF LK=I THEN LET Z=I
8800 NEXT Z
8850 PRINT "CLASSIFICACAO SHELL - ENCERRAR ! ! ! ! ! "
8900 RETURN
```

Para incluir essa rotina no programa do procedimento de classificação, na página 287, altere a linha 350 para:

```
350 LET I=1:LET O=0:LET II= 2:LET TH=3
```

e a linha 900 para:

```
900 ON SR GOSUB 6000,7000,8000
```

Um método mais eficaz que esses dois é a "classificação Shell". Esse método garante que a desordem na matriz seja reduzida no começo da classificação (de modo que os itens não fiquem muito distantes de suas posições efetivas) e possibilita a realização das trocas a distâncias longas. Um método para essa classificação inclui cinco procedimentos:

1) Disponha todas as cartas de um naipe em ordem decrescente, de modo que o rei seja a carta da extrema esquerda, e o ás, a da extrema direita. Conte as cartas, divida o número resultante (no caso, 13) por 2, deixando de lado a que restar, e registre o resultado (6) num pedaço de papel assinalado "Elo".

2) Coloque uma moeda pequena sob a carta da extrema esquerda (posição Um) e uma moeda maior na posição Elo (isto é, posição Seis, na primeira vez). Todas as cartas, da primeira até a posição Elo, serão as cartas da extremidade esquerda numa série de "correntes" de cartas. O número de correntes coincide com o valor do Elo naquele momento. Forma-se cada corrente a partir da carta da extremidade a que pertence, somando-se o Elo ao número de posição da carta da extremidade. Consegue-se desse modo a posição da carta seguinte, à qual se acrescenta o Elo para se obter a posição da próxima carta e assim sucessivamente, até que o final da matriz seja alcançado ou ultrapassado. A primeira corrente, desse modo, abrange as cartas nas posições Um, Sete e Treze; a segunda corresponde às cartas nas posições Dois e Oito e a terceira equivale às cartas nas posições Três e Nove. A última coincide com as cartas que ocupam as posições Seis (o valor do Elo no momento) e Doze.

3) Agora, tendo assinalado os limites com as moedas, desloque para fora da tabela as cartas incluídas na primeira corrente, de modo que possam ser vistas isoladamente, e proceda a sua ordenação, utilizando a classificação bolha ou por inserção, como descrito na página 286 (a listagem, nesse artigo, utiliza o método de inserção).

4) Desloque a corrente ordenada de volta a suas posições na matriz e repita o procedimento acima com a corrente seguinte, e assim por diante, até que todas as correntes cujas cartas da extrema esquerda caiam entre as moedas estejam ordenadas.

5) Quando as correntes estiverem ordenadas, divida o Elo por 2, deixando de lado a que restar. Se o Elo for agora menor que 1, a matriz estará classificada. Caso contrário, repita a operação a partir do procedimento 2 com o novo valor do Elo.

## Painel da classificação Shell

| Posição n.° | Valor de Elo | Comentários |
|---|---|---|
| 1 2 3 4 5 6 7 8 9 | | |
| 2 8 9 3 T 5 K 6 7 | (9/2)=>4 | Início da passagem |
| * + @ $ * + @ $ * | | Formação de correntes |
| T 7 2 | | Corrente de classificação 1 |
| 8 5 | | Corrente de classificação 2 |
| K 9 | | Corrente de classificação 3 |
| 6 3 | | Corrente de classificação 4 |
| T 8 K 6 7 5 9 3 2 | | Início da passagem |
| T 8 K 6 7 5 9 3 2 | (4/2)=>2 | Fim da passagem |
| * + * + * + * + * | | Formação de correntes |
| K T 9 7 2 | | Corrente de classificação 1 |
| 8 6 5 3 | | Corrente de classificação 2 |
| K 8 T 6 9 5 7 3 2 | | Fim da passagem |
| K 8 T 6 9 5 7 3 2 | (2/2)=>1 | Início da passagem |
| * * * * * * * * * | | Formação da corrente 1 |
| K T 9 8 7 6 5 3 2 | | Encerramento da passagem |

| CHAVE | |
|---|---|
| * | Membro da corrente 1 |
| + | Membro da corrente 2 |
| @ | Membro da corrente 3 |
| $ | Membro da corrente 4 |

**Classificação Shell**
O exemplo da classificação Shell com reduzida mão de cartas apresentado no painel demonstra seu excepcional método de divisão da matriz numa série de correntes (com espacejamentos no número do Elo em determinado momento). Essas cadeias são ordenadas uma a uma, no caso, pelo método de inserção, antes da passagem.

Esta listagem de programa para classificação Shell deve ser usada com o programa-teste da página 287.
Quando o testamos, houve considerável melhora com relação a outros métodos de classificação, sempre que o número de itens a serem ordenados excedia a quarenta.

# Mão única

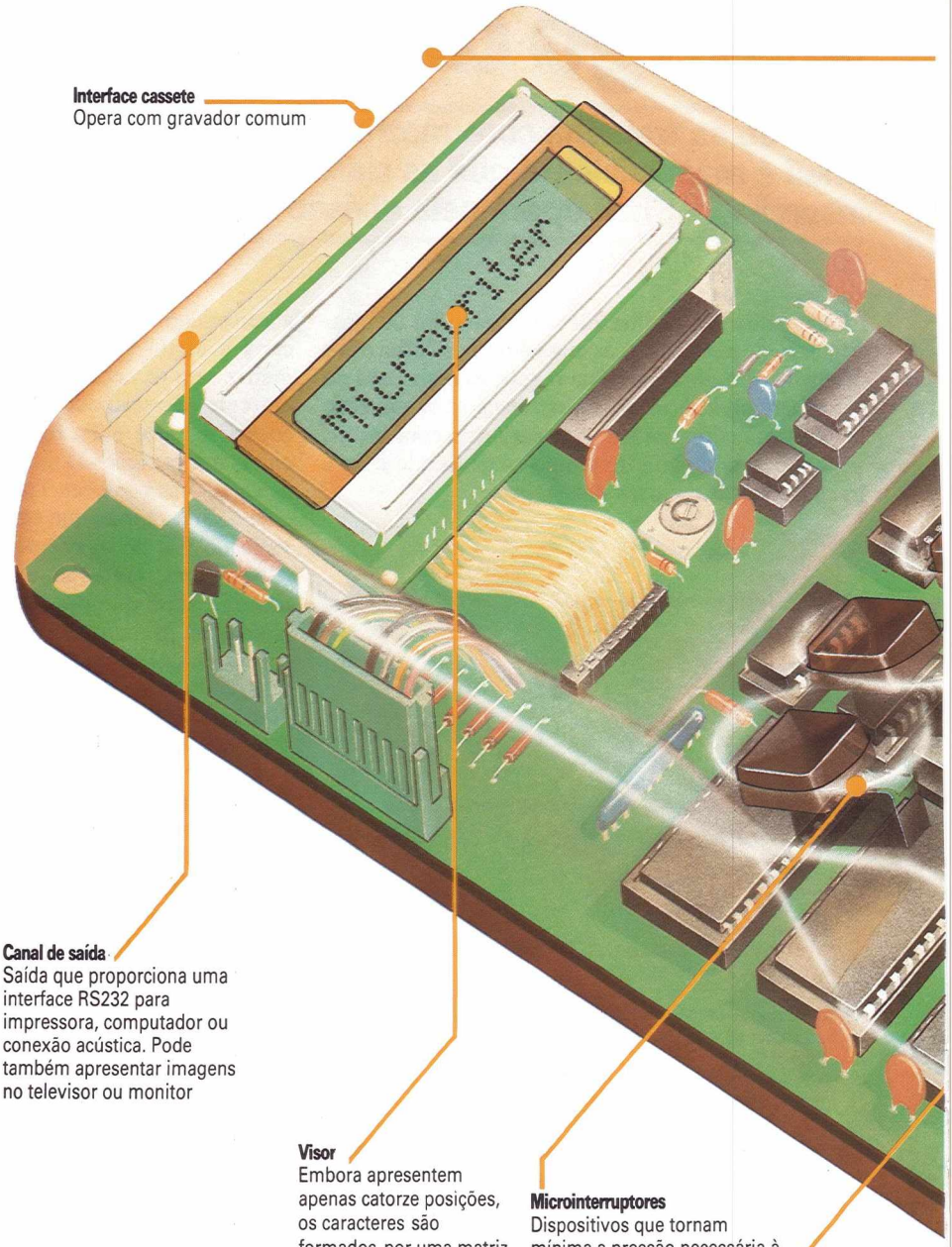
**O Microwriter é um processador de textos portátil operado por uma só mão. Seu painel de seis botões pode ser conectado a computadores.**

Ter em seu escritório um processador de textos ou, em sua casa, um microcomputador com programa processador de textos é uma boa idéia. Além de eliminar o lado aborrecido da elaboração rotineira de trabalhos escritos, cartas e relatórios, auxiliam na documentação de programas, na produção rápida de cópias e no trabalho com uma agenda de endereços. Podem se mostrar tão úteis que, sendo necessário registrar alguma informação, nossos dedos tenderão a procurar o teclado, em vez de lápis e papel. No entanto, surge um problema sempre que se deseja tomar notas, fora de casa ou do escritório, de forma que um computador possa "entender".

Há um mercado crescente para sistemas portáteis de computação, como o Model 100, da Tandy, e o Epson HX-20. Embora estes apresentem a vantagem de funcionar como processadores de palavras portáteis, são pouco eficazes como substitutos dos blocos de anotação ou dos ditafones. Por isso se desenvolveu um equipamento de processamento de texto pequeno o bastante para ser transportado no bolso. Funciona com baterias, exige uma só mão para operá-lo e pode ser conectado a uma impressora ou mesmo outro computador.

Tal dispositivo, denominado Microwriter, está à venda na Inglaterra desde o início da década de 80. Originalmente concebido por Cy Endfield, esse equipamento não usa o teclado do tipo QWERTY, e sim um sistema exclusivo de pressionamento de múltiplas teclas pelo emprego de apenas seis botões. A idéia surgiu da necessidade de criar um jogo controlado manualmente e operado por meio de palavras, para o qual até mesmo um teclado em miniatura seria muito grande e caro. A solução foi criar um tipo especial de teclado que utiliza apenas alguns botões com combinações suficientes para a especificação de todos os símbolos alfanuméricos. O grande avanço se deu com a invenção de um sistema de códigos simbólicos exclusivos para o Microwriter.

Parece impossível que as letras do alfabeto, os algarismos e os sinais de pontuação possam ser criados pela combinação de apenas seis botões. Mas eles bastam; e em poucas horas se toma conhecimento das combinações mais usadas. Os fabricantes têm motivos para afirmar que nesse teclado se aprende a digitação com muito mais facilidade do que nos do tipo QWERTY. A combinação de botões necessária para cada letra baseia-se na forma física da letra, código esse que os não datilógrafos acham mais fácil de aprender. Por ser digitado com uma única mão, o Microwriter também abre perspectivas para deficientes físicos que não podem operar o pressionamento simultâneo de várias teclas, necessário nos teclados convencionais, para geração de comandos.



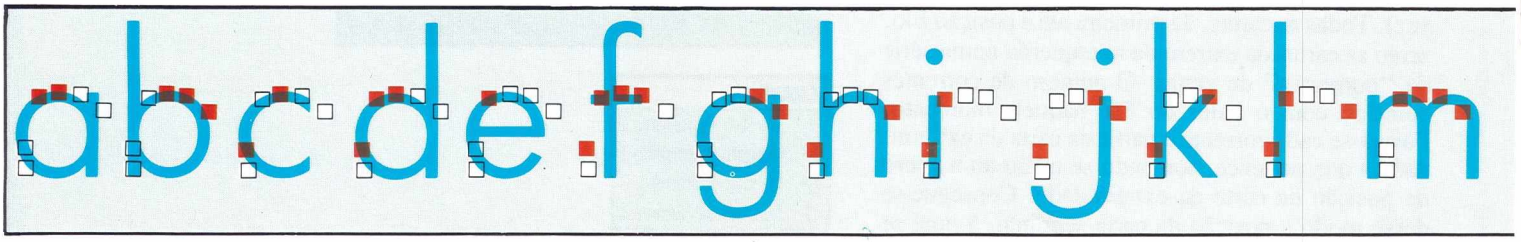
**Interface cassete**
Opera com gravador comum

**Canal de saída**
Saída que proporciona uma interface RS232 para impressora, computador ou conexão acústica. Pode também apresentar imagens no televisor ou monitor

**Visor**
Embora apresentem apenas catorze posições, os caracteres são formados por uma matriz de tamanho grande

**Microinterruptores**
Dispositivos que tornam mínima a pressão necessária à ativação dos botões

**RAM**
A máquina-padrão possui 8 K de RAM, mas pode ser equipada com chips maiores nos mesmos encaixes, para aumentar sua capacidade

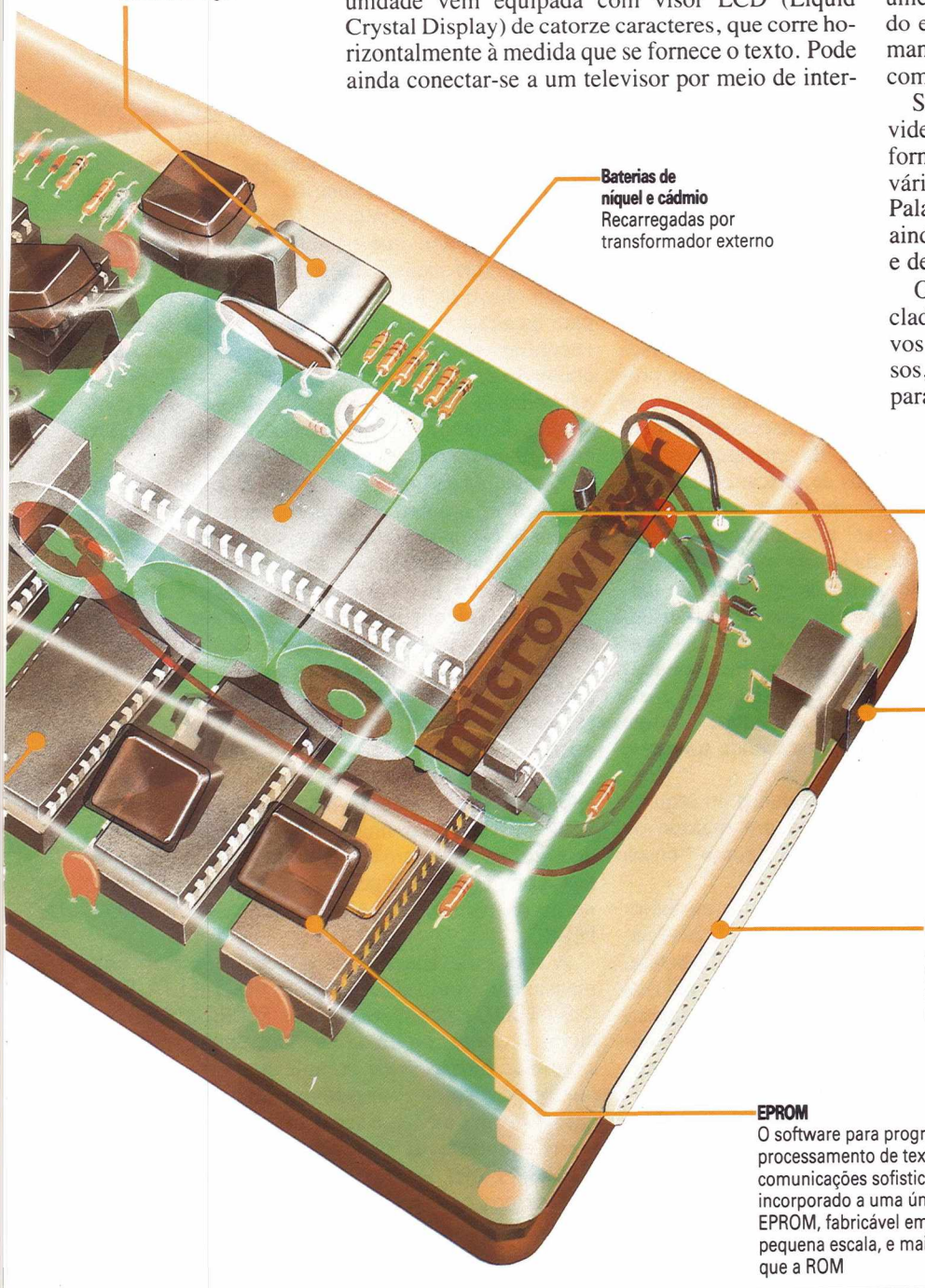


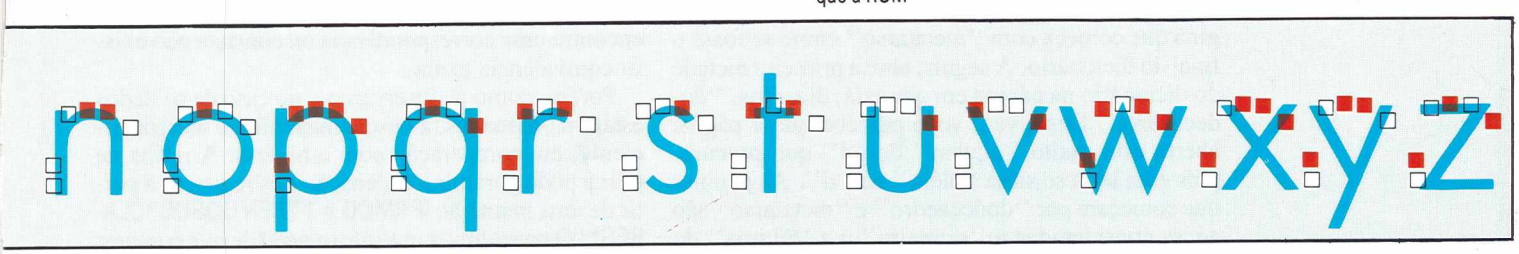
O projeto interno do Microwriter visou a tornar o aparelho tão portátil quanto possível. O microprocessador e sua memória são dispositivos de CMOS (Complementary Metal Oxide Silicon, silício com óxido metálico complementar), que auxilia a reduzir o consumo de energia. Um conjunto recarregável de baterias de níquel e cádmio pode fornecer suprimento de energia suficiente para trinta horas de uso. A fim de apresentar visualmente os caracteres, a unidade vem equipada com visor LCD (Liquid Crystal Display) de catorze caracteres, que corre horizontalmente à medida que se fornece o texto. Pode ainda conectar-se a um televisor por meio de interface opcional. Isso permite *mais tarde* (no escritório ou em casa) a revisão, na tela, do texto armazenado.

Além de uma interface serial RS232, para ser conectada à impressora, uma interface cassete equipa o Microwriter. Isso possibilita que o texto armazenado na memória seja gravado ou fornecido de novo. A interface serial também permite o uso do Microwriter como terminal de computador comum ou processador de palavras controlado por uma única mão. Os documentos digitados fora de casa ou do escritório são fornecidos ao equipamento de tamanho normal para correções ou estruturação mais completa.

Se necessário, o texto contido no Microwriter divide-se em diversos segmentos, o que possibilita o fornecimento e a manipulação em separado de várias unidades de texto não inter-relacionadas. Palavras podem ser incluídas ou eliminadas, ou ainda deslocadas pelo emprego da interface cassete e de um circuito separador temporário.

O objetivo do projetista foi a incorporação do teclado Microwriter de seis botões a outros dispositivos eletrônicos. Contudo, apesar de seus bons recursos, o Microwriter ainda não se mostrou atraente para os fabricantes de microcomputadores.



# Mandado de busca

**O tempo necessário para se localizar um registro pode ser muito reduzido usando-se "busca binária", contanto que o arquivo já tenha sido adequadamente ordenado.**

As três atividades mais importantes do programa da agenda de endereços — acréscimo de novos registros, gravações do arquivo em fita ou em disco e leitura do arquivo a partir do armazenamento em massa, ao se processar o programa pela primeira vez — já foram desenvolvidas. Mas a agenda de endereços será inútil se você só puder acrescentar (e não extrair) dados. Uma rotina que encontre os registros faz-se necessária.

Localizar o registro completo de um nome será com certeza a atividade mais freqüente; portanto, a primeira opção do menu de escolha (*ESCLHA*) é ENCONTRAR UM REGISTRO (PELO NOME). A busca constitui atividade importantíssima em muitos programas de computador, sobretudo nos de bancos de dados, em que com freqüência se necessita da recuperação de itens específicos de um arquivo. De modo geral, há dois métodos de busca: o linear e o binário. A busca linear examina os elementos numa matriz, desde seu início, até a localização do item desejado. Se os itens de dados na matriz não estiverem ordenados, a busca linear será o único método eficaz. O tempo para localização do item, usando-se a busca linear numa matriz de N itens, tem valor médio proporcional de N/2. Se houver poucos itens a pesquisar, considera-se N/2 adequado, mas, à medida que o número de itens aumentar, o tempo consumido pela execução da busca pode tornar-se excessivo.

Contudo, se soubermos que os dados no arquivo estão em determinada ordem, há um método de busca muito mais eficiente, conhecido como "busca binária".

Suponha que alguém queira encontrar no dicionário a definição da palavra "levulose". Não sabendo que o dicionário está em ordem alfabética, começará pela primeira página, para ver se a palavra se encontra ali. Não a encontrando, passa à segunda, e assim por todo o dicionário até localizá-la! No entanto, em vez disso, você (que não ignora que os termos se apresentam ordenadamente) coloca o polegar numa página próxima ao meio do volume, abre-o e verifica as palavras que ali se encontram. Se a página aberta começar pela palavra "metatarso", você constata que avançou muito; assim, a segunda metade do volume torna-se irrelevante e a palavra que se busca estará na primeira metade do livro. Então você repete o procedimento, considerando a página que começa com "metatarso" como se fosse o final do dicionário. A seguir, abre a primeira metade do dicionário na página em que está, digamos, "dodecaedro". Desta vez, você percebe que a página aberta está muito "aquém" do "l" que procura, pois essa letra se situa "além" de "d". As páginas que começam por "dodecaedro" e "metatarso" são agora consideradas a "primeira" e a "última" do volume. De novo você coloca o polegar na parte relevante e abre, por exemplo, na palavra "jasmim". Está ainda "aquém" e, desse modo, a palavra que você busca deve estar entre esta página e a de "metatarso". A repetição desse procedimento é eficaz na localização da palavra procurada — desde que ela esteja registrada no dicionário.

Nesse exemplo, a palavra "levulose" é a "chave de busca", o item que estamos tentando encontrar. Toda vez que examinamos um registro, comparamos a chave de busca com a "chave de registro" para localizar o "alvo" ou "objetivo". Esperamos encontrar, junto com a chave de registro, o que é chamado de "informação adicional". A informação adicional da chave de registro "levulose" é a definição no dicionário — neste caso, "açúcar levogiro encontrado no mel e em alguns frutos".

A analogia da busca num arquivo de banco de dados com um arquivo alvo é apropriada, desde que os registros estejam previamente ordenados como os verbetes de um dicionário. Imagine como seria difícil consultar o dicionário se os verbetes se apresentassem na ordem em que o autor os fosse coligindo ou segundo a importância que ele lhes atribuísse.

A rotina de busca necessária para nossa agenda de endereços precisa ser mais complexa do que se pensa, por razões que ficarão evidentes. A primeira operação da rotina de busca — vamos chamá-la *BUSREG*, por exemplo — é solicitar o nome a ser encontrado. Esse procedimento denomina-se "chave de busca". Suponha que haja, em algum ponto do arquivo, o registro de alguém chamado Pedro Jonas. O registro dessa pessoa tem um campo (com o nome da forma padronizada) que contém JONAS PEDRO.

A rotina de busca pode advertir-nos com uma mensagem como QUE NOME VOCÊ ESTÁ PROCURANDO?, e nós respondemos PEDRO JONAS, ou talvez P. Jonas, ou ainda Pedrinho Jonas. Antes de examinar formas mais complexas, vamos admitir que respondemos com o nome completo Pedro Jonas. A primeira coisa a ser feita pela rotina de busca será a conversão da resposta na forma padronizada: JONAS, PEDRO. A seguir, ela compara o item que fornecemos (a chave de busca) com os vários conteúdos dos campos CAMPMOD$. Se o programa utiliza uma busca linear, ele compara a chave de busca ao campo CAMPMOD$ em seqüência, até que encontre uma correspondência ou constate não existir equivalência exata.

Porém, como já observamos, no caso de os dados estarem ordenados, a busca linear não se mostra eficiente, em comparação com a binária. A rotina de busca pode garantir a ordenação dos registros a partir de uma instrução IF RMOD = 1 THEN GOSUB *CLAREG*. O programa tem a informação de que o menor

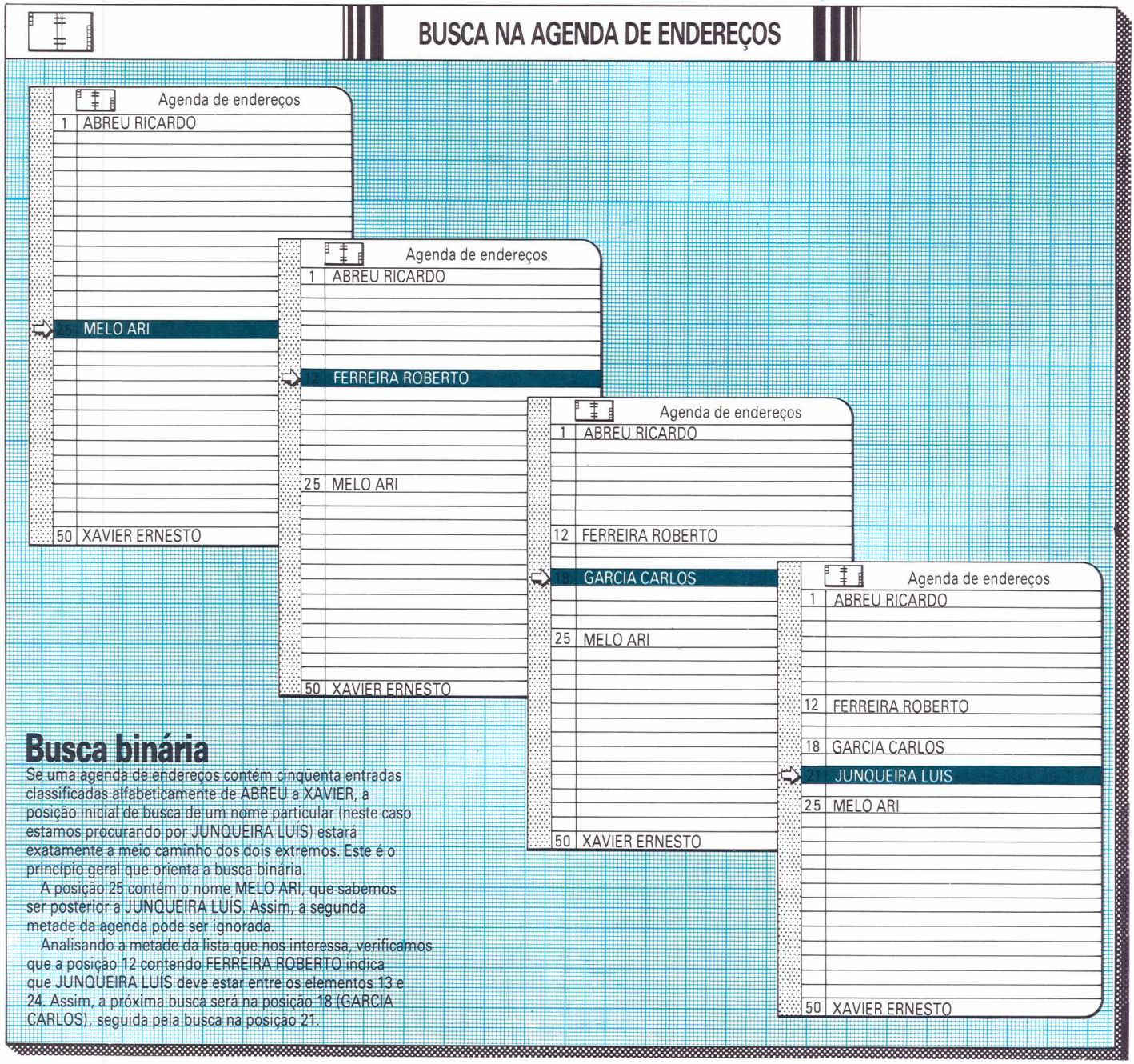
elemento na tabela em que será realizada a busca é CAMPMOD$(1) e que o maior corresponde a CAMPMOD$(TAMA − 1). Para organizar a busca, necessitamos de três variáveis: FIM, para o final da matriz (CAMPMOD$(1), no início); INI, para o início da matriz (CAMPMOD$(TAMA − 1), no início); e MEI, para o valor correspondente ao elemento do meio.

Usando a analogia do dicionário, podemos supor que FIM = MATRIZ(1) e INI = MATRIZ(TAMA − 1). Ou seja, a matriz que temos de levar em conta com relação à busca se inicia com o "menor" elemento e se encerra com o "maior". Podemos então executar os procedimentos LET FIM = 1 e LET INI = TAMA − 1. Lembre-se de que TAMA é sempre uma unidade maior que o total de registros presentes na agenda.

Caso haja 21 entradas válidas na agenda de endereços, os valores serão 22 (para TAMA), 1 (para FIM) e 21 (para INI). O valor de MEI — ou seja, a posição do elemento do meio — pode derivar em BASIC de INT ((FIM + INI)/2). Se o valor de FIM é 1, e o valor de INI, 21, o valor de MEI corresponde a 11.

Para organizar uma busca binária, precisamos de início admitir a validade de todo o arquivo e encontrar o ponto médio INT((FIM + INI)/2) no interior de um loop que se encerra — quer o objeto seja encontrado quer não haja correspondência possível. A seguir, verificamos se a chave de busca (CHAVE$) equivale ao valor MEI da matriz. Caso esse valor seja muito pequeno, inferimos que MATRIZ(MEI) corresponde à parte mais baixa da matriz a ser considerada; desse modo, podemos atribuir MEI a FIM. No entanto, revela-se mais eficiente atribuir MEI + 1 a FIM, uma vez que MATRIZ(MEI) não corresponde à chave de busca. De modo semelhante, se tivermos o procedimento IF MATRIZ(MEI) > CHAVE$, pode-se atribuir MEI − 1 a INI.

Como estágio intermediário ao desenvolvimento de rotinas completamente eficientes, o programa apresentado pode receber uma entrada fictícia (que deve estar no mesmo formato que os campos




## Busca binária

Se uma agenda de endereços contém cinqüenta entradas classificadas alfabeticamente de ABREU a XAVIER, a posição inicial de busca de um nome particular (neste caso estamos procurando por JUNQUEIRA LUIS) estará exatamente a meio caminho dos dois extremos. Este é o princípio geral que orienta a busca binária.

A posição 25 contém o nome MELO ARI, que sabemos ser posterior a JUNQUEIRA LUIS. Assim, a segunda metade da agenda pode ser ignorada.

Analisando a metade da lista que nos interessa, verificamos que a posição 12 contendo FERREIRA ROBERTO indica que JUNQUEIRA LUIS deve estar entre os elementos 13 e 24. Assim, a próxima busca será na posição 18 (GARCIA CARLOS), seguida pela busca na posição 21.

CAMPMOD$). Ele vai imprimir REGISTRO NÃO ENCONTRADO, se não houver correspondência, ou O REGISTRO É NO (MEI), se encontrar alguma correspondência. Como se inicia com o número de linha 13000, a rotina pode ser incluída no final do programa apresentado na página 399, e funcionará se a linha 4040 for alterada para IF ESCL = 1 THEN GOTO 13000.

A linha 13240 apresenta a instrução STOP, que encerra o programa por algum tempo, tão logo seja apresentada a mensagem REGISTRO NÃO ENCONTRADO ou O REGISTRO É NO (MEI). O programa se reinicia no mesmo número de linha, sem perda de dados, pela digitação de CONT. Sem a instrução STOP, o programa segue de imediato para a instrução RETURN na linha 13250, e a mensagem apresentada desaparece muito depressa, o que impossibilita sua leitura.

Consideremos em pormenores esse fragmento de programa. Na linha 13100, FIM recebe o valor 1, que corresponde à posição do menor elemento na matriz CAMPMOD$. INI recebe o valor TAMA − 1, na linha 13110. Essa é a posição na matriz CAMPMOD$ em que se localiza o elemento mais elevado. A linha 13120 inicializa um loop que se encerra quando é encontrada uma correspondência ou se verifica que não existe correspondência possível.

A linha 13130 determina o ponto médio, dividindo a soma dos índices superior e inferior da matriz (INT é utilizada para arredondar a divisão, de modo que MEI jamais possa assumir valores como 1,5). Existe a possibilidade de os conteúdos de CAMPMOD$(MEI) serem os mesmos que a chave de busca (CHAVE$). Mas se não forem, como é provável, L receberá o valor 0, garantindo que o loop seja repetido. Se o teste na linha 13140 falhar, CAMPMOD$(MEI) terá valor maior ou menor que CHAVE$. O valor de FIM será então uma unidade acima do valor anterior de MEI (linha 13150), ou o valor de INI corresponderá a uma unidade a menos que o valor anterior de MEI. O valor de MEI sozinho não é usado porque a resposta negativa do teste na linha 13140 já demonstrou que CAMPMOD$(MEI) não é o elemento que estamos procurando. Logo, não teria sentido examinar esse ponto da matriz na próxima passagem pelo loop.

Se não foi encontrada correspondência, o valor de FIM terminará por superar o valor de INI. O loop pode ser encerrado (linha 13170) e a mensagem REGISTRO NÃO ENCONTRADO será impressa (linha 13200).

Apresentamos esse fragmento de programa para demonstração e para possibilitar o teste da rotina de busca. Do modo como está, seu emprego se mostra bem limitado. Sem a instrução STOP, na linha 13240, não haveria tempo sequer para a leitura da mensagem na tela. Convém apresentar o registro completo, como foi digitado da primeira vez. Tão logo se conheça o número de registro, um procedimento simples recupera quaisquer dados adicionais necessários — CAMPMOD$, CAMPRUA$ etc. Abaixo da apresentação do registro convém que constem uma mensagem como PRESSIONAR A BARRA DE ESPAÇO PARA CONTINUAR (retornando ao menu principal) e opções complementares como PRESSIONAR "P" PARA IMPRIMIR.

Não é tão fácil decidir como lidar com a entrada de *ENCREG*. No fragmento do programa, a entrada esperada (na linha 13020) deve estar em forma padronizada — JONAS PEDRO, por exemplo. É claro que isso não é o mais adequado. Não costumamos pensar primeiro no sobrenome e depois no prenome da pessoa, e não constitui exigência razoável o fornecimento dos nomes em letras maiúsculas. Além do mais, a menor diferença em relação ao nome inicialmente fornecido resultará em mensagem de REGISTRO NÃO ENCONTRADO. *MODNOM* não basta para resolver os dois primeiros problemas. O terceiro — como lidar com uma correspondência aproximada — mostra-se bem mais interessante, mas de solução difícil.

Antes de considerar este problema, vejamos por que motivo *MODNOM* não pode resolver os dois primeiros. Se você voltar e examinar *MODNOM*, que começa na linha 10200, encontrará um bom exemplo de uma das armadilhas mais comuns em que os programadores caem — falta de generalidade. Essa sub-rotina deveria lidar com conversões de nomes ''normais'' em formas padronizadas, sempre que essa operação se fizesse necessária. Embora desenvolvida como sub-rotina isolada, ela foi escrita tendo em vista *ACRREG*. Isso faz supor que o nome a ser convertido estará sempre em CAMPNOM$(TAMA), e que, após a conversão, o campo modificado será armazenado em CAMPMOD$(TAMA). Diante desse fato, o programador tem três escolhas: refaz *MODNOM* para torná-la mais abrangente, o que por sua vez exige alterações adicionais em outras partes do programa; desenvolve uma rotina quase idêntica só para lidar com a entrada para *ENCREG* (o que significa esforço desperdiçado e exige mais espaço na memória); ou recorre a alguma técnica de programação que permita o uso da rotina *MODNOM* não modificada. Esta última alternativa é, sob certos aspectos, a menos atraente. Resolve o problema, mas a própria parte operante do programa que foi modificado fica obscura, mesmo para o programador, e se converte num pesadelo para quem tentar usá-lo.

Disso tudo se conclui que o ideal é fazer as sub-rotinas tão abrangentes quanto possível, de modo que possam ser chamadas por qualquer parte do programa.

Para exemplificar a má técnica de programação (ou programação ''suja'') e mostrar como ela pode tornar o programa obscuro, observe a linha 13020 do fragmento de programa INPUT "FORNECER A CHAVE";CHAVE$, e a seguir examine a modificação ou ''dificuldade'' que permitiria o emprego de *MODNOM*:

```
13020 INPUT "FORNECER A CHAVE";
      CAMPNOM$(TAMA)
13030 GOSUB 10200: REM SUB-ROTINA
      *MODNOM*
13040 LET CHAVE$=CAMPMOD$(TAMA)
13050...
```

O valor de TAMA tem sempre uma unidade a mais que o maior registro válido. Ou seja, não há registro na posição TAMA nas matrizes e assim esse problema não altera qualquer registro existente. Mas, sem algumas instruções REM abrangentes que expliquem

os procedimentos que estão se realizando, imagine a confusão que essas três linhas criariam para quem não estivesse vinculado ao desenvolvimento do programa.

Voltemos ao problema mais interessante de lidar com "quase erros". Suponha que demos entrada ao nome de alguém como Pedrinho Jonas na operação *ACRREG*, mas como Pedro Jonas em *ENCREG*. Ambas as formas seriam convertidas às formas padronizadas JONAS PEDRINHO e JONAS PEDRO. Portanto, não seria encontrada correspondência durante a busca. Não tentaremos resolver por completo esse problema porque a solução adequada constitui uma tarefa maior de programação. No entanto, para o leitor interessado em experimentar, eis algumas indicações:

```
INICIAR {busca de matriz para correspondência
exata}
  IF correspondência exata for encontrada
    THEN PRINT registro completo
    ELSE buscar matriz para correspondência
    aproximada
      IF correspondência aproximada for encontrada
        THEN PRINT registro da correspondência
        aproximada
        ELSE PRINT "NENHUM REGISTRO
        ENCONTRADO"
      ENDIF
  ENDIF
END
```

O procedimento para correspondência aproximada poderia ser semelhante a:

```
INICIAR {correspondência aproximada}
Buscar matriz para correspondência exata do
sobrenome
  IF correspondência exata do sobrenome
    THEN buscar prenomes para correspondência
    máxima
    PRINT registro de correspondência máxima
    ELSE buscar sobrenomes para correspondência
    máxima
      IF correspondência máxima de sobrenome for
      encontrada
        THEN PRINT registro de correspondência
        máxima
      ENDIF
  ENDIF
END
```

O procedimento de correspondência máxima pode ser definido de modo aproximativo como o encontro da variável alfanumérica objeto com o número máximo de caracteres em comum com a variável alfanumérica chave. Admita uma situação na qual a "variável chave" está inteiramente contida pela "variável objeto", ou vice-versa. Não há solução simples, apenas campo suficiente para programação criativa.

Existe um efeito paralelo no fragmento de programa apresentado. Admitamos que a seguinte seqüência de eventos venha a ocorrer. Há dez registros no arquivo de dados. Você processa o programa e a seguir utiliza *ACRREG* para incluir novo registro, seguido por *ENCREG*, para localização de registros. Quando *SAIPRG* é processado, para gravar o arquivo e encerrar o programa, o registro que você acrescentou não será gravado (embora todos os outros registros sejam). Essa é uma conseqüência direta de algo que aconteceu na execução de *ENCREG*. Você percebe por que o registro incluído não será gravado?

O próximo fascículo deste curso explica como evitar essa perda de dados; mostra com que finalidade se emprega a variável CORR e descreve como eliminar ou modificar um registro. Outras opções no menu principal (*ENCCID* etc.) são muito semelhantes às rotinas já examinadas.

Por fim, observe o que aconteceria se houvesse exatamente cinqüenta registros no arquivo de dados e fosse modificada a rotina *ENCREG*, que chama *MODNOM*. (Uma dica: TAMA valerá 51.)

## A propósito...

| Comando/Instrução/Função | Ação/Resultado | MS BASIC | Applesoft | TRS-80 | Sinclair |
|---|---|---|---|---|---|
| OUT porta, valor | Endereça um valor à porta especificada | | | + | |
| PAUSE n | Pára a execução e apresenta imagem de n quadros | | | | + |
| PEEK (n) | Fornece o byte correspondente à posição n da memória | + | + | + | + |
| PI | Fornece o valor 3,14159265... | | | | + |
| PLOT x,y | Mostra o ponto (x,y) em gráfico de baixa resolução | | + | | + |

```
13000 REM VERSAO DE *ENCREG* PARA TESTE
13010 IF RMOD=1 THEN GOSUB 11200
13020 INPUT "ENTRE COM A CHAVE";CHAVE$
13030 REM
13040 REM
13050 REM
13060 REM
13070 REM
13080 REM
13090 REM
13100 LET FIM=1
13110 LET INI=TAMA-1
13120 FOR L=1 TO 1
13130 LET MEI=INT((FIM+INI)/2)
13140 IF CAMPMOD$(MEI) <> CHAVE$ THEN L=0
13150 IF CAMPMOD$(MEI) < CHAVE$ THEN FIM=MEI+1
13160 IF CAMPMOD$(MEI) > CHAVE$ THEN INI=MEI-1
13170 IF FIM > INI THEN L=1
13180 NEXT L
13190 REM
13200 IF FIM > INI THEN PRINT "REGISTRO NAO ENCONTRADO"
13210 IF FIM <= INI THEN PRINT "O REGISTRO E O NO. ";MEI
13220 REM
13230 REM
13240 STOP
13250 RETURN
```

# Ma Bell

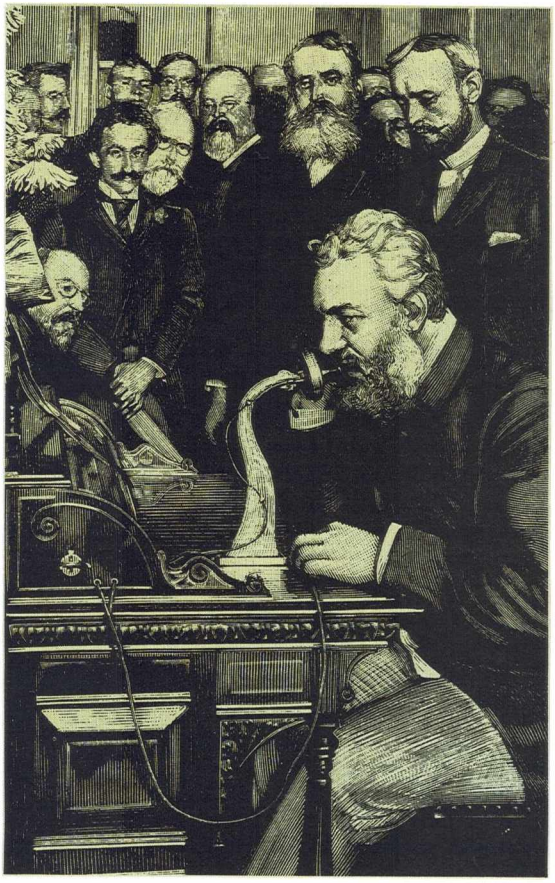
**Os Laboratórios Bell foram responsáveis por grande desenvolvimento na história do computador, tanto em hardware quanto em software.**

Dom Pedro II foi um entusiasta do telefone. No mesmo ano em que Graham Bell recebeu a patente da invenção (1876), nos Estados Unidos, um aparelho telefônico foi instalado no Palácio São Cristóvão, no Rio de Janeiro.

Graças à pesquisa e aos conseqüentes aperfeiçoamentos, o telefone evoluiu muito desde os dias desse aparelho movido a manivela. E um dos produtos paralelos de tal desenvolvimento foi o computador.

No estágio inicial da telecomunicação sonora, a American Telephone and Telegraph Company decidiu criar uma organização que pesquisasse os meios para desenvolver o sistema telefônico. Em 1925, nasceram os Laboratórios Bell — conhecidos como Ma Bell —, em Murray Hill, New Jersey.

Embora pertençam a uma empresa que visa sobretudo ao lucro, os Laboratórios Bell dedicam-se apenas à pesquisa. A empresa mantém seus cientistas afastados dos problemas rotineiramente encontrados na administração desse ramo de negócios, pois considera a pesquisa um investimento especulativo a longo prazo. Incentivam-se os cientistas talentosos a prosseguir nas pesquisas que considerem importantes porque, acredita a empresa, basta que eles tenham umas poucas idéias para que o investimento valha a pena. Os Laboratórios Bell já haviam conquistado dois prêmios Nobel até 1984, por suas descobertas em áreas diversas da pesquisa científica. Neste artigo são comentados e analisados apenas os aspectos de sua pesquisa que foram relevantes para o desenvolvimento do computador.



**Campainhas tocando**
O nome dos Laboratórios Bell foi tirado do de Alexander Graham Bell (1847-1922), reconhecido como o inventor do telefone em 1876. Dizem que as primeiras palavras transmitidas por meio de eletricidade foram pronunciadas por Bell para seu assistente, que se encontrava na sala ao lado: "Venha aqui, sr. Watson, eu preciso do senhor!"

Na década de 30 deste século, os sistemas telefônicos tornavam-se cada vez mais aperfeiçoados e automáticos. As mensagens eram enviadas sob forma analógica através de cabos telefônicos. Conectavam-se as chamadas por meio da informação contida num código de discagem digital. Primeiro, o número discado era convertido, na central telefônica, de sinal analógico numa seqüência de impulsos digitais. Estes ficavam armazenados numa memória formada por interruptores de relé até que a ligação fosse completada por um banco de interruptores de barra, que contavam os impulsos do código de discagem, convertendo-os em coordenadas num painel de controle eletromecânico. Todos os ingredientes de um computador estavam presentes.

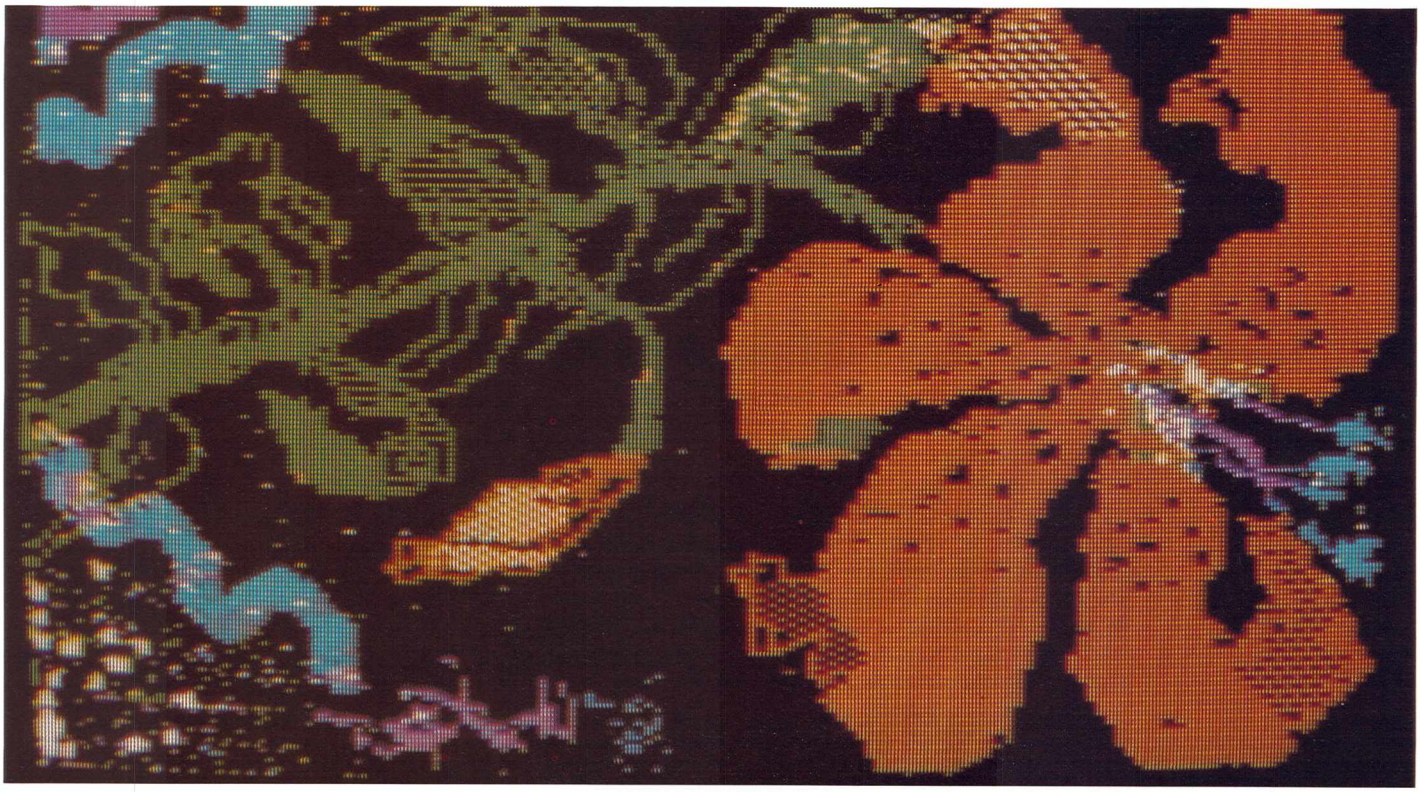
George Stibitz, matemático que trabalhava nos Laboratórios Bell, notou a semelhança entre *contar* e *somar* os impulsos. Trabalhando em casa, na mesa da cozinha, com alguns interruptores de barra e relés eletromecânicos usados, ele montou os primeiros circuitos de computador com relés.

Stibitz passou então a trabalhar com Samuel B. Williams, engenheiro que construía circuitos com interruptores havia 25 anos. Juntos criaram uma calculadora de números complexos. (Os números complexos incluem os imaginários — raízes quadradas de números negativos — e são necessários para as soluções completas das equações polinomiais.) O trabalho começou em 1937. O dispositivo consumiu 450 relés e dez interruptores de barra. Operava em notação binária e era capaz de dividir dois números de oito dígitos em 30 segundos. A calculadora de números complexos tornou-se operacional em 8 de janeiro de 1940, e em setembro do mesmo ano foi exibida à Sociedade Americana de Matemática, na Faculdade de Dartmouth, em New Hampshire (onde mais tarde foi criada a linguagem BASIC). A calculadora tinha a facilidade de acesso remoto e múltiplo por meio de teclados de máquinas de escrever ligados por fios telefônicos ao mecanismo que executava os cálculos, em Nova Iorque. As pessoas ficaram impressionadas principalmente por sua forma "humana" de operar: após receber uma questão, a calculadora parecia esperar alguns segundos antes de dar a resposta.

Muitos dispositivos secundários de hardware — como almofadas pneumáticas flutuantes usadas nas cabeças de fitas magnéticas e amplificadores de feedback negativo — também se originaram nos Laboratórios Bell. Mas sua invenção mais famosa foi o transistor, criado em 1947 por Bardeen, Brattain e Shockley (ver p. 47), que possibilitou a criação dos computadores de segunda geração.

# Linha de visão

**Computer Aided Design (projeto assistido por computador) requer cálculos complexos e saídas de alta qualidade gráfica, princípios empregados em pacotes para micros.**



**Flor de maçã**
Por mais sofisticado que seja o software, o componente fundamental de pacotes para Computer Aided Design é o próprio desenhista.
O Versawriter, cujos resultados são aqui apresentados, roda um equipamento tipo Apple II e exige duas unidades de disco e um digitalizador. Um usuário experiente levou tempo considerável para introduzir este desenho na máquina.

A idéia de empregar computadores no processo de desenho industrial surgiu no começo dos anos 60, no Instituto de Tecnologia de Massachusetts. No entanto, só na década seguinte a tecnologia permitiu que o designer visse (numa tela de monitor) a representação gráfica do trabalho do computador e interagisse com ele por meio de um digitalizador ou de uma caneta óptica, como se estivesse diante de uma prancha de desenho.

O digitalizador, a caneta óptica e o plotter são instrumentos básicos da arte do Computer Aided Designer (projetista assistido por computador). Com esses periféricos essenciais ele cria imagens do mesmo modo que os animadores "desenhando" numa prancha digitalizadora. O designer pode modificar essa imagem com uma caneta óptica, incorporando submontagens e componentes prétraçados. Produz então uma cópia do desenho já pronto num plotter. O computador torna-se um sistema delineador semelhante, em princípio, a um processador de palavras — só que trabalha basicamente com imagens, e não com um texto.

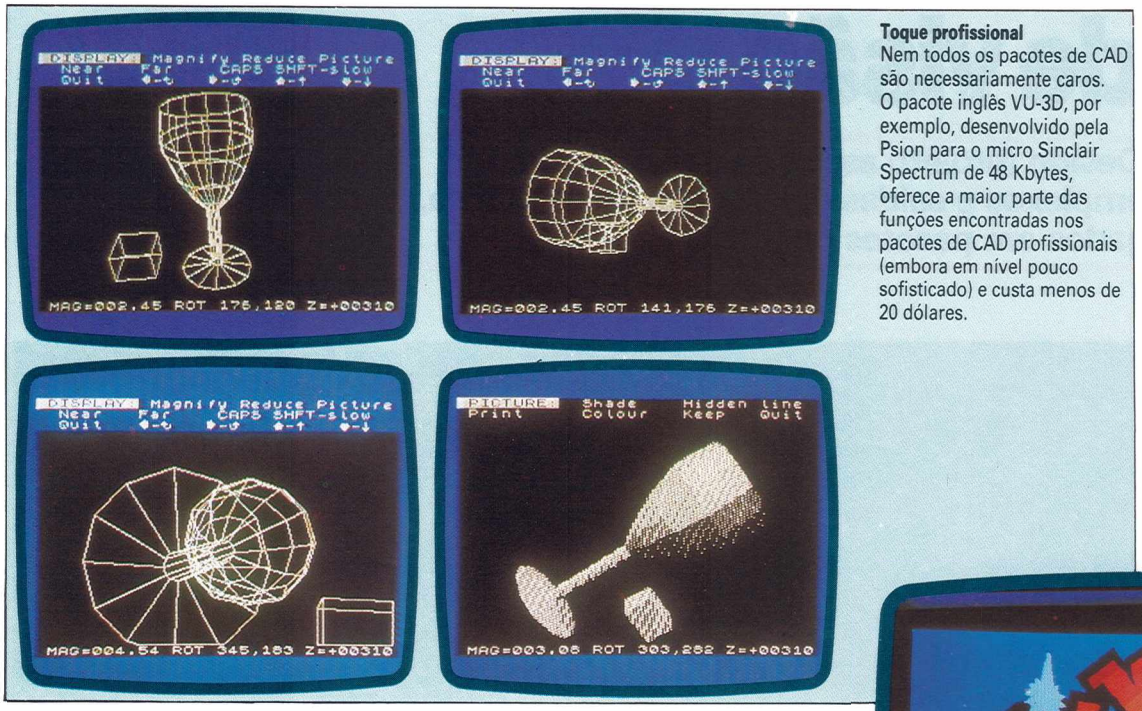
A qualidade da imagem produzida num monitor depende de sua capacidade de resolução (isto é, o tamanho de uma unidade de elemento gráfico ou pixel) e da potência e do tamanho da memória do computador que controla esse monitor. Quando examinamos imagens geradas pelo computador, deparamos com a mesma exigência — um monitor que tenha resolução de 1.000 x 1.200 pixels e um computador capaz de processar esse 1,2 milhão de elementos gráficos em menos de 1/24 de segundo.

Continuemos a analogia entre o pacote do projeto assistido por computador (Computer Aided Design) e o processador de palavras.

Em vez de mexer em parágrafos, frases ou palavras do texto — corrigindo, inserindo ou eliminando —, o programa CAD atua sobre elementos imagéticos numa página. O efeito pode ser diferente, mas o princípio é o mesmo.

## Matriz tridimensional

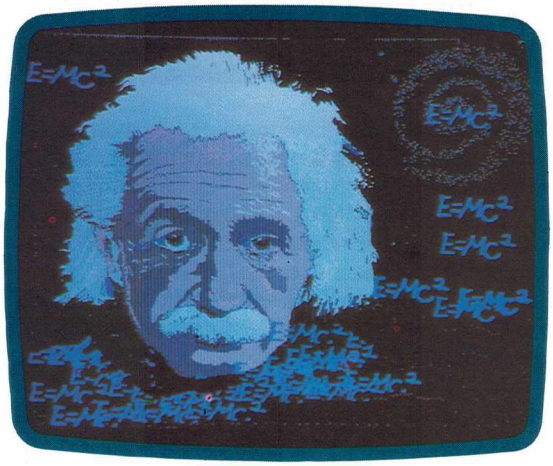
O problema, para este programa, consiste em armazenar a imagem permitindo que ela seja manipulada e/ou alterada. Se tivéssemos de fazer o modelo físico de um objeto, usaríamos o processo aditivo ou o dedutivo. O primeiro desses métodos básicos lembra a arquitetura: montamos o objeto peça por peça, até chegarmos à forma final. O método dedutivo segue o princípio usado pelo escultor, que retira matéria-prima visando ao mesmo resultado.




**Toque profissional**
Nem todos os pacotes de CAD são necessariamente caros. O pacote inglês VU-3D, por exemplo, desenvolvido pela Psion para o micro Sinclair Spectrum de 48 Kbytes, oferece a maior parte das funções encontradas nos pacotes de CAD profissionais (embora em nível pouco sofisticado) e custa menos de 20 dólares.




A analogia do computador para um bloco de material sólido é a matriz tridimensional. Em conseqüência, o tamanho e o desempenho do computador envolvido na operação tornam-se importantes. Se o arranjo permite o emprego de 1 byte inteiro para a definição de cada pixel ou elemento da imagem, a quantidade de informação que se pode reter sobre um único elemento é bem grande (256 pedaços separados, no caso de se usar um processador de 8 bits, e bem mais para dispositivos de 16 e 32 bits). Mas a criação de tanto espaço para armazenamento




**Hot dog**
Um sistema chamado Pluto, da Io Research, leva a geração de imagens de alta resolução a uma grande variedade de microcomputadores, apenas com o acréscimo de um processador mais rápido e memória suplementar.
O sistema básico oferece oito cores fixas e um poder de resolução de 670 x 576 pixels.

gera dificuldades incontornáveis. Portanto, em vez de 1 byte inteiro para cada elemento, basta alocarmos 1 bit, pois só queremos, no caso, indicar a presença ou a ausência de um elemento em determinada posição do modelo.

O software para o Computer Aided Design tem muitos atributos em comum com os pacotes de Computer Generated Image (imagem gerada por computador): ele suaviza curvas, remove elementos escondidos, sombreia, preenche os blocos, muda as cores. Para formar uma curva, basta repetir a solução de uma equação simples para uma série de valores. Se estabelecermos os pontos extremos de uma linha e a distância máxima dessa linha até a curva, resolveremos a equação. Podemos trabalhar com a solução no sentido inverso para deduzir a equação e então resolvê-la para os demais valores, formando assim a curva.

A capacidade de compor um desenho a partir de componentes básicos constitui a verdadeira força dos sistemas CAD. Já não se precisa redesenhar componentes individuais comuns — depois de definidos, são solicitados quantas vezes forem necessárias e incorporados a novos desenhos.

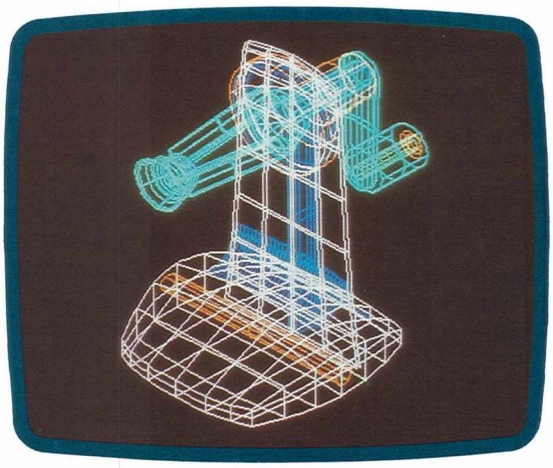
Um exemplo bem interessante é o emprego de computadores para ajudar no desenho das gerações



futuras. O desenho de placas de circuito impresso mostra-se muito complexo, envolvendo técnicas de otimização a fim de dispor os componentes e suas interconexões de maneira mais econômica (tendo em mente que as interconexões jamais se cruzam). Com freqüência o designer se vê forçado a voltar atrás, usando o método de tentativa e erro — e é aí que os pacotes de CAD são particularmente úteis. Eles armazenam todos os componentes como imagens predefinidas e podem trazê-los à tela sempre que necessário. O designer precisa apenas experimentar determinado desenho na exposição visual para verificar se ele preenche as condições necessárias. Depois coloca-o no papel como parte de um desenho em desenvolvimento. Por esse método, consegue montar um desenho e até experimentar a eficiência de várias soluções, no mesmo período de tempo que levaria para completar um esboço feito a mão.

Circuitos integrados são desenhados quase da mesma forma, mas, devido à densidade dos componentes e dos caminhos de conexão, torna-se necessário mais um serviço do software: a capacidade de ampliar parte do desenho, trabalhar nele em escala e recolocá-lo em sua posição e em seu tamanho originais no desenho. Esse efeito constitui capacidade importante do repertório CAD e aumentou de modo considerável a eficiência do sistema. A função ampliadora possibilita a especificação completa de um objeto em desenho único, e a escala se ajusta de acordo com as necessidades do observador.

Mas nem só a escala varia. Se considerarmos um objeto mais complexo — um carro, por exemplo —, encontraremos inúmeros subsistemas que se unem para formar um todo: a rede elétrica, o equipamento hidráulico, a descarga de gases, a suspensão etc. En-



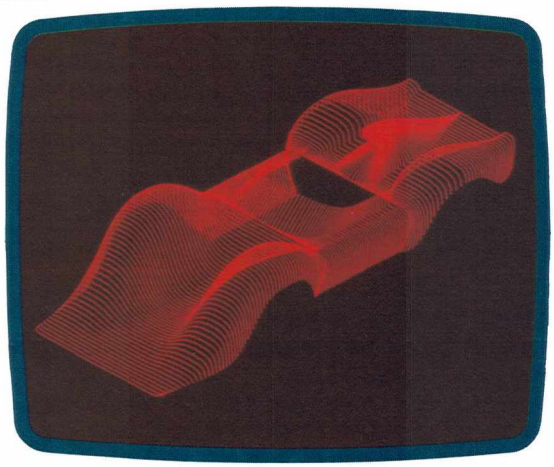
quanto o designer artístico se preocupa com a estética do pacote como um todo, os engenheiros especializados costumam se interessar mais pelos subsistemas que lhes dizem respeito. Para estes, portanto, convém manter cada subsistema numa cor diferente e então extrair do desenho completo os objetos de certa cor. Isso não significa que o desenho deva obrigatoriamente ter cores misturadas — o código pode ser retirado à vontade.

A verdadeira inovação consiste na capacidade de reter a especificação completa de um objeto (não apenas sua forma e aparência, mas também informações sobre o material de que é construído, seu peso,



o custo etc.). Manter informações sobre a forma e o tamanho do objeto é apenas uma função do sistema, também considerado um banco de dados com orientação visual. Fazendo diversas perguntas a esse banco de dados, temos condições de emitir pedidos para fornecedores, programar submontagens e manufatura de componentes; integrar linhas de programação para assegurar que os componentes cheguem onde e quando forem necessários; analisar custos; controlar a eficiência da produção etc. É tentador especular que a etapa seguinte será um sistema regular de controle direto da produção pelo computador. Com o crescente uso da automação nos processos industriais, esse próximo passo não deve ser considerado muito grande. Corresponderá a uma evolução natural.

A maioria das aplicações aqui examinadas requer o uso de computadores de grande porte ou então minicomputadores muito potentes, mas isso não significa que microcomputadores não possam desempenhar importante papel no processo de design. Existe grande variedade de software CAD disponível para máquinas que funcionam com o sistema operacional CP/M, por exemplo, e a maior parte dos fabricantes oferece pelo menos um pacote até para computadores menos sofisticados, como o Sinclair ZX81. Como observamos, o tamanho e a velocidade do computador respondem pela qualidade da imagem armazenada, mas as exigências de um usuário de micro pessoal são menores que as de um designer profissional. Portanto, é possível alcançar resultados animadores com um gasto relativamente modesto.



**Imagine só...**
O fundo desta cena de *Road to Point Reyes*, uma produção da Lucasfilms, foi composto por fractals, nova e engenhosa técnica do CAD. Os fractals são fenômenos que se tornam mais complexos à medida que nos aprofundamos em seu exame. As colinas e as montanhas ao fundo nasceram de simples polígonos, gravados na memória de um computador. Cada polígono foi ficando mais complexo com o acréscimo de sua própria forma a cada um de seus lados. O processo foi repetido randomicamente. (Abaixo, o desenvolvimento da forma de um floco de neve, a partir de um simples triângulo.)



**Armado em arame**
A primeira etapa na criação de um desenho em três dimensões recebe o nome de *wire framing* (armação de arame). A imagem é definida como uma série de coordenadas de pontos, ligadas por linhas retas. Essas linhas podem ser manipuladas usando-se algoritmos para suavização de curvas. Em seguida removem-se as linhas escondidas e preenchem-se os planos com cores e sombras, a fim de aumentar a ilusão de profundidade.

# Máquina abstrata

## A máquina de Turing é um dispositivo puramente teórico, usado para decidir se um problema é computável ou não.

Até agora, no MICROCOMPUTADOR — CURSO BÁSICO, procuramos enfatizar assuntos práticos e intens com os quais você pudesse lidar em seu microcomputador. Neste artigo, contudo, vamos examinar o lado teórico dos computadores, o campo da "ciência da computação". Ele está para a informática como a matemática pura está para a engenharia — um assunto altamente abstrato, mas do qual derivam quase todas as idéias práticas.

A máquina de Turing, por exemplo, é uma idéia teórica desenvolvida para auxiliar no estudo de algoritmos e computação. Ela constitui um computador tão elementar quanto possível. Assim, se estudiosos provarem que determinado problema não pode ser resolvido pela máquina de Turing, consideram a questão "não computável".

Turing decidiu que esse computador mínimo precisaria de três funções: um armazenamento externo para gravar e arquivar as entradas e saídas de informação; um meio de ler e gravar esse arquivo; e uma unidade de controle para determinar as atitudes a ser tomadas.

Inclui-se na constituição da máquina de Turing, portanto, uma fita — para ficar mais claro, imagine uma fita magnética — de comprimento infinito (quer dizer: qualquer que seja a quantidade de fita necessária para a solução de um problema, sempre haverá o suficiente). Ela é dividida em quadrados, que podem estar em branco ou conter símbolos. O mecanismo da cabeça da fita, que pode ler ou gravar os símbolos nos quadrados, anda junto com ela, recebendo instruções de uma unidade de controle que determina quais os símbolos a ser gravados e que direção ela deve tomar em seguida.

## Cinco quíntuplos

A unidade de controle contém um programa de execução e, nesse aspecto, pode-se considerar que a máquina de Turing foi "construída" para executar uma única aplicação, pois não há nenhuma cláusula em sua especificação para carregar ou alterar um programa. Usamos o termo "construída" entre aspas porque máquinas de Turing concretas só foram construídas com fins educativos. No entanto, é relativamente simples escrever um programa em BASIC que simule a operação de uma máquina de Turing num microcomputador.

O programa de controle de uma máquina de Turing é formado por uma coleção de "quíntuplos", ou frases compostas por cinco elementos. A decisão de qual quíntuplo deve ser executado em cada fase depende de dois fatores: o símbolo que está no quadrado embaixo da cabeça da fita, e o "estado" ou "condição" da máquina. O estado é uma qualidade arbitrária: pode-se especificar que a máquina começou no estado $S_A$, e, quando alcançar o estado especial H, ela pára (*halt*, parada), encerrando a computação. Nesse meio tempo, o estado mudará muitas vezes, de acordo com as instruções dos quíntuplos. Ele apenas reflete o que aconteceu até o momento na computação e serve para selecionar o quíntuplo que será executado em seguida. (Para ficar mais claro, imagine que seja uma variável de flag na programação BASIC.)

Os cinco elementos de cada quíntuplo são:

1) o estado da máquina;
2) o símbolo no quadrado da fita que está embaixo da cabeça;
3) o símbolo a ser inscrito no quadrado (é o mesmo que 2, se não houver qualquer mudança na informação);
4) o estado para o qual a máquina deverá ir em seguida; e
5) a direção em que a cabeça da fita deve se movimentar — para a esquerda (L) ou para a direita (R).

O quíntuplo ($S_A$, 5, 3, $S_B$, R), por exemplo, é executado sempre que a máquina estiver no estado $S_A$ e a cabeça da fita ler 5. O 5 será então substituído por 3, a máquina passará do estado $S_A$ para o estado $S_B$ e a cabeça da fita andará um quadrado para a direita (R, de *right*, direita).

O projeto de uma máquina de Turing teórica para executar determinada tarefa implica especificar o formato do input da informação que será introduzido em forma de fita na máquina, o formato do output da informação em forma de fita quando a computação estiver terminada (isto é, quando a máquina estiver no estado H) e o conjunto de quíntuplos necessários para executar o algoritmo.

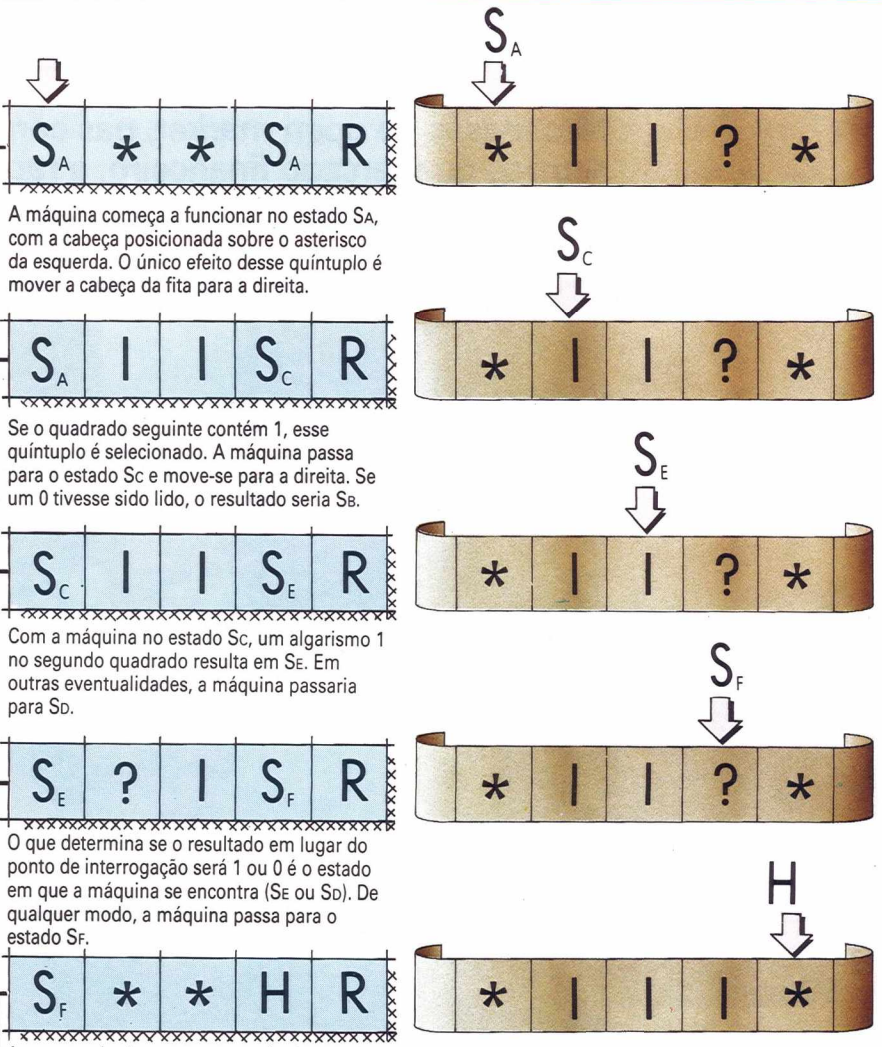
A ilustração mostra uma máquina de Turing para "desempenhar" a função AND. Colocamos os dois bits do input (1 ou 0) em quadrados adjacentes, seguidos de símbolo representado por um ponto de interrogação, que deve ser substituído pela resposta (de novo com os algarismos 1 ou 0, dependendo dos dois inputs). Por uma questão de ordem, acrescentamos um asterisco a cada ponta do segmento de informação, e ligamos a máquina no estado $S_A$, no asterisco da esquerda, terminando no da direita.

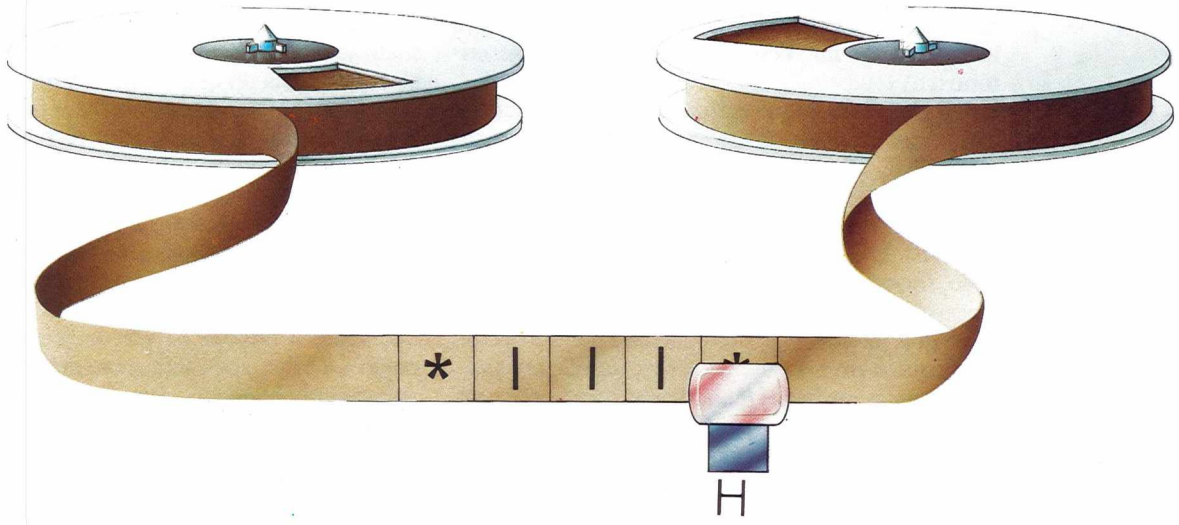
É necessário um total de dez quíntuplos para especificar essa máquina, embora, como você pode constatar no exemplo executado (1 AND 1 = 1), somente cinco são usados para qualquer execução. Se você experimentar a mesma máquina para, digamos, 0 AND 1, verá que um conjunto diferente de quíntuplos será selecionado entre os dez.

# Máquina de Turing

Ilustração da máquina de Turing para desempenhar a função AND. Os dois bits de input são colocados em quadrados adjacentes, seguidos de um ponto de interrogação que será substituído pelo resultado. Dois asteriscos são postos nas extremidades do segmento de informação para agir como delimitadores. Os dez quíntuplos abaixo especificam a operação da máquina, embora, para cada exemplo executado (neste caso, 1 AND 1), apenas cinco dos dez sejam usados.

| | | | | |
|---|---|---|---|---|
| $S_A$ | * | * | $S_A$ | R |
| $S_A$ | 0 | 0 | $S_B$ | R |
| $S_A$ | 1 | 1 | $S_C$ | R |
| $S_B$ | 0 | 0 | $S_D$ | R |
| $S_B$ | 1 | 1 | $S_D$ | R |
| $S_C$ | 0 | 0 | $S_D$ | R |
| $S_C$ | 1 | 1 | $S_E$ | R |
| $S_D$ | ? | 0 | $S_F$ | R |
| $S_E$ | ? | 1 | $S_F$ | R |
| $S_F$ | * | * | H | R |

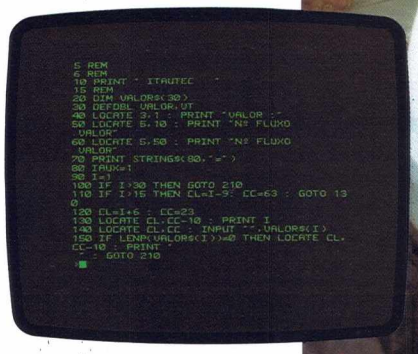


# Micro e finanças

**Nas mesas de open market, nas corretoras de valores e demais setores do mercado financeiro, a rapidez das decisões impõe a presença do microcomputador.**






**Informalização**
Corretoras de câmbio como a Patente paulista utilizam o microcomputador inclusive como um modo de tornar a informática menos formal. Os resultados conseguidos têm agradado a diretores e clientes.

Foi-se o tempo em que negócios eram fechados após conversas prolongadas. A agilidade necessária às modernas operações financeiras não pode repousar sobre formalidades que impliquem qualquer gasto de tempo: os negócios devem ser concluídos em minutos, ou uma das partes sofre grande prejuízo.

Dessa forma não foi difícil e tornou-se mesmo imprescindível para muitas instituições bancárias, financeiras, corretoras e distribuidoras de valores a instalação de microcomputadores em seus escritórios.

"Todo processo que agilize a tomada de decisões é importante por causa das características de nossos negócios", afirma Amauri Marcos Barra Ferreira, assessor da Levy Vieira Pereira Lopes e Associados Corretores de Valores e Câmbio, de São Paulo. A noção exata do papel do microcomputador no mercado financeiro pode ser avaliada imaginando-se como um operador de mesa de open atende pelo menos a dois telefonemas simultâneos e precisa decidir um negócio em minutos. No caso, papel e lápis não só tornam as respostas muito lentas como nem sempre levam a melhores opções. "O micro", assegura Ferreira, "diminui o tempo de resposta no desenvolvimento de aplicações e, dada a facilidade de sua manipulação, mostra-se muito eficaz. Esse fator desmitifica a máquina e permite o acesso a ela de maior número de pessoas."

A versatilidade do micro já lhe rendeu muitos adjetivos, na comparação com um computador de maior porte; chega a ser qualificado como "mais democrático", por Oswaldo Barbosa de Oliveira, coordenador de atendimento ao usuário do grupo de microinformática das instituições Itaú.

Roberto Luciano Farina Júnior, gerente de CPD da Patente, uma corretora de câmbio e valores mobiliários de São Paulo, resume: "O micro é a informalização da informática. É o primeiro passo para se criar a idéia do que é a informática".

A Patente começou a trabalhar com micro no desenvolvimento de um cadastro de relações pessoais da diretoria, controlando nomes, endereços e correspondência, além de projeções de quadros de ações. Em pouco tempo, a máquina já conquistava novas áreas: todos os sistemas de apoio aos setores operacionais, como open market, bolsa e câmbio. Exemplifica seu desempenho na agilização de operações a rapidez na elaboração de um cálculo como o de fluxo de caixa sobre debêntures, cujo resultado era obtido em 25 minutos com uma calculadora — ao passo que o micro faz isso em 2 minutos.

Com seu espaço conquistado, o primeiro micro implantado pela Patente divide tarefas com um minicomputador (equipamento de porte maior), operando negócios de cerca de 7.000 clientes. Na área do open, tem processado sistemas de preços de

debêntures, valorização de carteiras e cadastro de valores de ORTNs (Obrigações Reajustáveis do Tesouro Nacional).

Um segundo micro está à disposição do departamento técnico, analisando balanços e demonstrativos de cerca de seiscentas empresas nos últimos cinco anos, além do quadro de ações — estudos da relação preço/lucro das empresas, análises horizontais e verticais, taxas de retorno e outros índices financeiros. Na área de bolsa de valores, a Patente coloca um microcomputador com um sistema de controle de carteiras e ainda desenvolve programas de lançamento de opções, taxas de financiamento a termo e projetos para clubes de investimentos.

Em seu escritório no Rio de Janeiro, a corretora, com mais um micro, elabora projeções da correção monetária, que determinam o preço das ORTNs futuras, e programas para leilões de títulos federais efetuados pelo Banco Central. "Aqui na Patente", conclui Farina, "a máquina é acessível a qualquer pessoa que saiba manuseá-la."

Essa tendência é seguida também pelas instituições Itaú, com mais de 150 micros implantados, até meados de 1984, em sua administração central. O micro, na opinião dos executivos do Itaú, não tem as mesmas características de um computador de grande porte e deve, por isso, ser usado em atividades mais sofisticadas, visando ao aumento da eficácia e não à redução de custos.

Na defesa da melhor utilização desse equipamento, Oswaldo Barbosa chega mesmo a apontar algumas de suas limitações. Entre elas, destaca a incapacidade para tratar com volumes muito grandes de informação. "O micro não é um computador, é uma extensão da máquina de calcular", ousa afirmar. Mas ressalta que aí se trata de uma limitação relativa, já que o micro, por outro lado, realiza operações muito complexas que não são executadas por computadores de grande porte, devido aos custos elevados.

Nas instituições Itaú, o microcomputador auxilia na mesa de open market, dando suporte nos leilões de títulos; na área de estudos econômicos, desenvolve simulação de inflação, índices de preços, índices econômicos em geral e sua evolução. No setor de planejamento, faz avaliação de agências analisando informações não contábeis.

"Basicamente, todos os departamentos da empresa estão capacitados a desenvolver programas para sua área", diz Barbosa, "e nossa intenção não é entregar sistemas prontos, mas ensinar cada usuário a desenvolver seu próprio programa."

Na Levy Vieira Pereira Lopes e Associados, o micro instalado na diretoria é acionado para resolver problemas nas áreas de open/renda fixa, *commodities*, calculando taxas de rentabilidade para operações a termo. Isso torna possível aos clientes receber via telex, todo o dia, as diversas alterações de mercado.

Para o micro também não é problema o cálculo rápido do valor de ORTNs cambiais, para evitar a incidência de deságio em sua venda. "Uma corretora de valores", diz Amauri Ferreira, "deve ser ágil e estar atenta às modificações da legislação, às oscilações da economia, a fim de detectar as melhores oportunidades de investimento. Uma operação malfeita pode estar jogando com o patrimônio da empresa. Em alguns momentos, se as corretoras não contassem com o auxílio do micro, seria um deus-nos-acuda!"





**Eficiência e economia**
As instituições Itaú empregam microcomputadores objetivando ao aumento da eficiência e a realização menos onerosa de operações mais complexas.

A utilização da informática atende às empresas que operam no mercado financeiro agindo em duas frentes. Na primeira, resolve problemas relacionados ao crescimento do setor e ao crescimento de cada empresa. Na outra, atende a questões específicas do momento econômico, quando a rapidez é imprescindível perante o número de participantes de uma operação, as variáveis nela envolvidas e a pressão exercida sobre o fechamento das transações pela rápida variação dos índices econômicos.

O mercado financeiro aderiu ao microcomputador apostando numa crescente e lucrativa relação. E o investimento tem conseguido um retorno altamente satisfatório.

# Novilíngua

## O mundo dos computadores criou um jargão próprio, uma linguagem imaginativa constituída pelas "buzzwords".

**Toda atividade humana possui seu jargão próprio (palavras, códigos e expressões que são utilizados sobretudo por pessoas relacionadas com a área). O pessoal ligado à computação, por sua vez, emprega até mesmo um neologismo inglês para designar os termos novos, chamando-os de "buzzwords" ("palavras zumbidoras" ou "computês").**

### BUZZWORDS

O termo **BUZZWORD** surgiu pela primeira vez na década de 60, quando o departamento de publicidade da Honeywell lançou um passatempo chamado "gerador de buzzwords". A base do jogo era constituída por três colunas de dez palavras cada, numeradas de 0 a 9. Duas colunas continham substantivos e a terceira, adjetivos. O usuário devia escolher aleatoriamente um número de três algarismos, procurar as palavras correspondentes e obter uma expressão sem significado, como "módulo situacional interativo". Essas expressões poderiam depois ser usadas em conversas com colegas, deixando-os impressionados.

### BOOT

**BOOT** provém da contração de bootstrap: como na expressão "to pull oneself up by one's bootstraps" (vencer por seu próprio esforço). O carregador bootstrap ou autocarregador constitui uma rotina que se processa automaticamente toda vez que o computador é ligado. Em máquinas que não têm um sistema operacional em ROM, a rotina boot deve conter instruções que chamem o sistema a partir do disco.

### BOMBAS LÓGICAS
### CAVALO DE TRÓIA

Os crimes de computação constituem solo particularmente fértil para a produção de buzzwords. **BOMBAS LÓGICAS** e **CAVALO DE TRÓIA** são dois métodos empregados com finalidades criminosas. O primeiro descreve um conjunto de códigos, desenvolvido num programa aplicativo que permanece inativo até que seja processado um número suficiente de vezes para que o crime (por exemplo, a transferência de dinheiro de uma conta para outra) passe despercebido. O Cavalo de Tróia, a julgar pelo nome, constitui um programa disfarçado em outro, de modo a permitir acesso ao sistema.

### TURNKEY

Esse termo designa uma modalidade de contrato. Muitas organizações comerciais e empresas de consultoria de computação que instalam hardware e software possibilitam ao cliente assumir o controle do equipamento já preparado para funcionamento. É a operação **TURNKEY**, na qual tudo o que o cliente tem a fazer é girar a chave — "turn the key" — e dar a partida.

### BASIC

Essa buzzword significa "Beginners All-purpose Symbolic Instruction Code" (Código de Instruções Simbólicas de Uso Geral para Principiantes).

### HARDWARE
### SOFTWARE

**HARDWARE** e **SOFTWARE** são buzzwords: *hard* significa "duro" e é empregado como "tangível", e *soft*, o oposto. Há mais dois outros tipos de "ware": **FIRMWARE** significa que o software está protegido por hardware (como na ROM e na EPROM) e **PEOPLEWARE** refere-se a todas as pessoas que trabalham com computadores.

### BAUD

A velocidade em que os dados são transmitidos recebeu o nome de **BAUD** em homenagem a Émile Baudot, inventor de um código telegráfico, que rivalizou com o de Samuel Morse.

## BYTE

Até o aparecimento do microprocessador de 8 bits, 1 byte correspondia a bits suficientes para codificar um caractere individual — ora 6, ora 8. Nessa época, os computadores raramente utilizavam palavras com menos de 24 bits; e alguns equipamentos, sobretudo os projetados com fins científicos, chegavam a 64 bits. A grafia excepcional de byte (próxima da do termo inglês *bite*, "mordida") levou à criação do termo **NYBBLE** (que lembra a grafia do termo inglês *nibble*, "mordiscada"). Nybble corresponde à metade de 1 byte.

## BIT

Embora a maioria dos dicionários informe que **BIT** corresponde a uma contração de BInary digiT — dígito binário —, é possível que essa buzzword provenha de "pequeno pedaço". Na gíria americana o termo bit corresponde à oitava parte de 1 dólar e é sempre precedido de "dois": "2 bits", por exemplo, correspondem a um quarto de dólar.

Bit freqüentemente aparece como um prefixo. É o caso de *bit-slicing*, termo utilizado para explicar como certos microprocessadores podem ser montados a partir de "unidades menores" com capacidade para 2, 4 ou 8 bits, que resultam em dispositivos com capacidade de até 32 bits. Estudiosos da computação especulam que programas não utilizados por muito tempo desenvolvem a "deterioração bit": geram-se *bugs* maiores e insolúveis.

## BOMBA-RELÓGIO

O termo **TIME BOMB (BOMBA-RELÓGIO)** refere-se a uma engenhosa técnica para proteção de software comercial contra pirataria. É um conjunto de códigos no interior do pacote que seria desativado quando o sistema fosse instalado por um vendedor autorizado. Na cópia pirateada, contudo, a Bomba-Relógio espera algum tempo — em geral, até que o usuário esteja bastante dependente do pacote. No dia seguinte à "explosão", os arquivos do usuário estarão inutilizados e a cópia do programa, destruída (a menos que o disco tenha sido protegido contra gravação superposta).

## GARBAGE

É um termo ("sucata", em inglês) que aparece em várias expressões de glossários de jargões para usuários de computador. A sigla **GIGO**, por exemplo, representa "Garbage In, Garbage Out" ("Sucata Entrando, Sucata Saindo"). Isso serve para lembrar que computadores apenas processam dados e, desse modo, você não pode esperar resultados confiáveis se não fornecer dados corretos.

### GARBAGE COLLECTION

É o nome dado a um processo interno que pode ser utilizado em seu microcomputador, se este usar uma versão do BASIC que admita variáveis alfanuméricas dinâmicas (isto é, que podem mudar de comprimento durante o programa). Cada vez que uma dessas variáveis aumenta de comprimento, faz-se nova cópia completa em RAM. Assim, se houver muitas instruções na forma `LET A$=A$+"*"` (sobretudo no interior de loops), não demorará muito tempo para que a memória seja completamente preenchida. Nesse ponto, a execução do programa se interrompe por algum tempo e uma rotina em ROM, denominada "coletor de sucata", limpa a área das variáveis alfanuméricas e elimina todas as partes dessas variáveis que foram deixadas de lado em manipulações anteriores. Embora o programa passe a ser muito mais eficiente após a "limpeza" do coletor de sucata, o processo pode demorar vários segundos ou até minutos, durante os quais o computador interrompe o processamento do programa.

## HANDSHAKE

Muitas buzzwords originam-se de analogias. Quando se chega a um acordo comercial, os participantes de fato podem dar as mãos: assim, em termos de computação, **HANDSHAKE** ("aperto de mão") é o nome dado ao sinal eletrônico que significa que uma troca de dados se completou.

| | | |
|---|---|---|
| 0. Integrado/a | 0. de Banco de Dados | 0. Rede |
| 1. Interativo/a | 1. Situacional | 1. Capacidade |
| 2. de Memória | 2. Randômico/a | 2. Sistema |
| 3. Digitado/a | 3. de Diagnóstico | 3. Algoritmo |
| 4. Estocástico/a | 4. de Endereçamento | 4. Processador |
| 5. Periférico/a | 5. Linear | 5. Matriz |
| 6. Heurístico/a | 6. Gráfico/a | 6. Módulo |
| 7. Relacional | 7. Alfanumérico/a | 7. Recurso |
| 8. Homeostático/a | 8. de Imagem | 8. Hierarquia |
| 9. Programável | 9. Esquemático/a | 9. Gerador |

**Capacidade randômica periférica**

Você pode criar seu gerador de buzzwords com três colunas de dez palavras cada, como fizemos aqui. A escolha de um número aleatório de três dígitos origina uma expressão aparentemente tecnológica, impressionante e sem sentido, como a do título.

# PC16

## Seu teclado eletrônico e ergonométrico pode ser operado à distância de até 1,20 m.

O PC16 é um microcomputador profissional totalmente compatível com o IBM Personal Computer (IBM PC), tanto em hardware quanto em software, comercializado pela Dismac.

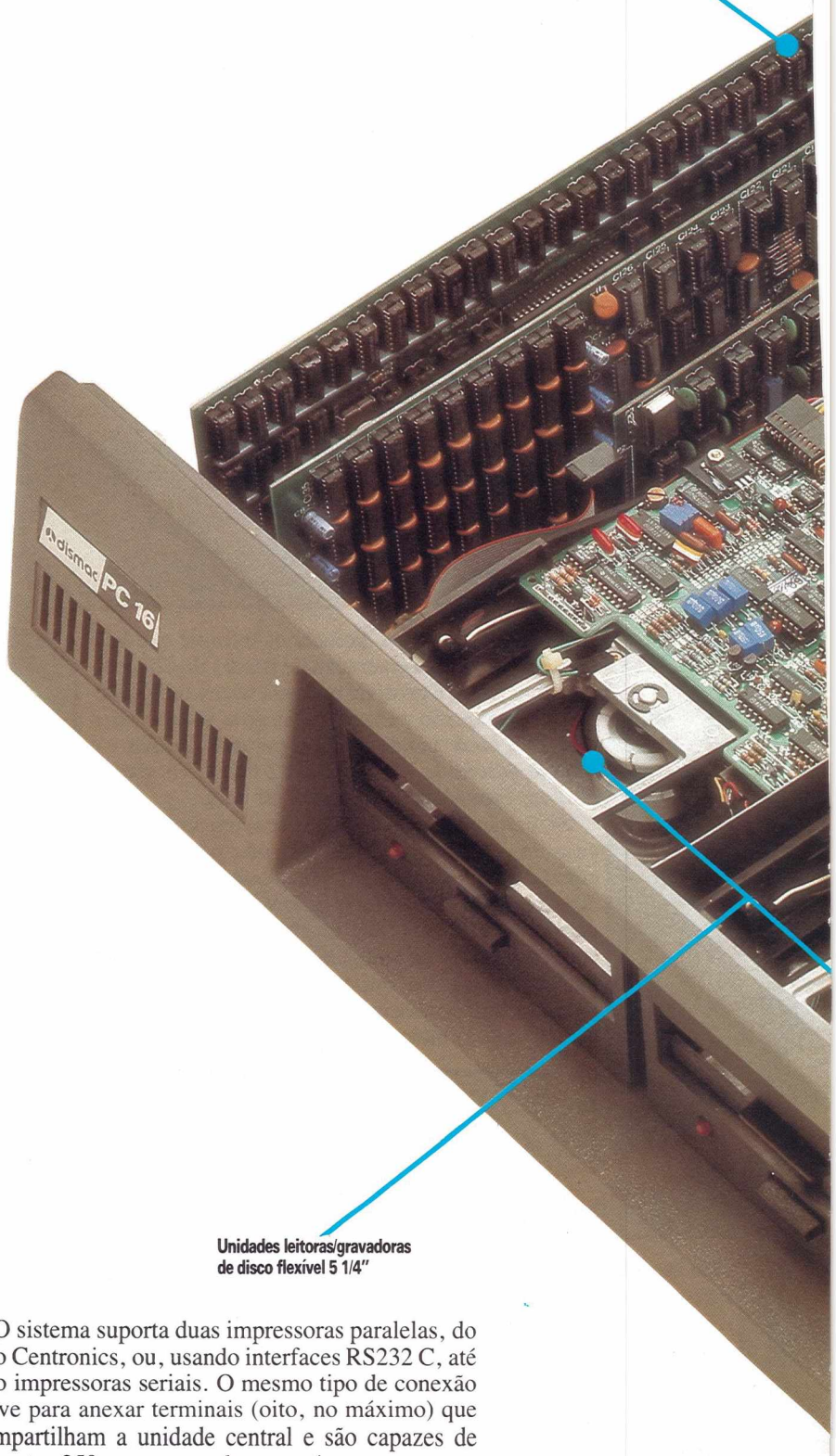
A Unidade Central de Processamento desse equipamento é o microprocessador Intel 8088. Sua principal característica é transferir informações em dois blocos de 8 bits e operá-las internamente em 16 bits. Com isso, todo o processo fica bem mais rápido do que em outros micros disponíveis no mercado. A RAM do PC16 é de 64 Kbytes, expansível por módulos até 256 Kbytes ou por placas adicionais de até 512 Kbytes. A EPROM é de 48 Kbytes e contém programas de autoteste do hardware, útil para evitar erros de programação.

O monitor, com tela de fósforo verde de 12 polegadas, opera em modo alfanumérico com 25 linhas x 80 colunas e em modo gráfico com 100 x 160 pontos, 200 x 320 pontos e 200 x 610 pontos, com excelente resolução de cores (oito cores e dezesseis tonalidades).

O teclado do PC16, eletrônico e ergonométrico, é semelhante ao de uma máquina de escrever. Com 85 teclas, dispõe de letras maiúsculas e minúsculas, e bloco numérico reduzido, separado. O teclado, destacado do gabinete, pode ser operado à distância de até 1,20 m, por meio de um cabo flexível.

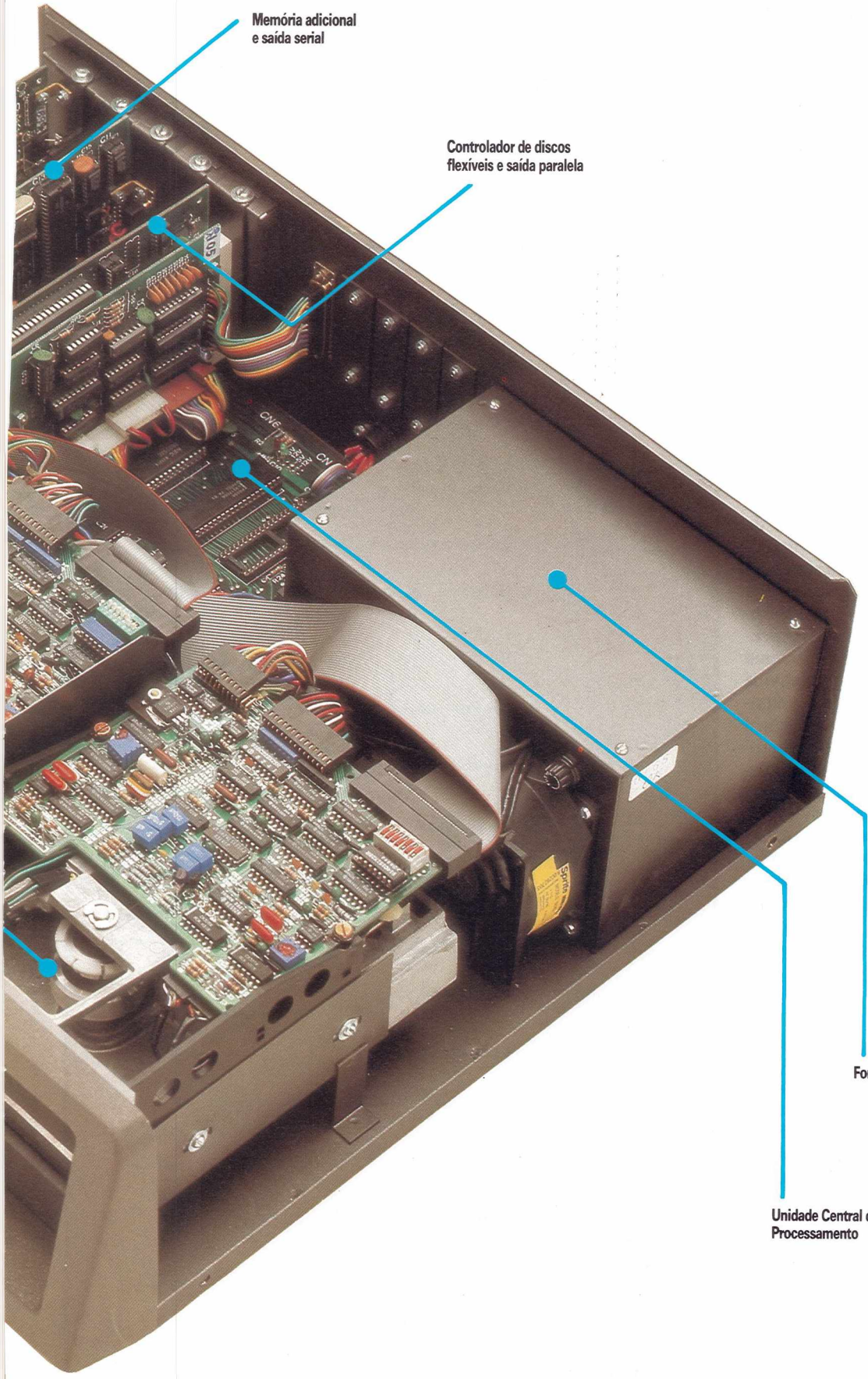
O equipamento opera até quatro unidades de discos flexíveis: duas no interior do gabinete do processador e duas em módulo separado. Se preferir, o usuário pode conectar ao aparelho quatro unidades de discos rígidos de 5 ou 10 Mbytes do tipo Winchester.




O sistema suporta duas impressoras paralelas, do tipo Centronics, ou, usando interfaces RS232 C, até oito impressoras seriais. O mesmo tipo de conexão serve para anexar terminais (oito, no máximo) que compartilham a unidade central e são capazes de executar 250 programas de uma só vez.

Utilizando-se uma interface serial RS232 C, a máquina pode ser conectada a equipamentos de grande porte, servindo como terminal inteligente, ou a outros microcomputadores para transferência do arquivo e simples intercomunicação.




## PC16

**MICROPROCESSADORES**

Intel 8088 — Processador central.
Intel 8087 — Co-processador aritmético de ponto flutuante.

**CLOCK**

5 MHz

**VÍDEO**

Monitor de 12", em fósforo verde, com capacidade multinível (múltiplas tonalidades). Interface gráfica — modo alfanumérico (80 x 25) e gráfico (100 x 160, 200 x 320 e 200 x 610).

**MEMÓRIA**

48 Kbytes de EPROM e 64 Kbytes de RAM (expansíveis até 512 Kbytes).

**TECLADO**

Tipo máquina de escrever, com 85 teclas e teclado numérico separado.

**LINGUAGENS**

BASIC, FORTRAN, COBOL, PASCAL

**PERIFÉRICOS**

Unidades de disco (5 1/4" e Winchester), impressora, interface serial RS232 C.

**DOCUMENTAÇÃO**

A máquina é acompanhada de completo manual de instalação e operação.

# Elementos subversivos

## O planejamento cuidadoso e a abordagem passo a passo reduzem o tempo gasto na eliminação de erros do programa.

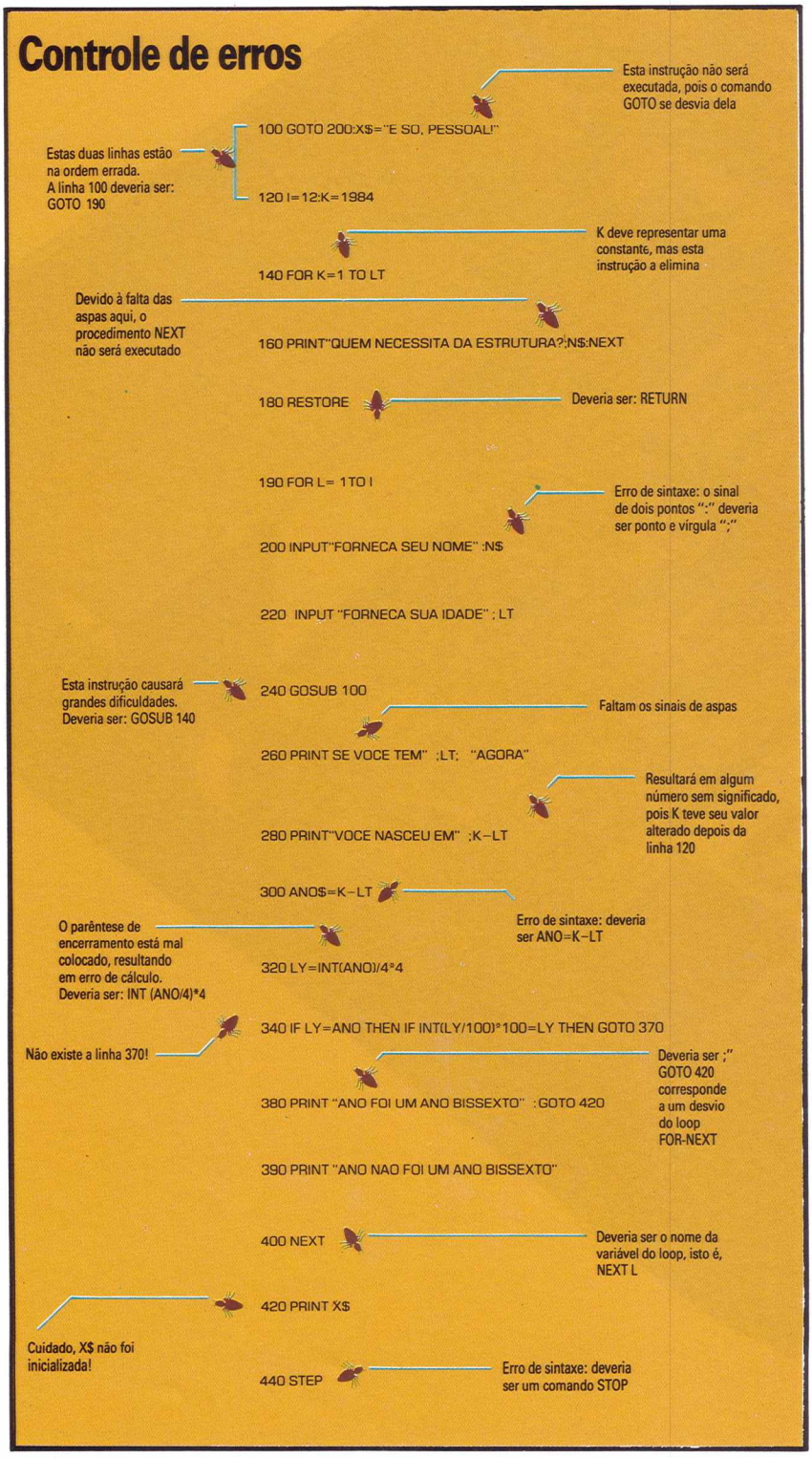
Conforme adquire maior habilidade no desenvolvimento de programas, você tende a ficar mais apto a desembaraçá-los de erros ("debugá-los"). Os enganos lógicos e de sintaxe, que mesmo os programadores mais experientes cometem, tornam-se menos freqüentes ou problemáticos à medida que sua experiência aumenta. Os recursos aqui expostos podem ajudá-lo a evitar erros e tornar-se mais eficiente na eliminação de eventuais incorreções.

O programa começa na cabeça do usuário. Logo, se sua estrutura for pensada de modo superficial, o desenvolvimento se fará com grande quantidade de erros. De início, portanto, convém colocar o problema de modo tão claro quanto possível para si mesmo ou para outra pessoa. A seguir, o problema deve ser dividido em unidades logicamente completas — input, output, algoritmos, estruturas de dados, procedimentos etc. Cada parte deve ser examinada como um aspecto individual. Se necessário, subdivida cada um desses aspectos nas unidades que o compõem e torne a dividi-los até que o problema original se transforme num conjunto estruturado de subproblemas programáveis com facilidade. Uma abordagem formal, como o emprego de pseudolinguagem ou diagrama de fluxo, mostra-se essencial no estágio de estruturação como forma de preservar a estrutura global do programa e mantê-lo sob controle. Você deve ficar longe do teclado até ter certeza de que sabe como programar cada unidade do problema. Essa abordagem é denominada top-down, um método que pode condensar bem o tempo necessário para eliminação de erros.

## A reinvenção da roda

A subdivisão de problemas em tarefas de resolução mais fácil leva você a desenvolver programas que constituem conjuntos de sub-rotinas ou procedimentos vinculados por um programa básico principal. Isso torna mais fácil a detecção de erros e possibilita a construção de um arquivo de sub-rotinas de eliminação de erros para emprego em programas futuros. A alternativa é denominada "reinvenção da roda": cada vez que desenvolve, por exemplo, um programa que classifica dados, você resolve de novo o problema de como elaborar uma rotina de classificação. E é bem provável que volte a incorrer em erros cometidos antes. Mas é simples desenvolver o programa, isento de erros, gravá-lo e chamá-lo, sempre que for necessário.

Tente utilizar nomes apropriados de variáveis, mesmo quando tiver de empregá-los sob forma abreviada. LÍQUIDO=BRUTO-IMPOSTOS, por exemplo, indica com clareza seu significado; e LQ=BR-IMP não é mau substituto; porém, L=B-I revela-se uma expressão muito ambígua e não indica as variáveis envolvidas. Constitui bom procedimento manter uma tabela que mostre todas as variáveis utilizadas no programa, bem como sua finalidade. Isso o levará a padronizar seu emprego (como o uso freqüente de certas variáveis com uma única letra para os contadores de loop) e evita a utilização da mesma variável para diferentes finalidades. De



modo semelhante, convém armazenar os valores freqüentes nas variáveis no começo do programa, remetendo-se depois a elas. Isso torna o programa mais rápido e organizado, e você pode alterar os valores sem ter de procurar por todo o programa.

Mesmo com essa abordagem formal, é difícil eliminar por completo os erros. Assim, convém adotar um método organizado de encontrá-los e suprimi-los.

O tipo mais comum de erros ocorre com a sintaxe. Em geral, você pode corrigi-los tão logo os encontre. Observe:

```
10 PRINT"GRANDES ERROS POSSUEM
   PEQUENOS"
20 PRINT"ERROS PARA ATRAPALHA-LOS"
```

Essas linhas quase sempre ocasionam mensagens de erro, caso não sejam digitadas como duas linhas separadas. A linha 10 tem quarenta caracteres; desse modo, quando você a digita numa tela de 40 colunas, o cursor pára no início da linha seguinte, o que pode fazer você esquecer de teclar RETURN na linha 10, antes de iniciar a digitação da linha 20. Caso isso aconteça, o que parece ser um conjunto de linhas perfeitas em seu programa será, de fato, uma linha com um erro de sintaxe (o número 20) no meio dela. Um modo de detectar esses erros consiste em listar linhas suspeitas uma por vez, e não como partes de um programa.

As mensagens de erro podem se mostrar enganadoras. Observe este exemplo:

```
25 DATA 10.2,34,56,9,0.008,15.6
30 FOR K=1 TO 5: READ N(K):NEXT K
```

Esse programa pode não realizar sua tarefa em razão de suposto erro de sintaxe na linha 30, quando efetivamente o erro está nos dados da linha 25 (um dos caracteres zero foi erroneamente digitado como a letra O).

Instruções com erros que não prejudicam a sintaxe são a falha mais comum e, em geral, a mais difícil de se detectar. Nesse caso, o método torna-se fundamental. Comece pela tentativa de encontrar de modo aproximativo o erro do programa. Isso é relativamente simples com programas modulares bem estruturados e torna-se ainda mais fácil pelo emprego do recurso TRACE, que ocasiona a impressão da linha atual do programa na tela, à medida que é executado. Se sua máquina não possibilita esse procedimento, crie instruções TRACE a intervalos regulares por todo o programa (PRINT "LINHA 150", no início da linha 150, por exemplo). Empregue também o comando STOP para interromper a execução do programa em seus pontos significativos, de modo que você possa examinar os valores das variáveis mais importantes. Realize isso de modo direto pelo emprego de PRINT, ou desenvolva uma sub-rotina no final do programa:

```
11000 REM IMPRIMIR AS VARIAVEIS
11100 PRINT"RESULTADO,TAMANHO,FLAGS"
11200 PRINT RS;TM;F1;F2
11300 PRINT"MATRIZ DO PAINEL"
11400 FOR K= TO 10:PRINT PN$(K):NEXT K
```

Por conseqüência, quando o programa encontra um comando STOP, você pode digitar GOTO 11000 e apresentar o estado atual das variáveis. Convém até mesmo alterá-las (pela digitação, digamos, de TM=17 e o pressionamento de RETURN) e a seguir reiniciar o programa com o comando CONTinuar.

Quando tiver descoberto que o erro está num conjunto de linhas ou em determinada variável, você estará próximo de eliminá-lo. Aja, porém, com cuidado. Tente uma solução por vez, para verificar seu exato efeito na execução do programa. É muito fácil realizar várias alterações no intervalo dos processamentos, eliminando um erro mas criando outros, e depois esquecer o que você fez.

Os loops e desvios, sobretudo quando internos, são solos férteis para erros e exigem atenção especial tanto no escrever quanto no eliminar incorreções. Observe este trecho de programa:

```
460 IF SM< 0 AND SC < > -1 THEN IF SC>0 OR
    SM=SC-F9 THEN LT=500
470 FOR C1=1 TO LT:FOR C2=LT TO C1 STEP-1
480 SC=SM+SC*C2
490 NEXT C2:SM=0:NEXT C1
```

O que significa todo esse procedimento? Mesmo que você saiba o que ele deve realizar, pode avaliar sua eficácia? A inserção de instruções no interior de loops, quando deveriam estar fora deles, implica quase sempre criação de erros. O mesmo ocorre quando não se cobrem todas as condições possíveis, desenvolvendo instruções do tipo IF-THEN. Um caso especial desse tipo de erro acontece quando você escreve instruções múltiplas, após IF-THEN. Por exemplo:

```
655 IF A$= " " THEN GOTO 980:A$=B$
660 PRINT A$
```

A instrução A$=B$ não será executada porque, ou A$=" ", em que o controle passa para a linha 980, ou A$<>" ", em que o restante da linha 655 é ignorado.

Na eliminação de erros, a experiência é a melhor mestra, mas um procedimento passo a passo e um método disciplinado mostram-se auxiliares inestimáveis. Vá com calma e — acima de tudo — *não entre em pânico*.



## Erro de principiante

Para o iniciante em programação, os erros parecem reagir instintivamente, como animais, ocultando-se. O curioso é que um erro histórico de fato resultou da ação de um inseto — "bug", em inglês que também é uma buzzword para designar "erro". Ao tentar eliminar o erro do programa que estava desenvolvendo no equipamento Harvard Mark II, em 1945, Grace Hopper descobriu que uma mariposa havia se enroscado na aparelhagem eletromecânica do computador e estava ocasionando o problema.

# Show de laser

## A tecnologia de discos ópticos (a laser) descobre aplicações muito importantes para os microcomputadores.

Quando se conversa sobre microcomputadores, a primeira característica citada quase sempre se refere ao tamanho da memória. De fato, a capacidade interna de armazenamento do computador é importante, mas a limitação de seu sistema de armazenamento em massa tende a se tornar mais crítica a longo prazo.

Após alguns meses, o usuário entusiasmado por microcomputador terá acumulado considerável número de fitas cassete ou várias caixas de discos. Contudo, muitos desses programas jamais serão modificados. Seria melhor, portanto, que eles ficassem armazenados em cartuchos de ROM e não em delicados meios magnéticos. Uma forma de sistema de armazenamento digital utilizável para leitura, como um cartucho, mas que tivesse capacidade maior, seria bem proveitosa.

Esse sistema existe — na forma de disco óptico a laser. Mas ainda é usado, num ambiente doméstico, apenas como alternativa para o gravador de videocassete, na exibição de programas ou filmes prégravados. Outro uso da mesma tecnologia é o disco de áudio compacto, que vem substituindo o prato giratório e o formato da agulha de vitrola dos sistemas de alta-fidelidade.

A diferença entre os dois sistemas (além dos diâmetros de seus discos) reside nos métodos operacionais. Enquanto o disco de vídeo é um sistema analógico, o disco de áudio compacto armazena informação sob forma digital, isto é, como uma seqüência de algarismos 1 e 0. Essa informação reverte para o sinal de áudio original por um conversor digital-analógico, que é o oposto eletrônico do processo que criou originalmente a informação. Como há muitos campos elétricos no ambiente doméstico, o uso de meios magnéticos, como discos flexíveis, revela-se contraproducente para gravações de vídeo. De qualquer maneira, a quantidade de informação de um disco óptico a laser pode chegar a milhares de megabytes, o que é muito mais do que um disco — mesmo um Winchester — pode conter.

Existem vários sistemas de discos ópticos a laser no mercado internacional. O que teve maior aceitação (lançado pela Philips) emprega um disco plástico de 35 cm de diâmetro, que na realidade é apenas um "envelope" protetor. A informação está gravada no interior do plástico, na forma de uma série de sulcos (de 0,5 micrômetro de largura por 0,1 micrômetro de profundidade) produzidos numa chapa metálica. Como no disco flexível, cataloga-se a informação armazenada no disco de vídeo de forma que, com o tipo certo de toca-discos, é possível localizar instantaneamente qualquer parte da informação. Com a cabeça de leitura no local desejado, o raio laser lê a informação no disco. A luz passa através do plástico e incide na superfície da chapa metálica. Uma célula fotossensível capta a informação à medida que a luz se reflete nos sulcos da chapa. A informação é gravada numa trilha espiralada, com um quadro do vídeo a cada volta. Isso fornece o total de 54.000 quadros em cada lado do disco, ou 36 minutos de tempo de execução.

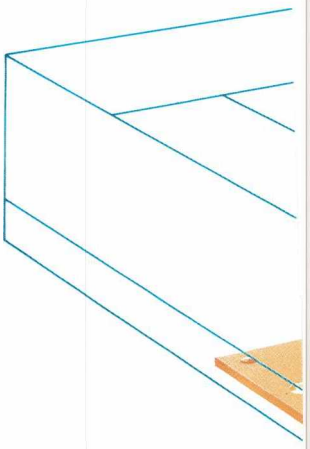
No campo da computação, os principais usos potenciais para discos ópticos dividem-se em dois.

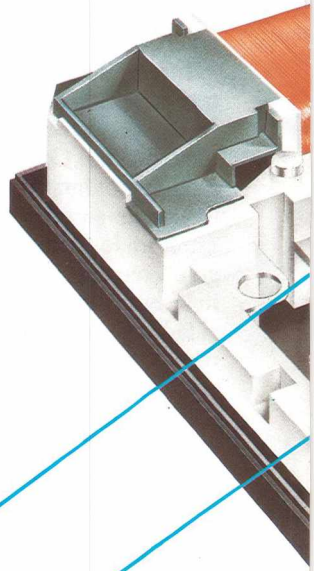
A primeira área de desenvolvimento é a do vídeo interativo, já disponível. Um programa transmitido pela televisão não é interativo — o espectador não interage com o veículo, não controla a ordem em que as cenas são apresentadas. Com o vídeo interativo, no entanto, as informações textual e visual são armazenadas num disco de vídeo, conectado a um computador. O disco pode ser usado como uma biblioteca para consulta e o texto exibido sobrepõe-se às imagens do vídeo numa tela de televisão convencional.

Em resposta a sugestões do computador, o usuário pode selecionar "trilhas" ou "cenas" específicas no disco de vídeo tocado. O disco é usado também como dispositivo de auxílio para treino com cenas (em movimento ou fixas) em exibição no vídeo de um televisor e com as respostas do *trainee* a questões importantes sendo colocadas no computador. Este pode controlar o desempenho do usuário e informar sobre ele. Como as interfaces de um disco de vídeo doméstico para microcomputador ainda não estão sendo distribuídas em larga escala, muitas pessoas entusiasmadas construíram as suas por conta própria. No entanto, a Philips comercializou um modelo de seu LaserVision, que pode ser ligado diretamente a um vídeo interativo e, por meio de uma interface IEEE488 ou RS232, a um computador.

Outra área em que a tecnologia do disco óptico tende a ser explorada é no fornecimento de software para computadores. Imagine, por exemplo, as vantagens de prover um computador com software para todos os seus sistemas — processador de palavras, banco de dados, folha eletrônica e várias dezenas de jogos — num único disco incorruptível.

Ele tende a tomar a forma do disco de áudio compacto, mas até 1984 nenhum toca-discos compacto fora equipado com interface para computador. Com um mercado potencial tão amplo, esperava-se que toca-discos compactos domésticos logo adotassem essas interfaces, e que se fizessem toca-discos compactos adaptáveis a computadores de uso pessoal.

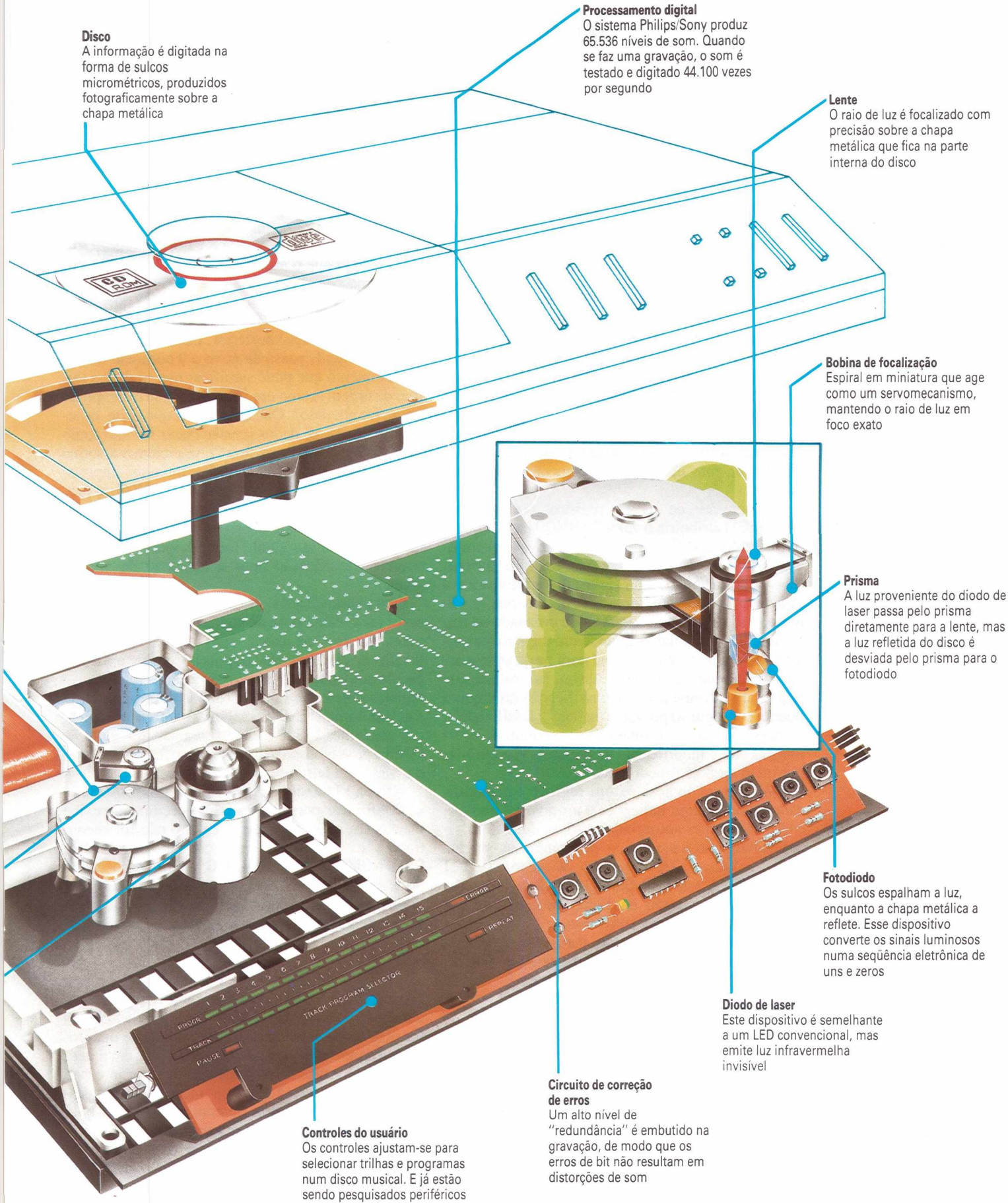




**Motor linear**
O servomecanismo para movimentar o braço sobre o disco é uma simples "espiral" que funciona com uma mola leve. O arranjo assemelha-se muito ao encontrado em medidores feitos com bobina móvel, como os medidores de corrente ou voltagem

**Braço**
Preso no centro por um eixo, o braço pode girar livremente em torno dele e é balanceado com precisão. Em conseqüência, a cabeça de leitura traça um arco sobre o disco

**Motor**
Um circuito de feedback controla com extrema precisão a velocidade de rotação do disco. À medida que o braço percorre o disco, de dentro para fora, a velocidade muda de 500 para 200 rpm




**Disco**
A informação é digitada na forma de sulcos micrométricos, produzidos fotograficamente sobre a chapa metálica

**Processamento digital**
O sistema Philips/Sony produz 65.536 níveis de som. Quando se faz uma gravação, o som é testado e digitado 44.100 vezes por segundo

**Lente**
O raio de luz é focalizado com precisão sobre a chapa metálica que fica na parte interna do disco

**Bobina de focalização**
Espiral em miniatura que age como um servomecanismo, mantendo o raio de luz em foco exato

**Prisma**
A luz proveniente do diodo de laser passa pelo prisma diretamente para a lente, mas a luz refletida do disco é desviada pelo prisma para o fotodiodo

**Fotodiodo**
Os sulcos espalham a luz, enquanto a chapa metálica a reflete. Esse dispositivo converte os sinais luminosos numa seqüência eletrônica de uns e zeros

**Diodo de laser**
Este dispositivo é semelhante a um LED convencional, mas emite luz infravermelha invisível

**Circuito de correção de erros**
Um alto nível de "redundância" é embutido na gravação, de modo que os erros de bit não resultam em distorções de som

**Controles do usuário**
Os controles ajustam-se para selecionar trilhas e programas num disco musical. E já estão sendo pesquisados periféricos para computadores com o uso de tecnologia DC (disco compacto)

# Recursos extras

## A eliminação das anomalias ocasionadas pela unificação de módulos e o acréscimo de alguns recursos continuam nossa agenda de endereços.

No fascículo anterior deste curso, os leitores ficaram com a tarefa de imaginar o motivo pelo qual o processamento de um programa de agenda de endereços, com o acréscimo de um registro (empregando *ACRREG*), a localização de outro (*ENCREG*) e ainda o desvio do programa (*SAIPRG*), resultaria na não gravação do registro incluído. O problema surgiu a partir do uso da variável RMOD como flag indicativa da modificação de um registro (significando que o arquivo poderia estar fora de ordem). A sub-rotina *CLAREG* colocaria o arquivo em ordem alfabética e a seguir atribuiria o valor 0 a RMOD, supondo-se o arquivo ordenado. A execução de *SAIPRG* verificava se o arquivo estava em ordem (RMOD=0), mas não previa sua gravação, em caso positivo.

A inclusão de novo registro (empregando *ACRREG*) atribuiria o valor 1 a RMOD (se o arquivo tivesse sido modificado, isto é, um novo registro tivesse sido acrescentado). Porém, *CLAREG* atribuiria o valor 0 a RMOD, indicando que o arquivo estava ordenado. No entanto, tornam-se necessárias — não importando se o arquivo tenha sido ordenado ou não — uma flag que sinalize quando um registro for modificado e uma flag individual para mostrar se o arquivo está ordenado ou não. Assim, sub-rotinas que requeiram informe sobre a classificação do arquivo podem consegui-lo por meio da flag "classificada", ao passo que as sub-rotinas que necessitam de informações sobre modificação de registros vão obtê-las por meio da flag "modificada".

RMOD, para indicar se um registro foi modificado, e CLAS, para indicar se o arquivo foi classificado, são nomes apropriados para as duas flags.

Quando apresentamos o programa, na página 399, a linha 1230 continha a instrução LET GRAVO=0. A variável GRAVO não fora utilizada, mas, com a inclusão da linha, percebemos que RMOD apenas não seria suficiente.

Um nome mais adequado para essa flag seria CLAS (para indicar que o arquivo está classificado). A linha 1230 original foi alterada para:

```
1230 LET CLAS = 1
```

Há agora quatro estados possíveis com relação às condições do arquivo de dados, que são:

| RMOD | CLAS | |
|---|---|---|
| 0 | 0 | Não modificado, não classificado (inconsistente) |
| 1 | 0 | Modificado, não classificado |
| 0 | 1 | Não modificado, classificado |
| 1 | 1 | Modificado, classificado |

RMOD=0 e CLAS=0 são procedimentos inconsistentes porque o programa garante que o arquivo de dados sempre esteja classificado, antes da gravação. Quando se processa o programa, RMOD recebe o valor 0 (linha 1220), indicando que não ocorreram modificações, e CLAS recebe o valor 1 (linha 1230), significando que o arquivo foi classificado.

Alguma operação que altere um registro (como *ACRREG*, *ELMREG* ou *MODREG*) atribui a RMOD o valor 1 e essa flag não será recolocada em 0 por alguma operação posterior. CLAS, que recebe de início o valor 1, passa de novo a 0 por meio de um procedimento que pode significar que os dados saíram da ordem (como em *MODREG*, se o campo do nome for alterado). Qualquer rotina que parte da suposição de que os dados já estão classificados (como *ENCREG*) sempre verifica CLAS e chama a rotina de classificação, se CLAS=0. Por meio do emprego dessas duas flags, em vez de apenas RMOD, o programa pode se encerrar sem gravar o arquivo de dados, caso não tenham ocorrido alterações de ordem no processamento atual. Nada o induz a isso, se ocorrer classificação após a alteração de um registro.

Outra variável até agora não utilizada é CORR. Ela serve para reter a posição atual do registro "corretamente em uso" na matriz, após a rotina de busca ter localizado essa posição. Depois de receber a atribuição de um valor, CORR transporta dados relativos ao registro objeto para outras rotinas do programa. O final da rotina *ENCREG* (busca) sofreu modificação nas linhas 13320 e 13330 para passar o valor de CORR a 0, se a busca não encontrar o registro objeto; e a MEI, se a busca tiver bom resultado.

A linha 13340 desvia para a sub-rotina *NAOREG* se CORR corresponder a 0. Esse procedimento exibe mensagem indicando que o registro não foi encontrado e apresenta a chave de busca, CAMPNOM$(TAMA). *NAOREG* retorna ao menu principal após o pressionamento da barra de espaço. Pode ser modificada com facilidade, dando ao usuário as duas oportunidades seguintes:

```
PRESSIONE RETURN PARA NOVA TENTATIVA OU A
BARRA DE ESPAÇO PARA CONTINUAR
```

O modo mais simples para conseguir isso parece ser uma nova chamada de *ENCREG*, se a tecla RETURN for pressionada. No entanto, a chamada de uma rotina de dentro de si mesma, embora válida em BASIC, causará "confusão" no endereço de retorno e fará com que a sub-rotina se repita, mesmo quando desnecessária. Há maneiras de contornar o problema, mas elas complicam a programação.

O método mais fácil seria usar uma flag (como NREG, para não registro) e atribuir-lhe o valor 0 em *NAOREG*, possibilitando que a sub-rotina retornasse pelo procedimento normal e forçasse uma volta para *EXECUT* no programa principal. Por exemplo: 95 IF NREG = 0 THEN 80. Com essa aborda-

gem, porém, a codificação começa a ficar desordenada. De acordo com nossa regra de evitar GOTOs, decidimos simplificar e apenas retornar ao menu principal se *ENCREG* não encontrar registro.

Houve um pequeno acréscimo em *MODNOM*, na linha 10490. A variável numérica S também deve receber valor 0 (LET S=0). A não execução desse procedimento, em algumas circunstâncias inabituais, ocasiona o funcionamento ineficaz de *MODNOM*.

*MODREG* corresponde a outra rotina implementada nesta versão final do programa. Ela primeiro localiza o registro a ser modificado, chamando *ENCREG* (linha 14120). Essa linha, por sua vez, chama a 13030, não a 13000, para suprimir a instrução de limpeza da tela de *ENCREG*. Caso não se localize o registro, o programa retornará ao menu principal pelo procedimento habitual (linha 14130). Se localizado, o registro aparece na tela e o usuário recebe a seguinte solicitação:

QUER MODIFICAR O NOME?
TECLE RETURN PARA ENTRAR COM O NOVO NOME
OU A BARRA DE ESPAÇO PARA O PRÓXIMO CAMPO

A rotina que determina a opção necessária pode ser encontrada nas linhas de 14190 a 14280.

As linhas de 14190 a 14220 constituem um loop simples que se encerra apenas se a barra de espaço ou RETURN for pressionada. Se A$ não for CHR$(13) — o valor ASCII para RETURN —, nem um espaço (você pode também usar CHR$(32) em vez de " "), I recebe o valor 0 e o loop se repete. Caso a tecla pressionada seja RETURN (e o campo do nome deva ser alterado), as poucas linhas seguintes vão preencher CAMPNOM$(CORR) com o novo nome; determinar RMOD; atribuir o valor 0 a CLAS; e chamar *MODNOM*, além de preencher CAMPMOD$(CORR) com o nome padronizado criado por *MODNOM* e localizado em CAMPMOD$(TAMA).

O restante de *MODNOM* funciona do mesmo modo. Observe, porém, que a modificação de outros campos determina RMOD sem atribuir o valor 0 a CLAS (ver a linha 14490, por exemplo). Isso porque a alteração apenas do campo do nome significa que o arquivo de dados pode estar fora de ordem, pois é classificado pelo nome. A alteração de qualquer outro campo indica que um registro foi modificado (RMOD=1) e que o arquivo deve ser gravado ao se encerrar o programa.

Outra rotina implementada é *ELMREG*, para eliminar um registro. Trata-se de um procedimento bastante direto. *ELMREG* limpa a tela (linha 15020) e apresenta mensagem informando sobre o que acontece no programa. A seguir, chama *ENCREG* para localizar o registro a ser eliminado. Uma escolha é então oferecida: tecle RETURN para eliminar ou a BARRA DE ESPAÇO para continuar. Também se apresenta uma mensagem de advertência (linha 15160). Abordagem ainda melhor pode ser a mensagem VOCÊ TEM CERTEZA?, se RETURN for pressionada e, em seguida, a eliminação do registro, se a tecla S for pressionada (isto é, IF INKEY$ = "S" THEN...).

*ELMREG* não atribui o valor 0 à flag CLAS. Uma vez que o arquivo já está em ordem alfabética, a eliminação de todo um registro não altera essa ordem. Isso significa que o arquivo foi modificado e, desse modo, RMOD recebe o valor 0 na linha 15340 e TAMA reduz-se em uma unidade na linha 15350 (pois o arquivo agora tem um registro válido a menos). Todos os registros deslocaram-se para "uma unidade abaixo" nas linhas de 15260 a 15320.

Você deve ter notado que *ENCREG* inclui a chamada condicional de uma sub-rotina denominada *IMPCOR* para impressão do registro atual localizado por *ENCREG*. Se você não tem impressora, apenas substitua a linha 13540 por uma instrução REM (para ampliação futura) e omita as linhas de 13600 a 13690.

Desenvolvemos, na agenda de endereços, todas as opções fundamentais apresentadas no menu principal: o encontro, a inclusão, a alteração e a eliminação de registros, além do desvio do programa. O objetivo dessa agenda de endereços computadorizada é exemplificar o modo como se deve especificar, elaborar e desenvolver programas. Uma modificação fundamental para quem deseja um software de utilização mais flexível consiste em resolver o problema que surgirá se TAMA chegar a ser 51. Isso acontecerá tão logo haja cinqüenta registros no arquivo.

No próximo fascículo, apresentaremos um exemplo de programa das demais funções de nossa agenda, ou seja, encontrar nomes, registros de uma cidade, registros de uma inicial e listar todos os registros.

## Programa para agenda de endereços

```
10 REM "PRGPRN"
20 REM *INICIL*
30 GOSUB 1000
40 REM *SAUDAR*
50 GOSUB 3000
60 REM *ESCLHA*
70 GOSUB 3500
80 REM *EXECUT*
90 GOSUB 4000
100 IF ESCL < > 9 THEN 60
110 END
1000 REM SUB-ROTINA *INICIL*
1010 GOSUB 1100: REM SUB-ROTINA *CRIMAT* (CRIA MATRIZES)
1020 GOSUB 1400: REM SUB-ROTINA *LERARQ* (LE ARQUIVOS)
1030 GOSUB 1600: REM SUB-ROTINA *ESTFLG* (ESTABELECE FLAGS)
1040 REM
1050 REM
1060 REM
1070 REM
1080 REM
1090 RETURN
1100 REM SUB-ROTINA *CRIMAT* (CRIA MATRIZES)
1110 DIM CAMPNOM$(50)
1120 DIM CAMPMOD$(50)
1130 DIM CAMPRUA$(50)
1140 DIM CAMPCID$(50)
1150 DIM CAMPEST$(50)
1160 DIM CAMPTEL$(50)
1170 DIM CAMPIND$(50)
1180 REM
1190 REM
1200 REM
1210 LET TAMA=0
1220 LET RMOD=0
1230 LET CLAS=1
```

```
1240 LET CORR=0
1250 REM
1260 REM
1270 REM
1280 REM
1290 REM
1300 RETURN
1400 REM SUB-ROTINA *LERARQ*
1410 OPEN "I", #1, "AGEN.DAD"
1420 INPUT #1, TEST$
1430 IF TEST$="@PRIMEIRO" THEN GOTO 1540: REM CLOSE E RETURN
1440 LET CAMPNOM$(1)=TEST$
1450 INPUT #1,CAMPMOD$(1),CAMPRUA$(1),CAMPCID$(1),
     CAMPEST$(1),CAMPTEL$(1)
1460 INPUT #1,CAMPIND$(1)
1470 LET TAMA=2
1480 FOR L=2 TO 50
1490 INPUT #1,CAMPNOM$(L),CAMPRUA$(L),CAMPCID$(L),
     CAMPEST$(L)
1500 INPUT #1,CAMPTEL$(L),CAMPIND$(L)
1510 LET TAMA=TAMA+1
1520 IF EOF(1)=-1 THEN LET L=50
1530 NEXT L
1540 CLOSE #1
1550 RETURN
1600 REM SUB-ROTINA *ESTFLG*
1610 REM ESTABELECE FLAGS APOS *LERARQ*
1620 REM
1630 REM
1640 IF TEST$="@PRIMEIRO" THEN LET TAMA=1
1650 REM
1660 REM
1670 REM
1680 REM
1690 RETURN
3000 REM SUB-ROTINA *SAUDAR*
3010 PRINT CHR$(12): REM LIMPA TELA
3020 PRINT
3030 PRINT
3040 PRINT
3050 PRINT
3060 PRINT TAB(14);"*BENVINDO AO*"
3070 PRINT TAB(4);"*MICROCOMPUTADOR — CURSO BASICO*"
3080 PRINT TAB(1);"*AGENDA DE ENDERECOS
     COMPUTADORIZADA*"
3090 PRINT
3100 PRINT " (PRESSIONAR A BARRA DE ESPACO PARA CONTINUAR) "
3110 FOR L=1 TO 1
3120 IF INKEY$ < > " " THEN L=0
3130 NEXT L
3140 PRINT CHR$(12)
3150 RETURN
3500 REM SUB-ROTINA *ESCLHA*
3510 REM
3520 IF TEST$="@PRIMEIRO" THEN GOSUB 3860: REM SUB-ROTINA
     *PRIMEI*
3530 IF TEST$="@PRIMEIRO" THEN RETURN
3540 REM "MENESC"
3550 PRINT CHR$(12)
3560 PRINT "ESCOLHER UM DOS SEGUINTES:"
3570 PRINT
3580 PRINT
3590 PRINT
3600 PRINT "1. ENCONTRAR UM REGISTRO (PELO NOME) "
3610 PRINT "2. ENCONTRAR NOMES (POR TRECHO DE UM NOME) "
3620 PRINT "3. ENCONTRAR REGISTROS (DE UMA CIDADE) "
3630 PRINT "4. ENCONTRAR REGISTROS (DE UMA INICIAL) "
3640 PRINT "5. LISTAR TODOS OS REGISTROS"
3650 PRINT "6. ACRESCENTAR UM REGISTRO"
3660 PRINT "7. MODIFICAR UM REGISTRO"
3670 PRINT "8. ELIMINAR UM REGISTRO"
3680 PRINT "9. SAIR E GRAVAR"
3690 PRINT
3700 PRINT
3710 REM "ATRESC"
3720 REM
3730 LET L=0
3740 LET I=0
3750 FOR L=1 TO 1
3760 PRINT "ESCOLHER DE 1 A 9"
3770 FOR I=1 TO 1
3780 LET A$=INKEY$
3790 IF A$=" " THEN I=0
3800 NEXT I
3810 LET ESCL=VAL(A$)
3820 IF ESCL<1 THEN L=0
3830 IF ESCL>9 THEN L=0
3840 NEXT L
3850 RETURN
3860 REM SUB-ROTINA *PRIMEI* (DA MENSAGEM)
3870 LET ESCL=6
3880 PRINT CHR$(12): REM LIMPA TELA
3890 PRINT
3900 PRINT TAB(8);"NAO EXISTEM REGISTROS NO"
3910 PRINT TAB(8);"ARQUIVO. VOCE DEVERA COMECAR"
3920 PRINT TAB(8);"ACRESCENTANDO UM REGISTRO"
3930 PRINT
3940 PRINT " (PRESSIONAR A BARRA DE ESPACO PARA
     CONTINUAR) "
3950 FOR B=1 TO 1
3960 IF INKEY$ <> " " THEN B=0
3970 NEXT B
3980 PRINT CHR$(12): REM LIMPA TELA
3990 RETURN
4000 REM SUB-ROTINA *EXECUT*
4010 REM
4020 REM
4030 REM
4040 IF ESCL=1 THEN GOSUB 13000: REM *ENCREG*
4050 REM 2 = *ENCNOM*
4060 REM 3 = *ENCCID*
4070 REM 4 = *ENCINI*
4080 REM 5 = *LISREG*
4090 IF ESCL=6 THEN GOSUB 10000: REM *ACRREG*
4100 IF ESCL=7 THEN GOSUB 14000: REM *MODREG*
4110 IF ESCL=8 THEN GOSUB 15000: REM *ELMREG*
4120 IF ESCL=9 THEN GOSUB 11000: REM *SAIPRG*
4130 REM
4140 RETURN
10000 REM SUB-ROTINA *ACRREG*
10010 PRINT CHR$(12): REM LIMPA TELA
10020 INPUT "FORNECER O NOME";CAMPNOM$(TAMA)
10030 INPUT "FORNECER A RUA";CAMPRUA$(TAMA)
10040 INPUT "FORNECER A CIDADE";CAMPCID$(TAMA)
10050 INPUT "FORNECER O ESTADO";CAMPEST$(TAMA)
10060 INPUT "FORNECER O NUMERO TELEFONICO";CAMPTEL$(TAMA)
10070 LET RMOD=1: LET CLAS=0: REM MODIFICADO E NAO
      CLASSIFICADO
10080 LET CAMPIND$(TAMA)=STR$(TAMA)
10090 LET TEST$=" "
10100 GOSUB 10200: REM *MODNOM*
10110 LET ESCL=0
10120 LET TAMA=TAMA+1
10130 REM
10140 REM
10150 RETURN
10200 REM ROTINA *MODNOM*
10210 REM CONVERTE O CONTEUDO DE CAMPNOM$ EM MAIUSCULAS,
10220 REM REMOVE SINAIS E ARQUIVA NA ORDEM:
10230 REM SOBRENOME+ESPACO+PRIMEIRONOME EM CAMPMOD$
10240 REM
10250 LET N$=CAMPNOM$(TAMA)
10260 FOR L=1 TO LEN(N$)
10270 LET TEMP$=MID$(N$,L,1)
10280 LET T=ASC(TEMP$)
10290 IF T>=97 THEN T=T-32
10300 LET TEMP$=CHR$(T)
10310 LET P$=P$+TEMP$
10320 NEXT L
10330 LET N$=P$
10340 REM ENCONTRAR ULTIMO ESPACO
10350 FOR L=1 TO LEN(N$)
10360 IF MID$(N$,L,1)=" " THEN S=L
10370 NEXT L
10380 REM ELIMINAR SINAIS E ARQUIVAR PRIMEIRONOME
10390 REM EM CNOM$
10400 FOR L=1 TO S-1
10410 IF ASC(MID$(N$,L,1)) > 64 THEN
      CNOM$=CNOM$+MID$(N$,L,1)
10420 NEXT L
10430 REM ELIMINA SINAIS E ARQUIVA SOBRENOME
10440 REM EM SNOM$
10450 FOR L=S+1 TO LEN(N$)
10460 IF ASC(MID$(N$,L,1)) > 64 THEN SNOM$=SNOM$+MID$(N$,L,1)
10470 NEXT L
10480 LET CAMPMOD$(TAMA)=SNOM$+" "+CNOM$
10490 LET P$=" ": LET N$=" ": LET SNOM$=" ": LET CNOM$=" ": LET S=0
10500 RETURN
11000 REM SUB-ROTINA *SAIPRG*
11010 REM CLASSIFICA E GRAVA O ARQUIVO
11020 REM SE ALGUM REGISTRO FOI
11030 REM MODIFICADO (RMOD =1)
11040 REM OU NAO CLASSIFICADO (CLAS=0)
11050 REM RMOD=0 E CLAS=0 NAO E LEGAL
11060 REM
11070 IF RMOD=0 AND CLAS=1 THEN RETURN
11080 IF RMOD=1 AND CLAS=0 THEN GOSUB 11200: REM
      *CLAREG*
11090 GOSUB 12000: REM *GRAREG*
11100 RETURN
11200 REM SUB-ROTINA *CLAREG*
11210 REM CLASSIFICA TODOS OS REGISTROS POR ORDEM
11220 REM ALFABETICA SEGUNDO CAMPMOD$ E ATUALIZA
11230 REM CAMPIND$
11240 REM
11250 LET S=0
11260 FOR L=1 TO TAMA-2
11270 IF CAMPMOD$(L) > CAMPMOD$(L+1) THEN GOSUB 11350
11280 NEXT L
11290 IF S=1 THEN 11250
11300 REM
11310 REM
11320 LET CLAS=1: REM FLAG DE ARQUIVO CLASSIFICADO
11330 REM
11340 RETURN
11350 REM SUB-ROTINA *INVREG*
11360 LET TCAMNOM$=CAMPNOM$(L)
11370 LET TCAMMOD$=CAMPMOD$(L)
11380 LET TCAMRUA$=CAMPRUA$(L)
11390 LET TCAMCID$=CAMPCID$(L)
11400 LET TCAMEST$=CAMPEST$(L)
11410 LET TCAMTEL$=CAMPTEL$(L)
11420 REM
11430 LET CAMPNOM$(L)=CAMPNOM$(L+1)
11440 LET CAMPMOD$(L)=CAMPMOD$(L+1)
11450 LET CAMPRUA$(L)=CAMPRUA$(L+1)
11460 LET CAMPCID$(L)=CAMPCID$(L+1)
11470 LET CAMPEST$(L)=CAMPEST$(L+1)
11480 LET CAMPTEL$(L)=CAMPTEL$(L+1)
11490 LET CAMPIND$(L)=STR$(L)
11500 REM
11510 LET CAMPNOM$(L+1)=TCAMNOM$
11520 LET CAMPMOD$(L+1)=TCAMMOD$
11530 LET CAMPRUA$(L+1)=TCAMRUA$
11540 LET CAMPCID$(L+1)=TCAMCID$
11550 LET CAMPEST$(L+1)=TCAMEST$
11560 LET CAMPTEL$(L+1)=TCAMTEL$
11570 LET CAMPIND$(L+1)=STR$(L+1)
11580 LET S=1
11590 REM
11600 RETURN
12000 REM SUB-ROTINA *GRAREG*
12010 REM
12020 REM
```

```
12030 OPEN "O",#1, "AGEN.DAD"
12040 REM
12050 FOR L=1 TO TAMA-1
12060 PRINT #1,CAMPNOM$(L);",";CAMPMOD$(L);",";
      CAMPRUA$(L);",";CAMPCID$(L)
12070 PRINT #1,CAMPEST$(L);",";CAMPTEL$(L);",";CAMPIND$(L)
12080 NEXT L
12090 REM
12100 REM
12110 REM
12120 REM
12130 CLOSE #1
12140 REM
12150 RETURN
13000 REM SUB-ROTINA *ENCREG* (ENCONTRAR UM REGISTRO)
13010 PRINT CHR$(12): REM LIMPA TELA
13020 REM
13030 IF CLAS=0 THEN GOSUB 11200: REM *CLAREG*
13040 PRINT
13050 PRINT
13060 PRINT TAB(3);"*PROCURANDO UM REGISTRO PELO NOME*"
13070 PRINT
13080 PRINT
13090 PRINT TAB(9);"DIGITE O NOME COMPLETO"
13100 PRINT TAB(7);"NA ORDEM DE NOME SOBRENOME"
13110 PRINT
13120 PRINT
13130 REM
13140 INPUT "O NOME E";CAMPNOM$(TAMA)
13150 GOSUB 10200: REM SUB-ROTINA *MODNOM*
13160 LET CHAVE$=CAMPMOD$(TAMA)
13170 REM
13180 REM
13190 REM
13200 REM
13210 REM
13220 LET FIM=1
13230 LET INI=TAMA-1
13240 FOR L=1 TO 1
13250 LET MEI=INT((FIM+INI)/2)
13260 IF CAMPMOD$(MEI) < > CHAVE$ THEN L=0
13270 IF CAMPMOD$(MEI) < CHAVE$ THEN FIM=MEI+1
13280 IF CAMPMOD$(MEI) > CHAVE$ THEN INI=MEI-1
13290 IF FIM > INI THEN L=1
13300 NEXT L
13310 REM
13320 IF FIM > INI THEN LET CORR=0
13330 IF FIM <= INI THEN LET CORR=MEI
13340 IF CORR=0 THEN GOSUB 13700: REM *NAOREG*
13350 IF CORR=0 THEN RETURN
13360 REM
13370 REM
13380 PRINT CHR$(12)
13390 PRINT
13400 PRINT TAB(15);"*REGISTRO ENCONTRADO*"
13410 PRINT
13420 PRINT "NOME: ";CAMPNOM$(CORR)
13430 PRINT "RUA: ";CAMPRUA$(CORR)
13440 PRINT "CIDADE: ";CAMPCID$(CORR)
13450 PRINT "ESTADO: ";CAMPEST$(CORR)
13460 PRINT "TELEFONE: ";CAMPTEL$(CORR)
13470 PRINT
13480 PRINT TAB(7);"PRESSIONE QUALQUER LETRA PARA IMPRIMIR"
13490 PRINT TAB(7);"OU A BARRA DE ESPACO PARA CONTINUAR"
13500 FOR I=1 TO 1
13510 LET A$=INKEY$
13520 IF A$="" THEN I=0
13530 NEXT I
13540 IF A$ < > " " THEN GOSUB 13600: REM *IMPCOR*
13550 RETURN
13600 REM SUB-ROTINA *IMPCOR* (IMPRIME O REGISTRO CORRENTE)
13610 LPRINT
13620 LPRINT "NOME: ";CAMPNOM$(CORR)
13630 LPRINT "RUA: ";CAMPRUA$(CORR)
13640 LPRINT "CIDADE: ";CAMPCID$(CORR)
13650 LPRINT "ESTADO: ";CAMPEST$(CORR)
13660 LPRINT "TELEFONE: ";CAMPTEL$(CORR)
13670 LPRINT
13680 LPRINT
13690 RETURN
13700 REM SUB-ROTINA *NAOREG* (REGISTRO NAO ENCONTRADO)
13710 PRINT CHR$(12): REM LIMPA TELA
13720 PRINT TAB(11);"REGISTRO NAO ENCONTRADO"
13730 PRINT TAB(4);"NO FORMATO: ";CAMPNOM$(TAMA)
13740 PRINT
13750 PRINT " (PRESSIONAR A BARRA DE ESPACO PARA CONTINUAR) "
13760 FOR I=1 TO 1
13770 IF INKEY$ < > " " THEN I=0
13780 NEXT I
13790 RETURN
14000 REM SUB-ROTINA *MODREG* (MODIFICAR UM REGISTRO)
14010 REM
14020 PRINT CHR$(12): REM LIMPA TELA
14030 PRINT
14040 PRINT
14050 PRINT
14060 PRINT
14070 PRINT TAB(10);"*PARA MODIFICAR UM REGISTRO*"
14080 PRINT TAB(5);"*PRIMEIRO ENCONTRE O REGISTRO DESEJADO*"
14090 REM
14100 REM
14110 REM
14120 GOSUB 13030: REM SUB-ROTINA *ENCREG* SEM LIMPAR A TELA
14130 IF CORR=0 THEN RETURN: REM REGISTRO NAO ENCONTRADO
14140 PRINT
14150 PRINT TAB(8);"QUER MODIFICAR O NOME?"
14160 PRINT
14170 PRINT TAB(2);"TECLE RETURN PARA ENTRAR COM O NOVO
      NOME"
14180 PRINT TAB(2);"OU A BARRA DE ESPACO PARA O PROXIMO
      CAMPO"
14190 FOR I=1 TO 1
14200 LET A$=INKEY$
14210 IF A$ < > CHR$(13) AND A$ < > " " THEN I=0
14220 NEXT I
14230 IF A$=CHR$(13) THEN INPUT "NOVO NOME: ";
      CAMPNOM$(CORR)
14240 IF A$=CHR$(13) THEN RMOD=1
14250 IF A$=CHR$(13) THEN CLAS=0
14260 IF A$=CHR$(13) THEN CAMPNOM$(TAMA)=
      CAMPNOM$(CORR)
14270 IF A$=CHR$(13) THEN GOSUB 10200: REM SUB-ROTINA
      *MODNOM*
14280 IF A$=CHR$(13) THEN LET CAMPMOD$(CORR)=
      CAMPMOD$(TAMA)
14290 PRINT
14300 PRINT TAB(8);"QUER MODIFICAR A RUA?"
14310 PRINT
14320 PRINT TAB(2);"TECLE RETURN PARA ENTRAR COM A
      NOVA RUA"
14330 PRINT TAB(2);"OU A BARRA DE ESPACO PARA O PROXIMO
      CAMPO"
14340 FOR I=1 TO 1
14350 LET A$=INKEY$
14360 IF A$ < > CHR$(13) AND A$ < > " " THEN I=0
14370 NEXT I
14380 IF A$=CHR$(13) THEN RMOD=1
14390 IF A$=CHR$(13) THEN INPUT "NOVA RUA: ";
      CAMPRUA$(CORR)
14400 PRINT
14410 PRINT TAB(8);"QUER MODIFICAR A CIDADE?"
14420 PRINT
14430 PRINT TAB(2);"TECLE RETURN PARA ENTRAR COM A NOVA
      CIDADE"
14440 PRINT TAB(2);"OU A BARRA DE ESPACO PARA O PROXIMO
      CAMPO"
14450 FOR I= 1 TO 1
14460 LET A$=INKEY$
14470 IF A$ < > CHR$(13) AND A$ < > " " THEN I=0
14480 NEXT I
14490 IF A$=CHR$(13) THEN RMOD=1
14500 IF A$=CHR$(13) THEN INPUT "NOVA CIDADE: ";
      CAMPCID$(CORR)
14510 PRINT
14520 PRINT TAB(8);"QUER MODIFICAR O ESTADO?"
14530 PRINT
14540 PRINT TAB(2);"TECLE RETURN PARA ENTRAR COM O NOVO
      ESTADO"
14550 PRINT TAB(2);"OU A BARRA DE ESPACO PARA O PROXIMO
      CAMPO"
14560 FOR I=1 TO 1
14570 LET A$=INKEY$
14580 IF A$ < > CHR$(13) AND A$ < > " " THEN I=0
14590 NEXT I
14600 IF A$=CHR$(13) THEN RMOD=1
14610 IF A$=CHR$(13) THEN INPUT "NOVO ESTADO: ";
      CAMPEST$(CORR)
14620 PRINT
14630 PRINT TAB(8);"QUER MODIFICAR O TELEFONE?"
14640 PRINT
14650 PRINT TAB(2);"TECLE RETURN PARA ENTRAR COM O NOVO
      TELEFONE"
14660 PRINT TAB(2);"OU A BARRA DE ESPACO PARA CONTINUAR"
14670 FOR I=1 TO 1
14680 LET A$=INKEY$
14690 IF A$ < > CHR$(13) AND A$ < > " " THEN I=0
14700 NEXT I
14710 IF A$=CHR$(13) THEN RMOD=1
14720 IF A$=CHR$(13) THEN INPUT "NOVO TELEFONE: ";
      CAMPTEL$(CORR)
14730 REM
14740 REM
14750 RETURN
15000 REM SUB-ROTINA *ELMREG* (ELIMINA UM REGISTRO)
15010 REM
15020 PRINT CHR$(12): REM LIMPA TELA
15030 PRINT
15040 PRINT
15050 PRINT
15060 PRINT
15070 PRINT TAB(10);"*PARA ELIMINAR UM REGISTRO*"
15080 PRINT TAB(5);"*LOCALIZE PRIMEIRO O REGISTRO DESEJADO*"
15090 REM
15100 REM
15110 REM
15120 GOSUB 13030: REM SUB-ROTINA *ENCREG* SEM LIMPAR
      A TELA
15130 IF CORR=0 THEN RETURN: REM REGISTRO NAO ENCONTRADO
15140 PRINT
15150 PRINT TAB(3);"VOCE DESEJA ELIMINAR ESTE REGISTRO?"
15160 PRINT TAB(3);"*ATENCAO* - NAO HAVERA OUTRA CHANCE"
15170 PRINT
15180 PRINT TAB(7);"TECLE RETURN PARA ELIMINAR"
15190 PRINT TAB(3);"OU A BARRA DE ESPACO PARA CONTINUAR"
15200 FOR I=1 TO 1
15210 LET A$=INKEY$
15220 IF A$ < > CHR$(13) AND A$ < > " " THEN I=0
15230 NEXT I
15240 IF A$=" " THEN RETURN
15250 FOR L=CORR TO TAMA-2
15260 LET CAMPNOM$(L)=CAMPNOM$(L+1)
15270 LET CAMPMOD$(L)=CAMPMOD$(L+1)
15280 LET CAMPRUA$(L)=CAMPRUA$(L+1)
15290 LET CAMPCID$(L)=CAMPCID$(L+1)
15300 LET CAMPEST$(L)=CAMPEST$(L+1)
15310 LET CAMPTEL$(L)=CAMPTEL$(L+1)
15320 LET CAMPIND$(L)=STR$(L)
15330 NEXT L
15340 LET RMOD=1
15350 LET TAMA=TAMA-1
15360 REM
15370 REM
15380 REM
15390 RETURN
```

# Grace Hopper

## Uma americana teve participação decisiva no desenvolvimento de linguagens de alto nível.



A ciência da computação é, em geral, considerada uma área estritamente masculina. No entanto, as mulheres estão conquistando seu espaço também no desenvolvimento e na aplicação de computadores. Uma pioneira nesse campo foi a americana Grace Hopper, com contribuição significativa na área de software — ela criou o compilador e ajudou a inventar a linguagem COBOL. Também foi a primeira pessoa a isolar um erro no computador e corrigi-lo — ou "debugá-lo", no jargão dos especialistas — com êxito.

Após concluir o curso de pós-graduação em Yale, Grace Hopper voltou a sua universidade de origem, em Vassar, como professora de Matemática. Permaneceu no cargo até os 39 anos, quando a convocaram, em conseqüência da entrada dos Estados Unidos na Segunda Guerra Mundial, para trabalhar no projeto de computação da Marinha. Em 1945 foi enviada à Universidade de Harvard para assessorar o físico Howard Aiken. Ele já havia apresentado à IBM, em 1937, a idéia de construir um computador usando equipamento de tabulação adaptado. Seu primeiro projeto, embora mecânico, obteve êxito e conseguiu animar a IBM a investir num modelo aperfeiçoado que funcionaria com relés eletromecânicos. A máquina então desenvolvida ficou conhecida como Harvard Mark II.

Nessa época, para programar um computador, era necessário enrolar sua fiação a cada nova tarefa. Assim, no verão de 1945, Grace Hopper viu-se literalmente emaranhada em seu trabalho. Urgentes serviços de computação para balística faziam-se necessários para o esforço de guerra, e Aiken costumava entrar na oficina cobrando: "Por que você não está fazendo números, Hopper?" Após uma pane complicada do computador, quando se descobriu que o problema tinha sido uma mariposa que entrara pela janela e fora morta pelo interruptor de um relé, Grace respondeu: "Nós estamos tirando 'bugs' da máquina!" (*bug*, em inglês, significa "inseto" e passou a ser usado como sinônimo de "erro", em computação). Esse primeiro bug registrado foi cuidadosamente removido do relé com uma pinça e está preservado no Museu Naval, em Virgínia, junto ao Livro Diário do Harvard Mark II. Foi inscrito no registro de entrada, às 15h45, em 9 de setembro de 1945.

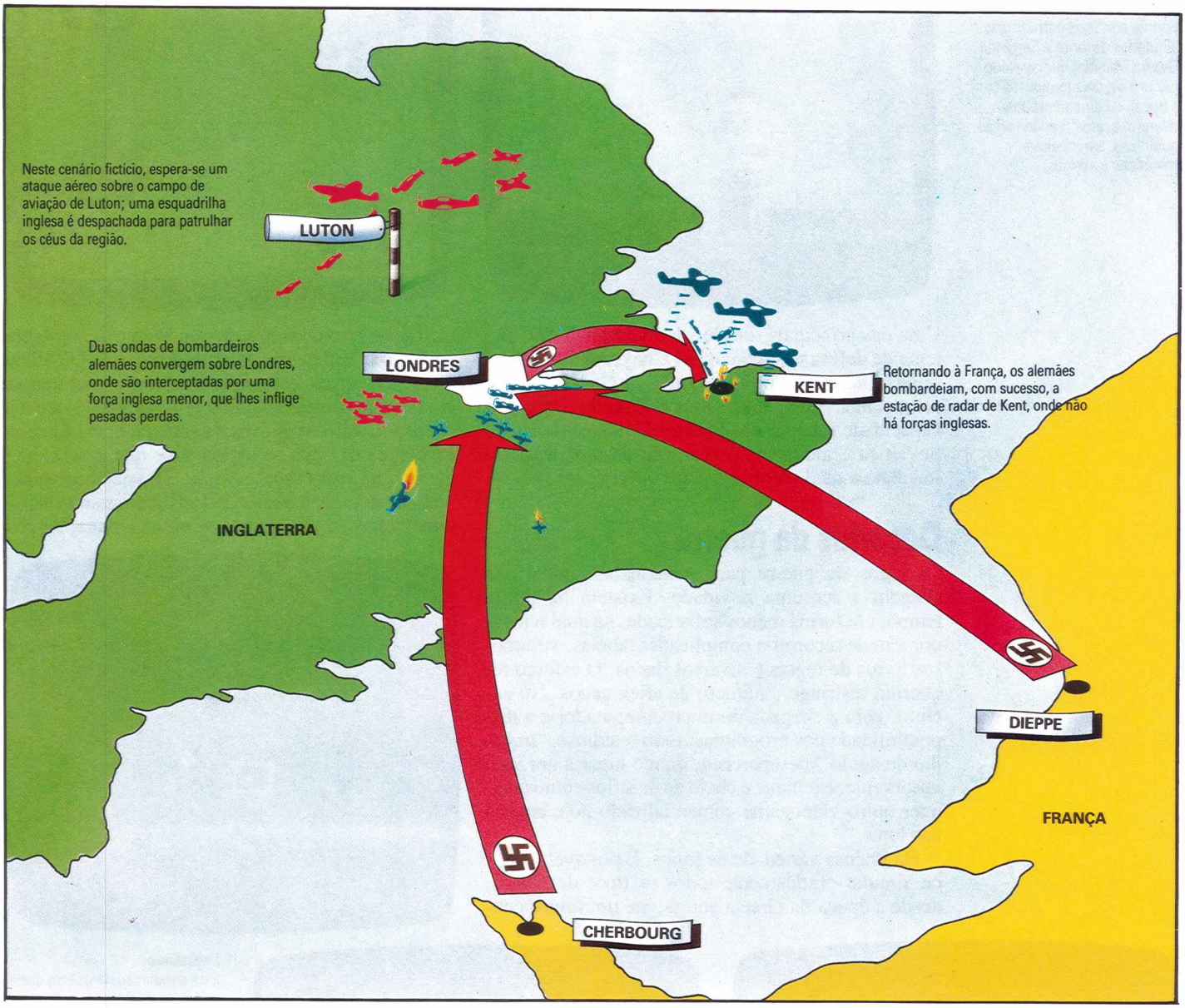
## Linguagem de programação

Nesse mesmo ano, outro computador, o ENIAC, estava sendo construído pelos engenheiros John Mauchly e Presper Eckert. Após a guerra, ambos montaram um negócio próprio para fabricar a versão comercial da máquina e convidaram Grace para juntar-se à equipe. A principal contribuição dela para o desenvolvimento desse computador, o UNIVAC (UNIVersal ACcounting machine), foi na criação de seu software. Enquanto tentava desenvolver programas para uso comercial no UNIVAC, Grace descobriu meios de não reescrever certas sub-rotinas repetitivas. Empregando a idéia, considerada notável na época, de que um computador podia escrever seus próprios programas, Grace criou a primeira linguagem de programação, junto com o compilador necessário para traduzi-la em linguagem de máquina, que recebeu o nome de "A-O". Quando esse compilador foi apresentado, causou incredulidade entre os especialistas. Eles achavam que suas máquinas só podiam fazer contas e manipular símbolos. Ficaram surpresos ao ver um computador pular para uma sub-rotina em sua biblioteca de armazenamento, ao encontrar um verbo no modo imperativo no começo do que parecia ser uma frase quase normal em inglês.

Em maio de 1959, Hopper (que tinha patente de capitão) foi convidada pelo Pentágono para fazer parte do grupo que trabalhava na criação e padronização de uma linguagem simples para computadores de uso comercial. Em menos de um ano, produziu-se a primeira versão do COmmon Business Oriented Language (COBOL). Grace contribuiu bastante para o trabalho, aproveitando o que havia de melhor em cada linguagem existente. Criou-se assim uma linguagem aceitável para a indústria, por sua simplicidade. Uma prova do sucesso é a sobrevivência do COBOL até hoje.

**COBOL**
Foi o COBOL uma das primeiras linguagens escritas com a intenção de tornar a programação acessível a leigos em matemática.
Essa linguagem incentiva o uso dos procedimentos generalizados, escritos em estilo narrativo, em vez de rotinas em código peculiares a determinado problema.
Um programa em COBOL é constituído de quatro unidades. O nome do programa, o autor e outras informações fazem parte da divisão de Identificação (Identification). Embora os programas em COBOL sejam portáteis (servem em muitas máquinas), todos os detalhes que dizem respeito ao computador para o qual foram originalmente escritos estão anotados na divisão Ambiental (Environment). Como os dados podem ser usados em muitas partes do mesmo programa, o COBOL tem uma divisão de Dados (Data). Os procedimentos necessários para executar os dados estão relacionados na divisão de Procedimentos (Procedure).

# Guerra na paz




**Você já pode comprar jogos que testam sua habilidade como estrategista e tático militar em simulações de batalhas históricas ou fictícias.**

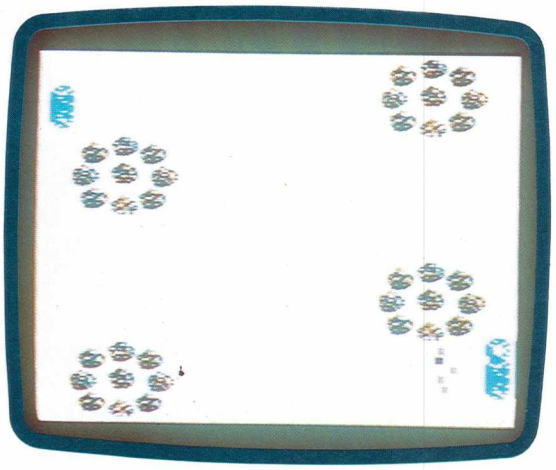
Nos dias de hoje, até militares profissionais se interessam pelos jogos de guerra, que usam para testar respostas planejadas a possíveis ataques inimigos ou a situações potencialmente ameaçadoras. Para esses entretenimentos sofisticados desenvolveram-se sistemas complexos de hardware e software, que simulam todos os aspectos conhecidos de um possível conflito, como a distribuição inicial das forças aliadas e inimigas, condições de suprimento, disponibilidade de reservas e assim por diante. O sistema também leva em conta condições climáticas adversas, mudanças de tática do inimigo, os efeitos da atividade dos quinta-colunistas, ou quaisquer variáveis que possam afetar o sucesso da operação militar.




**Batalha da Inglaterra**
O Fighter Command (da Strategic Simulations Inc., para o Apple) é um típico e sofisticado jogo de guerra, próprio para ser usado em microcomputadores. Antes de iniciá-lo, o jogador deve selecionar as opções entre uma grande variedade que inclui o tipo de avião e as condições climáticas. O jogo é exibido na forma de símbolos que se movimentam sobre o mapa, com informações adicionais em forma de texto.

Um mapa impresso e peças de papelão para referência visual adicional acompanham o disquete e as instruções.



**Comando Tático Blindado**
Este jogo (da Avalon Hill) pode ser usado com os computadores Atari, Apple, IBM PC e Commodore 64. Simula conflitos entre forças blindadas durante a Segunda Guerra Mundial. É disputado por um ou dois jogadores com a opção de cinco enredos diferentes, envolvendo forças britânicas, americanas, soviéticas e alemãs.

Uma das principais funções do radar NORAD, sistema de defesa nas montanhas Cheyenne, em Wyoming (apresentado no filme *Jogos de Guerra*), é a de acompanhar, atualizar e estimar de modo contínuo a capacidade relativa dos Estados Unidos e da União Soviética e ajudar na preparação de uma resposta imediata a qualquer novo desenvolvimento.

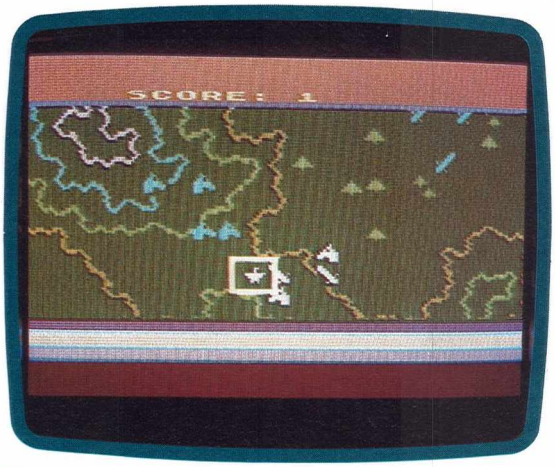
## Desafios da guerra

Os jogos de guerra para generais amadores não chegam a ser uma novidade. Existem há muito tempo, em forma menos sofisticada, na qual o jogador tem de recorrer a complicadas tabelas, volumosos livros de regras e diversos dados. O esforço requerido restringe o número de aficionados. No entanto, com a chegada do microcomputador e a disponibilidade dos programas, todo o tedioso "trabalho de apoio" desapareceu, dando lugar a um jogo absorvente, excitante e cheio de desafios como qualquer outro videogame comercializado hoje em dia nas lojas.

Há imensa variedade de jogos. É possível recriar ou simular praticamente todos os tipos de guerra, desde a época da Grécia antiga, até um futuro confronto imaginário entre as forças da Organização do Tratado do Atlântico Norte (OTAN) e as do Pacto de Varsóvia. Você pode travar batalhas aéreas e marítimas, combates espaciais e até guerras entre impérios mitológicos. O alcance é ilimitado.

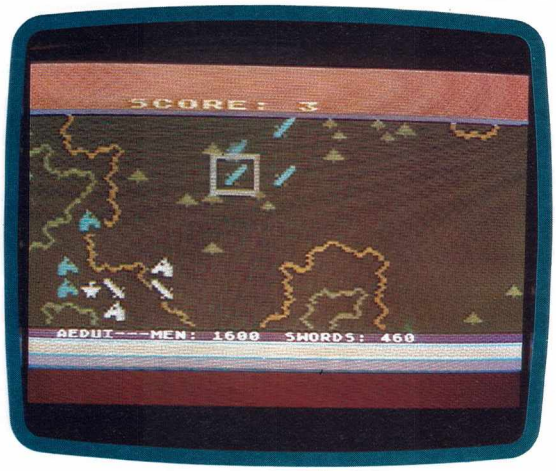
Os jogos históricos lhe dão a chance de descobrir qual foi o erro de Napoleão em Waterloo, ou tentar sobrepujar Hitler, frustrando a invasão da URSS em 1941. Os jogos espaciais oferecem oportunidade ainda maior de exercício da criatividade. Você não só manobra frotas através da galáxia, como também especifica os tipos de espaçonave que deseja. Sem dúvida, há concessões a fazer. Se você quer maior velocidade, talvez tenha de sacrificar os sistemas de armamentos, e os lugares que oferecem melhor proteção podem reduzir o suprimento de combustível. É preciso escolher a permuta que mais convém a seu estilo de empreender uma campanha. As opções variam de acordo com o jogador.

Ao contrário de seus equivalentes convencionais, os jogos de estratégia por computador não requerem










**Legionário**
Esta simulação de guerra entre as forças de César (você) e os bárbaros (o computador — Apple ou Atari) é jogada em tempo real. Infantaria, cavalaria e outras forças são representadas por símbolos que podem ser selecionados e movimentados por meio do cursor (quadrado branco), controlado por joystick. O jogo é produzido pela Avalon Hill.

habilidade ou conhecimento específico, e a maioria vem com observações e instruções para principiantes. Alguns jogos são classificados em elementares, intermediários e adiantados. Se você está pensando em se iniciar nos de estratégia, convém começar pelo nível elementar, absorvendo os conceitos básicos e as simulações, antes de passar para os níveis mais elevados.

O formato do jogo varia para adequar-se às diferentes necessidades dos diversos tipos de guerra representados. Em geral são utilizados grandes mapas. Quando eles extravasam os limites do vídeo, a tela funciona como uma janela que pode ser movimentada (por meio de joystick) sobre toda a sua extensão.

Para uma simulação histórica, os designers de jogos tentam reproduzir, o mais fielmente possível, o território em que se travou a batalha original. No Computer Bismarck, da SSI, a ação se desenvolve no Atlântico Norte, o que não causa muitos problemas de apresentação gráfica. Mas, para outro jogo da SSI, chamado Batalha da Normandia, o trabalho de ilustração foi mais difícil. Não só o território em geral tinha de estar correto, como também certos aspectos específicos — a costa, as praias, as cidades, os vilarejos e os rios. Nos jogos não históricos, o designer tem maior liberdade de criação, para permitir ao jogador o pleno uso das forças disponíveis, sem se esquecer, contudo, de incluir obstáculos e imprevistos em número suficiente, a fim de que as coisas não fiquem fáceis demais para um dos contendores.

O mapa também apresenta uma espécie de grade sobreposta que o subdivide, como se fosse um tabuleiro de xadrez, em quadrados (muitas vezes em hexágonos). Cada quadrado ou hexágono recebe um valor de acordo com o tipo de território nele contido. Esse valor reflete o grau de dificuldade que uma unidade enfrenta ao tentar entrar na área ou atravessá-la. O esforço para se movimentar nesse território faz com que a disponibilidade de pontos da unidade (uma espécie de cacife) seja reduzida no valor correspondente. Quando essa disponibilidade chega a zero, ou é menor que o valor da área em que a unidade se propõe a entrar, o único movimento possível é a volta.

## Objetivo: a vitória

O jogo quase sempre se divide em certo número de ''rodadas'' que apresentam o tempo transcorrido, e cada jogador recebe objetivos que devem ser alcançados no tempo determinado para vencer. Na maioria dos casos, não é necessário nem possível atingir todos os objetivos propostos. Portanto, a primeira decisão que o jogador tem de tomar corresponde à avaliação de suas chances e, de acordo com elas, deve determinar prioridades estratégicas. Nessa situação, cabe ao oponente impedir que o atacante atinja suas metas. Para ele também é impossível proteger tudo; em conseqüência, precisa decidir quando abandonar linhas insustentáveis, por quanto tempo se manter em fortificações, se pode ou não correr o risco de lançar um contra-ataque para recuperar posições perdidas ou destruir os preparativos do oponente visando a nova ofensiva, com o objetivo de ocupar mais territórios.

O jogador comunica-se com o programa por meio da representação gráfica e textual das forças sob seu comando que estão no mapa. A exibição gráfica permite localizar determinada unidade no campo de batalha e a textual fornece informações relacionadas com a eficiência dessa unidade em combate e a disponibilidade de pontos para movimentação. Efetua-se o deslocamento de tropas, em geral, depois que o jogador as aponta com o cursor. Assim que uma unidade é indicada, pode-se dar o comando para movimentá-la. No caso de um mapa com grade hexagonal, 1 manda a unidade para o norte, 2 remete-a para o nordeste, e assim por diante, de acordo com os pontos cardeais. Um número crescente desses jogos funciona com joysticks ou track balls. Utilizando esses periféricos, você pode simplesmente ''pegar'' suas forças e deslocá-las na direção desejada. Para encerrar o movimento, recorre-se ao comando FINISH ou F. Alguns jogos permitem que o jogador indique e desloque a unidade mais uma vez, se sua disponibilidade de pontos para a movimentação não estiver esgotada.

Quando as novas posições forem atingidas, o jogador leva o fato ao conhecimento do computador, utilizando para isso o comando EXECUTE ou E. Então, começa o combate propriamente dito.




**Frente Oriental**
O jogador assume o papel do Exército alemão tentando chegar a Moscou em 1941, enquanto o computador faz o papel das forças defensoras soviéticas. Escrito por Chris Crawford e distribuído pela Atari, o jogo incorpora novas características. Uma das mais interessantes é o modo pelo qual o mapa muda à medida que o ano vai passando. No outono, as florestas perdem as folhas; no inverno, os rios congelam, ficando o chão coberto de neve. Neste jogo, é extremamente difícil chegar a Moscou — aliás, na realidade foi impossível.

Durante essa fase, o computador indica as unidades amigas em condições de se confrontar com o inimigo, fornecendo informações sobre o poderio relativo das forças envolvidas. Baseado nesses dados, o jogador pode aceitar ou rejeitar as opções de combate que lhe são oferecidas. Quando a batalha estiver planejada e seus efeitos calculados e exibidos, o segundo jogador começa a agir.

Para muitos, o fascínio dos jogos de estratégia provém sobretudo de eles não terem apenas uma solução ''correta'' dos problemas que se apresentam. A diversão do jogador consiste em superar os obstáculos físicos e logísticos do território em que opera e em enfrentar o desafio de usar todos os recursos disponíveis para derrotar o inimigo (que, por sua vez, está fazendo exatamente o mesmo). Cada estrategista gostaria de vencer usando os esquemas mais audaciosos e as armadilhas montadas com mais esmero, mas, acima de tudo, ele quer a vitória!

# Kits de ferramentas

**Esses pacotes de software expandem um dialeto limitado de BASIC e facilitam a tarefa do programador.**

Os primeiros micros domésticos, como o Apple II e o Commodore PET, tinham capacidade limitada — foram projetados especialmente para manipular números e textos. O BASIC desenvolvido para eles apenas supria rotinas e comandos necessários a esses fins. Em conseqüência, escreveram-se muitos programas "utilitários" ou "kits de ferramentas", a maior parte em linguagem de máquina, que operavam fora da área de programação BASIC. Em forma de comandos diretos adicionais, ajudavam na construção e na correção de programas.

Desde essa época, os engenheiros têm criado uma infinidade de funções gráficas e sonoras, como resultado da explosão do interesse por jogos do tipo fliperama em microcomputadores. Cada modelo novo apresenta mais funções extensivas, que são logo embutidas no software escrito profissionalmente.

**Instrumentos de trabalho**
Estes são alguns dos pacotes e ferramentas de extensão do BASIC para os microcomputadores mais populares no exterior. Os pacotes para criação de facilidades para sprite em computadores que não as possuem normalmente estão crescendo em popularidade. No Brasil existem vários recursos desse tipo.

| Ferramentas | |
|---|---|
| SUPER TOOL KIT | Disponível para o Spectrum Sinclair 16 K e 48 K, da Nectarine |
| SPECTRUM EXTENDED BASIC | Para o Spectrum Sinclair 48 K, da CP Software |
| SPECTRUM KEYDEFINE | Para o Spectrum Sinclair 48 K, da Scientific Software |
| PROGRAMMER'S AID | Para o Commodore Vic-20, da Commodore |
| BUTI | Para o Commodore Vic-20, da Audiogenic |
| TOOL BOX | Para os micros ingleses modelo BBC, da BBC Software |
| SPRITE MAGIC | Disponível para o micro inglês Dragon 32, da Merlin Micro Systems |
| SPRITE GRAPHICS | Para o Spectrum Sinclair 48 K, da B Sides Software |
| SPRITE MASTER | Para o micro inglês BBC Model B, da Micro Dealer UK |

Contudo, com uma ou duas exceções, o BASIC incorporado tem pouco ou nenhum aperfeiçoamento sobre as versões anteriores. Isso resulta na obrigatoriedade de o usuário criar as rotinas, empregando repetidas vezes os comandos PEEK e POKE para incorporar essas novas características no conjunto de comandos disponíveis. Assim, existem alguns pacotes, kits de instrumentos e extensões para BASIC adaptáveis à maioria das máquinas mais populares de uso doméstico. Em geral, permitem fácil acesso aos recursos existentes (como editores de sprites ou de sons), ampliam as facilidades de software (por exemplo, criadores de sprite), ou funcionam como simples auxiliares na programação BASIC.

Extensões desse tipo podem se localizar em RAM, ROM interna ou cartucho de ROM. É preferível a extensão em ROM a uma que esteja carregada em RAM, já que ela não emprega a memória do usuário e não corre o risco de ser apagada por engano. Em geral, o programa escrito com ajuda de uma ferramenta dessas só pode ser executado em computador que possua equipamento similar. Contudo, existem pacotes geradores de programas independentes que são executados em versões não expandidas do computador. Essa é a base da maioria dos editores de gráficos e de sprites, como também de editores de sons.

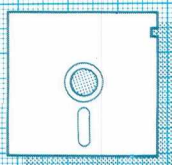
Algumas características úteis para se procurar nas extensões de BASIC são os comandos gráficos especiais (como PAINT, DRAW, PLOT, CIRCLE etc.) e comandos de som (SOUND, PLAY, MUSIC, ENVELOPE etc. ou palavras que descrevem um efeito sonoro, como BANG ou ZAP). Outros recursos úteis são os comandos de programação estruturada, como REPEAT-UNTIL e IF-THEN-ELSE. Palavras como essas capacitam o usuário a escrever programas que seguem uma seqüência lógica e evitam as linguagens confusas e difíceis de entender, resultantes do uso indiscriminado do GOTO.

## Simon's BASIC

A extensão mais completa da linguagem BASIC é o Simon's BASIC, próprio para o Commodore 64 na forma de cartucho ROM. O BASIC padrão do Commodore, que faz parte do 64, mostra-se um tanto antiquado, pois oferece um mínimo de comandos dedicados e nenhum de programação estruturada. Apesar de ter um hardware avançado, como sintetizador de som completo, alta resolução gráfica e gráficos sprites, o controle do BASIC sobre essas funções faz-se pelos comandos PEEK e POKE. O Simon's BASIC oferece uma extensão considerável ao BASIC do Commodore, mediante estes recursos extras:

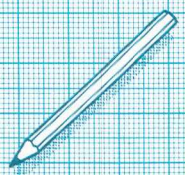
## Instrumentos da máquina

Estes comandos representam os recursos geralmente incluídos nas boas extensões do BASIC



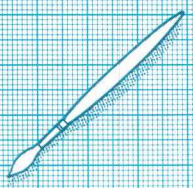
Diversos comandos para a utilização de discos são oferecidos, para formatá-los, copiar e anular unidades de arquivo e fazer cópias completas de reserva de um disco inteiro

DISK.



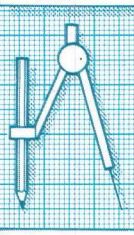
Além dos comandos para desenhar linhas retas em alta resolução gráfica, também pode haver recursos para desenho de ponto a ponto, uma forma mais fácil de construir imagens

DRAW



O comando PAINT proporciona um meio de preencher uma área definida na tela com qualquer cor. Em certas máquinas, o cursor fica posicionado no meio da figura, e o computador preenche a área, pintando do centro para fora, até encontrar uma linha sólida

PAINT



Alguns comandos gráficos do tipo CIRCLE chegam a desenhar círculos parciais (arcos) e elipses

CIRCLE



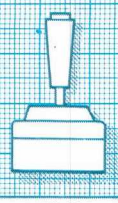
Embora as funções sonoras variem muito de uma máquina para outra, um comando MUSIC em geral oferece o recurso de tocar uma seqüência de notas predefinidas numa cadeia de dígitos

MUSIC



O diretório de um disco mantém uma lista atualizada dos nomes, tipos e tamanhos de todos os arquivos. O comando DIR proporciona a visão do diretório na tela, sem perder o programa em RAM

DIR



Em muitos computadores, o acesso aos joysticks é tarefa difícil, envolvendo os comandos PEEK e POKE. Os comandos de JOYstick colocam a posição desses instrumentos diretamente nas variáveis BASIC

JOY

1) Conjunto completo de dispositivos auxiliares de programação, incluindo funções que permitem controle extra sobre listagem de programas, correção e dispositivos auxiliares de segurança (proteção contra cópias clandestinas de seus programas).

2) Controle adicional de variáveis alfanuméricas e comandos de manipulação de texto.

3) Operadores extras de matemática e comandos para conversões numéricas.

4) Comandos simplificados para manipulação de discos.

5) Comandos de alta resolução gráfica que permitem misturar o texto com traçado de gráficos por meio de pontos e desenhar formas. Também está incluído um recurso para colorir dentro de contornos.

6) Comandos de baixa resolução gráfica e de controle de tela que podem duplicar áreas gráficas determinadas e manipular com facilidade a área da tela. Eles também permitem que o conteúdo de qualquer tela seja gravado em disco, fita ou impressora.

7) Criador e editor de sprites fáceis de usar.

8) Programação estruturada usando comandos de procedimentos — tais como PROC, CALL e EXEC; além disso, rotinas para testar loops e condições, como REPEAT-UNTIL, LOOP-EXIT, IF-END LOOP e IF-THEN-ELSE — que em geral elimina a necessidade de usar GOTOs e GOSUBs.

9) Rotinas para criação de som que permitem o aproveitamento máximo da capacidade sonora do Commodore 64, usando comandos simples de formação e execução do som.

10) Comandos simples para caneta óptica, joystick e paddle.

Poucas extensões incluem uma linha tão completa de rotinas adicionais. A maioria dos pacotes fornece recursos e comandos para uma área específica de programação. Por exemplo, o cartucho Super Expander da Commodore, para o Vic-20, fornece apenas uma linha simples de comandos para alta resolução gráfica e música. As extensões mais populares incluem dispositivos que auxiliam na construção de programas. Em geral, eles apresentam comandos de entrada por uma única tecla e diversas rotinas automáticas que simplificam a numeração de linhas, edição e correção no modo direto.

## Um prazer criativo

É comum utilitários e extensões permitirem que funções incorporadas num computador sejam alcançadas com facilidade. As rotinas que aumentam de modo significativo a capacidade de um microcomputador são difíceis de se achar, mas alguns pacotes engenhosos já estão à venda. Por exemplo, as muitas vantagens dos gráficos sprites para ação rápida em jogos de fliperama inspiraram algumas empresas a escrever pacotes criadores de sprites para computadores que não possuem essa função.

As facilidades para BASIC, as ferramentas e as extensões que destacamos representam pequena fração dos melhoramentos e dispositivos auxiliares disponíveis. Embora a atual tendência dos fabricantes seja apresentar versões completas e avançadas do BASIC, haverá sempre a necessidade de software auxiliar, para fazer da programação um prazer criativo, e não uma tarefa árdua.

# Voz de comando

**Sistemas de reconhecimento de voz são usados em aplicações comerciais e de segurança. Mas têm a capacidade limitada pelo tamanho da memória do computador.**

Para que um computador se torne de alguma utilidade, deve ter um meio funcional de permitir a introdução de comandos e informações. Em geral usamos o teclado como "interface" para nos comunicarmos com o microcomputador (o joystick e o mouse são outras alternativas possíveis). No entanto, recorrendo ao teclado, verificamos que a comunicação se dá por meio de linguagem artificial. Comandos como CLS, DIRECTORY, RUN, LOAD e SAVE podem ter significado para o sistema operacional, mas não são "naturais".

O meio natural de comunicação para os seres humanos é a fala, e não a digitação de mensagens em teclados, com respostas exibidas em telas de televisão. Se um computador pudesse entender comandos falados — ainda que tivessem a mesma forma daqueles passados por meio do teclado —, seria fácil de usar até por deficientes físicos. Mas, antes de "entender" palavras faladas, a máquina deve processar o som introduzido: os sinais analógicos precisam ser analisados e convertidos à forma digital, com a qual o computador consegue trabalhar. Apesar de parecer de fácil geração eletrônica, a fala resulta de uma combinação de sons muito complexa.

O reconhecimento imediato e completo da fala (de que era capaz, por exemplo, o computador HAL, no filme *2001 — Uma Odisséia no Espaço*) não será uma realidade por muitos anos. A máquina de escrever com entrada por comandos verbais também está distante; contudo, já existe tecnologia para ela e para o computador que "entende". Seu preço não é baixo, porque ainda se encontra grande dificuldade na criação de sistemas de reconhecimento da fala: há muitas palavras com o mesmo som cujos sentidos são bem diferentes, dependendo do contexto em que aparecem. A capacidade de processamento necessária para resolver o problema ainda não se encontra disponível a preço razoável.

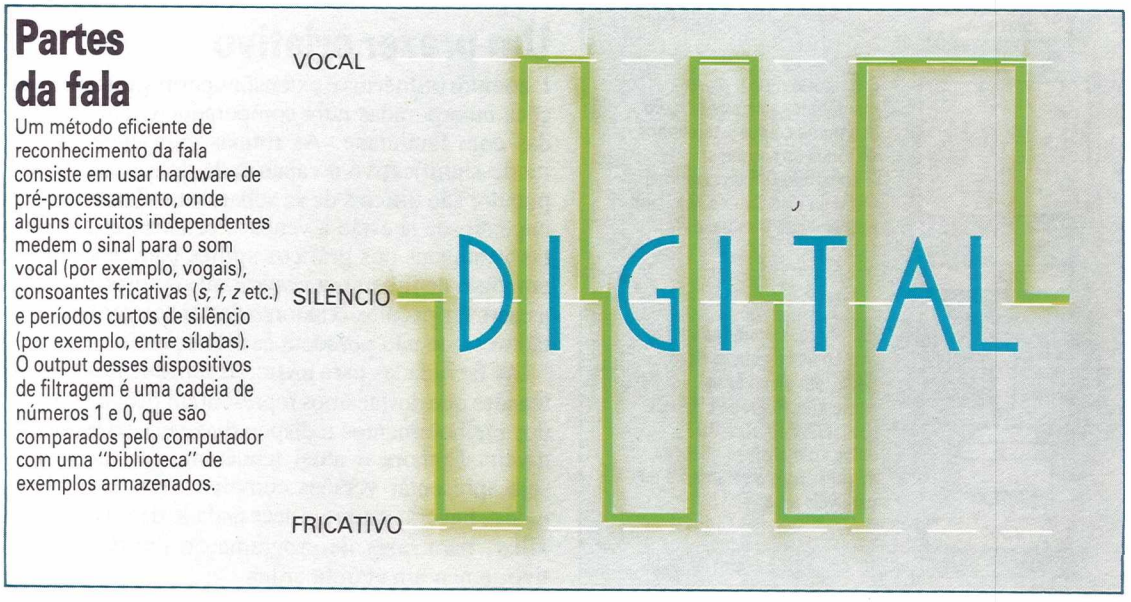
Os pesquisadores criaram sistemas que se aproximam desse objetivo, mas descobriram que, aumentando a quantidade de vozes reconhecíveis pelo computador, ocorre uma redução no número de palavras identificáveis de uma só vez. Um sistema típico para várias vozes permite o reconhecimento de vinte a trinta palavras por vez, com índice de sucesso de 85 a 90%.

## Análise da voz

Os usos potenciais desses sistemas são consideráveis. O serviço de correio alemão utiliza um deles para ajudar na separação da correspondência; há também muitas aplicações na navegação aeroespacial, tanto militar quanto civil, em que os pilotos não dispõem de mãos e pés suficientes para dar conta do recado. Em todas essas situações, o número de palavras identificáveis a qualquer tempo reduz-se a cerca de vinte. No entanto, isso não significa que todo o sistema seja limitado. O usuário seleciona uma entre vinte palavras de seu "menu", e cada comando reconhecido produz outro repertório de palavras a serem escolhidas. O computador só toma iniciativa depois de reconhecer toda a seqüência. No caso da separação de correspondência, o primeiro

### Partes da fala

Um método eficiente de reconhecimento da fala consiste em usar hardware de pré-processamento, onde alguns circuitos independentes medem o sinal para o som vocal (por exemplo, vogais), consoantes fricativas (*s, f, z* etc.) e períodos curtos de silêncio (por exemplo, entre sílabas). O output desses dispositivos de filtragem é uma cadeia de números 1 e 0, que são comparados pelo computador com uma "biblioteca" de exemplos armazenados.



nível de seleção relaciona-se com o Estado; uma vez selecionado o Estado correto, determinam-se a cidade, depois o bairro etc. Por fim, a carta é enviada ao destinatário.

O reconhecimento da fala efetua-se, em geral, de duas formas.

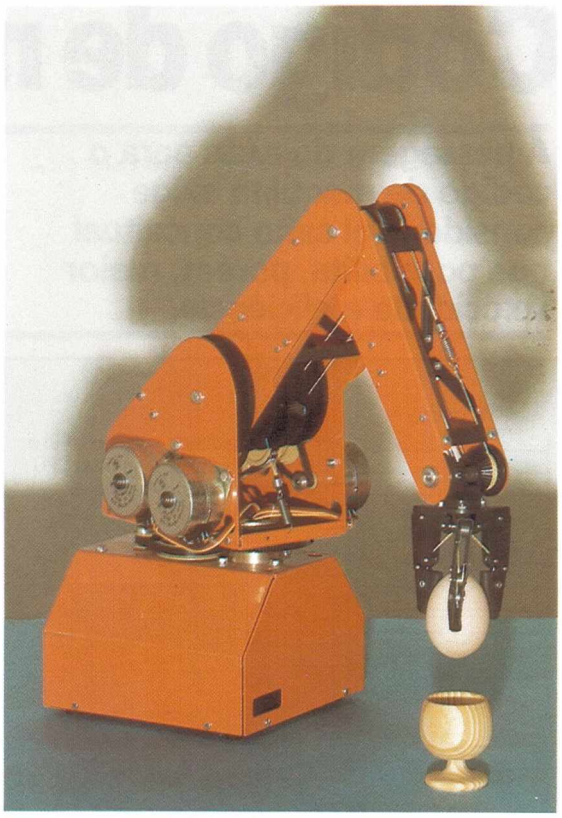
Pelo modo "rápido" ou "direto" basta introduzir toda a fala de uma só vez no conversor analógico-digital e usar a capacidade do computador para efetuar a análise. Esse método tem inconvenientes, e o maior deles corresponde ao tempo gasto em realizar a análise. Os sistemas "diretos" levam até 2 ou 3 segundos para reconhecer o input. Para que o reconhecimento da fala mostre alguma utilidade, o computador deve "entendê-la" tão depressa quanto um ser humano, e o chamado método "rápido" raras vezes surte efeito.

O outro sistema utiliza o pré-processamento. Em vez de analisar matematicamente o sinal de voz, pode-se fazer a maior parte do trabalho recorrendo à eletrônica comum. No caso, o que se libera para o computador é a informação sobre o input falado: a freqüência, a intensidade etc. Mede-se a freqüência por meio da filtragem do sinal e da detecção do nível em cada faixa, da mesma forma que se usam os controles de tonalidade num aparelho hi-fi para "destacar" o baixo. Como todo esse processo eletrônico ocorre ao mesmo tempo em que o sinal original de voz entra nos circuitos, a análise se dá de modo quase instantâneo. Para executar uma operação semelhante nos dados digitais de um conversor A/D (analógico/digital), seria necessário que vários computadores trabalhassem ao mesmo tempo com os números. Em meados de 1984, o método de pré-processamento ainda se encontrava no estágio de pesquisa — não disponível no mercado —, mas já aparentava ter grande potencial.

## Biblioteca de vozes

Uma vez que a informação sobre a freqüência, a intensidade etc. tenha sido extraída do sinal original (não importa qual o método empregado), efetua-se o reconhecimento pela comparação do conjunto de números em curso com os modelos armazenados na memória do computador. O "treino" do sistema de reconhecimento cria esses modelos: as palavras que o computador deve reconhecer são faladas para o sistema, uma a uma, e a informação resultante fica armazenada na "biblioteca" digital de exemplos. Depois diz-se de novo o conjunto completo de palavras e a máquina compara o input com seu modelo. Se combinarem, o segundo conjunto agrega-se ao primeiro para formar uma versão mais completa do modelo. Esse processo pode ser contínuo, acrescentando-se constantemente novas informações à biblioteca para um número cada vez maior de vozes diferentes.

A fim de reconhecer cada palavra falada, o computador combina o padrão de informação do input com um ou mais modelos armazenados em sua biblioteca. Em muitos casos, várias combinações possíveis serão encontradas quando partes de outras palavras combinarem com o padrão de input. As duas primeiras sílabas de "internacional", por exemplo, são as mesmas de "interpretável". No fim da busca, uma palavra se destaca por combinar melhor que qualquer outra com o modelo, e o computador a escolhe para identificar o tipo de input recebido.



**Controle ambiental**
As mais recentes aplicações do reconhecimento de voz são de natureza educativa. Uma delas é chamada "ambiente limitado", e envolve um computador, um braço de robô e alguns objetos simples, que devem ser manipulados. Falando no microfone, o usuário pode instruir o braço no sentido de "COLOCAR O OVO NO COPO". O computador terá de interpretar os comandos e verificar as posições dos objetos em sua memória.

Os recursos para reconhecimento da fala terão muitas aplicações no futuro, mas é provável que sejam usados como *front-end* para pacotes complexos de software, tais como bancos de dados, onde se selecionam os comandos de um menu na tela. Esse tipo de aplicação remove o obstáculo maior ao uso do computador por leigos: o teclado. Sistemas de viewdata, como o Videotexto da Telesp (Telecomunicações de São Paulo S/A), reduziram o dispositivo de input a simples teclado numérico, mas isso limita substancialmente a interação que o usuário pode alcançar. Uma interface que funciona por meio da fala e que reconhece um conjunto padrão de comandos para interrogar o banco de dados, além de símbolos numéricos e letras do alfabeto, torna-se um recurso importante, pois não exige muita prática com o computador.

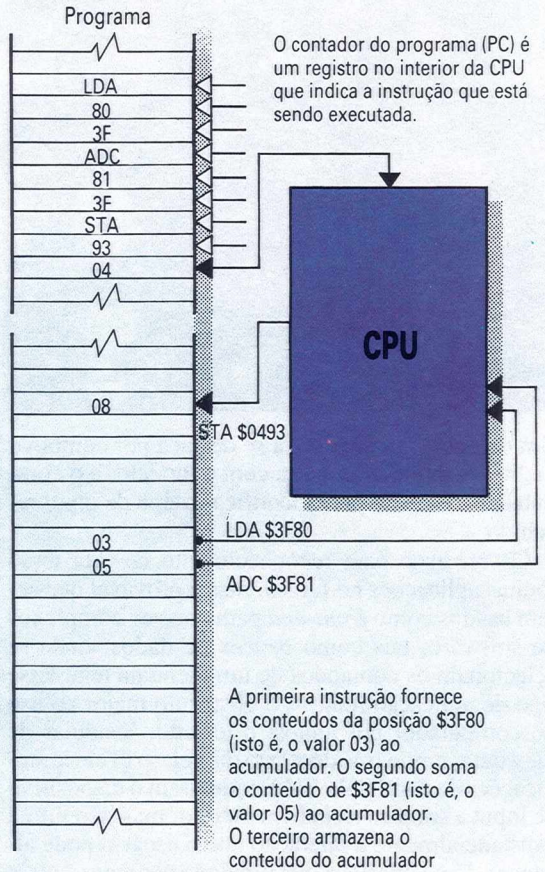
Já existem unidades de reconhecimento comercializadas que podem ser ligadas a microcomputadores, mas não são dispositivos sofisticados. Sistemas mais comuns encontrados no mercado usam muito processamento para reconhecer poucas palavras pronunciadas por uma pessoa. O que se mostra necessário antes que o reconhecimento da fala seja de fato útil é a capacidade de identificar palavras pronunciadas por *qualquer* pessoa, não importando a inflexão ou o sotaque. O fator limitante, nesta etapa, reside na quantidade de memória disponível para armazenar os modelos. O uso do videodisco para manter um conjunto padronizado de modelo constitui uma idéia interessante. Com esse esquema, seria possível gastar pouca memória interna sem redução aparente da velocidade.

# Código de máquina

**A passagem do BASIC para o código de máquina exige considerável salto conceitual que possibilita, porém, maior velocidade e eficiência.**

## Um passo por vez

O programa em linguagem de máquina é armazenado numa área da memória, embora os dados que ele opera possam estar em alguma outra posição. Observe que os operandos (por exemplo, $3F80) estão armazenados em 2 bytes, com o byte menor ($80) situado antes do maior ($3F).




Todos os programas em código de máquina são constituídos por operações simples que transportam bytes de memória para os registros internos da CPU, realizam seu processamento e, a seguir, os remetem de novo a uma posição da memória. Esse diagrama mostra o programa necessário para somar os conteúdos de duas posições e armazenar o resultado numa terceira.

Até este ponto, todo o nosso trabalho de programação centrou-se na linguagem BASIC, versátil e de uso fácil. No entanto, à medida que sua experiência aumenta e os projetos de programação com que você lida se tornam mais audaciosos, evidenciam-se as limitações dessa linguagem. Você descobrirá que os gráficos não podem ser deslocados por toda a tela tão depressa quanto deseja e que é necessário apelar com freqüência para os complexos comandos PEEK e POKE a fim de aproveitar melhor os recursos de sua máquina.

Em contrapartida, a programação em código de máquina impõe pouquíssimas restrições e, em comparação com o BASIC, dá impressão de velocidade enorme. Mas poucos usuários de microcomputador optam por esse recurso, em parte porque exige um processo de programação muito laborioso e também por ser conceitualmente diferente do BASIC ou de outra linguagem de alto nível. Todavia, não se pode deixar de ter uma idéia do funcionamento do código de máquina e neste artigo, o primeiro de uma série de dois, examinaremos os procedimentos fundamentais necessários a sua utilização.

O código de máquina, conforme já vimos antes, é a linguagem entendida pelo microprocessador (a CPU), que constitui o cerne do computador e pode apenas executar funções muito simples (adiciona dois dígitos a um número, por exemplo, mas não os multiplica). Faz isso, no entanto, a velocidades muito altas. Cada operação do microprocessador é especificada de acordo com o número de "ciclos de relógio" empregados. Se a CPU em seu computador funcionar a 1 MHz, o ciclo de relógio será de 1 microssegundo, e uma operação que requer quatro "ciclos de relógio" será realizada em 4 milionésimos de segundo.

Como conseqüência, um programa desenvolvido em código de máquina vai requerer grande quantidade de instruções e qualquer função deverá ser elaborada "a mão", a partir de operações simples. Toda a programação consistirá na manipulação de bits ou bytes isolados de memória, empregando-se funções lógicas simples como AND, OR e NOT, além de aritmética binária elementar.

Esse é um dos motivos por que o desenvolvimento de programas nessa linguagem se torna uma tarefa lenta; o outro reside no fato de o programador ser obrigado a saber a localização de tudo que está armazenado na memória. Em BASIC, sempre que se encontra uma instrução como LET A = 5, é função do interpretador achar um espaço na memória para armazenar a variável. Além disso, sempre que A for referida de novo no programa o interpretador recordará o local onde devem ser procurados os dados necessários. Ao iniciar a programação em código de máquina, você descobre que tem de especificar um endereço (uma posição de memória) para cada conjunto de dados a armazenar. E cabe também a você garantir que não haja superposição acidental com outros conjuntos de dados.

Examinemos em que consiste o código de máquina. (Todos os exemplos serão relacionados a CPUs de 8 bits de capacidade, como a do Z80 e do 6502; dispositivos de 16 bits operam de modo semelhante, mas processam o dobro de bits em cada operação.) O microprocessador conecta-se à memória do computador por meio de dois busses (vias): o bus de endereços e o de dados (ver p. 144). Há também um elemento denominado bus de controle que fornece apenas sinais de cronometragem à CPU e não é utilizado pelo programador.

O bus de endereços tem capacidade de 16 bits e a atribuição de um padrão de bits a esse bus possibilita à CPU selecionar qualquer dos 65.536 bytes em seu "mapa de memória" (ver p. 329). Em microcomputadores comuns, algumas dessas posições estarão na RAM, outras na ROM, algumas nos chips especiais de entrada/saída e ainda existirão as que não serão utilizadas. Se a CPU quiser ler determinada posição de memória (uma das linhas no bus de controle indica se deve haver leitura ou registro), o byte sele-

cionado coloca seus conteúdos no bus de dados, na forma de um padrão de 8 bits. De modo semelhante, a CPU pode registrar um padrão de 8 bits em qualquer posição escolhida. A CPU não sabe quais as partes da memória em que estão a ROM e a RAM; desse modo, determinar o endereço correto é outra responsabilidade importante do programador.

No interior do microprocessador, há talvez meia dúzia de "registros", semelhantes a posições individuais de memória, utilizados para o armazenamento de resultados temporários e execução de funções aritméticas lógicas e binárias. A maior parte desses registros corresponde a 1 byte de memória, embora alguns tenham 16 bytes de capacidade. Um dos tipos mais recentes de registro é o chamado contador do programa (PC, Program Counter), que possui na memória o endereço da instrução em código de máquina em execução. Você pode imaginá-lo como semelhante ao número de linha num programa em BASIC.

Outro registro muito importante (mas, agora, com apenas 8 bits de extensão) é o "acumulador" Como o nome indica, esse registro acumula totais (isto é, bytes que podem se somar ou subtrair). Na verdade, esse é, em geral, o único registro que pode executar qualquer tipo de procedimento aritmético. Assim, um código de máquina muito simples seria especificado da seguinte forma:

1) Carregar o acumulador com os conteúdos da posição de memória $3F80. Os endereços em código de máquina são quase sempre desenvolvidos em hexadecimais (ver p. 179), indicados pela anteposição de um sinal especial, quase sempre o $.

2) Acrescentar ao acumulador os conteúdos da posição de memória $3F81, levando em conta que o resultado pode ser maior do que a capacidade de armazenamento num único byte — neste caso também haverá um "bit de transporte".

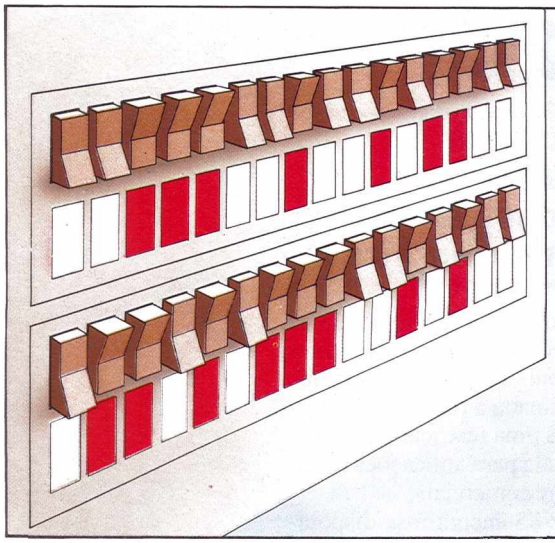
3) Armazenar os novos conteúdos do acumulador (isto é, o resultado) na posição de memória $0493.

Cada um desses procedimentos representa uma instrução em código de máquina e o programa normalmente seria desenvolvido da seguinte forma:

```
LDA $3F80 (LoaD Accumulator, carregar o
          acumulador)
ADC $3F81 (ADd with Carry, somar com transporte)
STA $0493 (STore Accumulator, armazenar o
          acumulador)
```

Os comentários entre parênteses, da mesma forma que as instruções REM do BASIC, não resultam em operações. A primeira entrada em cada linha é denominada "opcode" (código de operação), e indica o tipo de operação. A segunda coluna apresenta o "operand" (operando), ou seja, os detalhes ou as posições dos dados que serão operados. O microprocessador costuma apresentar várias dezenas de códigos de operação (isto é, pode realizar várias dezenas de tipos de operações simples) e cada código de operação fornecido à máquina ocupa apenas 1 byte de memória.

O código de operação pode, desse modo, ser especificado por um número entre 0 e 255 (ou, mais adequadamente, na base hexadecimal de $00 a $FF). Todavia, enquanto o programa é desenvolvido, costuma-se tornar a listagem mais legível pelo emprego de três letras mnemônicas, como LDA, ADC, STA.



**Luzes cintilantes**
A imagem de grandes painéis luminosos vistos nos computadores dos filmes veio do "painel frontal" existente em muitos mínis. Esse painel consistia numa linha de luzes e interruptores representando os busses de endereços e de dados da CPU. Antes da introdução dos teclados todos os programas em código de máquina tinham de ser fornecidos sob a forma binária por esse procedimento.

Cada operando apresentado consiste num número hexadecimal de $0000 a $FFFF e utiliza até 2 bytes da memória do programa. No entanto, alguns operandos têm apenas 1 byte de comprimento e alguns códigos de operação não possuem quaisquer operandos. O pequeno programa que apresentamos deve ocupar um total de apenas 9 bytes, sem incluir as três posições de memória ($3F80, $3F81 e $0493), com as quais o operará. Para esse exercício simples, o seguinte programa em BASIC chega ao mesmo resultado, mas ocupa quase 50 bytes e realiza a operação pelo menos cem vezes mais lentamente, devido ao tempo gasto pelo interpretador para traduzi-la:

```
10 A = PEEK (16256)
20 A = A + PEEK (16257)
30 POKE 1171,A
```

Observação: As posições utilizadas neste programa podem não ser adequadas a sua máquina.

No próximo fascículo examinaremos o procedimento para fornecer o código de máquina ao microcomputador e processá-lo, bem como os diferentes modos de apresentar a linguagem.

## LDA

**LoaD Accumulator (carregar o acumulador)**
Transfere o conteúdo de uma posição isolada de memória (byte) para o registro do acumulador interno.

## STA

**STore Accumulator (armazenar o acumulador)**
Executa o procedimento oposto ao do LDA.

## ADC

**ADd with Carry (soma com transporte)**
Soma os conteúdos de uma posição de memória aos que estão no acumulador, criando 1 bit de transporte.

## SBC

**SuBtract with Carry (subtração com transporte)**
Realiza função inversa à do ADC.

## JMP

**JuMP (salto)**
Transfere a operação do programa para uma nova posição. É semelhante, quanto ao modo de operar, ao GOTO do BASIC.

**Códigos de operação**
Estes são apenas alguns dos opcodes (códigos de operação ou de instrução) que um microprocessador comum pode executar.

# HP-85B

## Um microcomputador compacto, portátil e bem documentado, ideal para aplicações técnico-científicas.

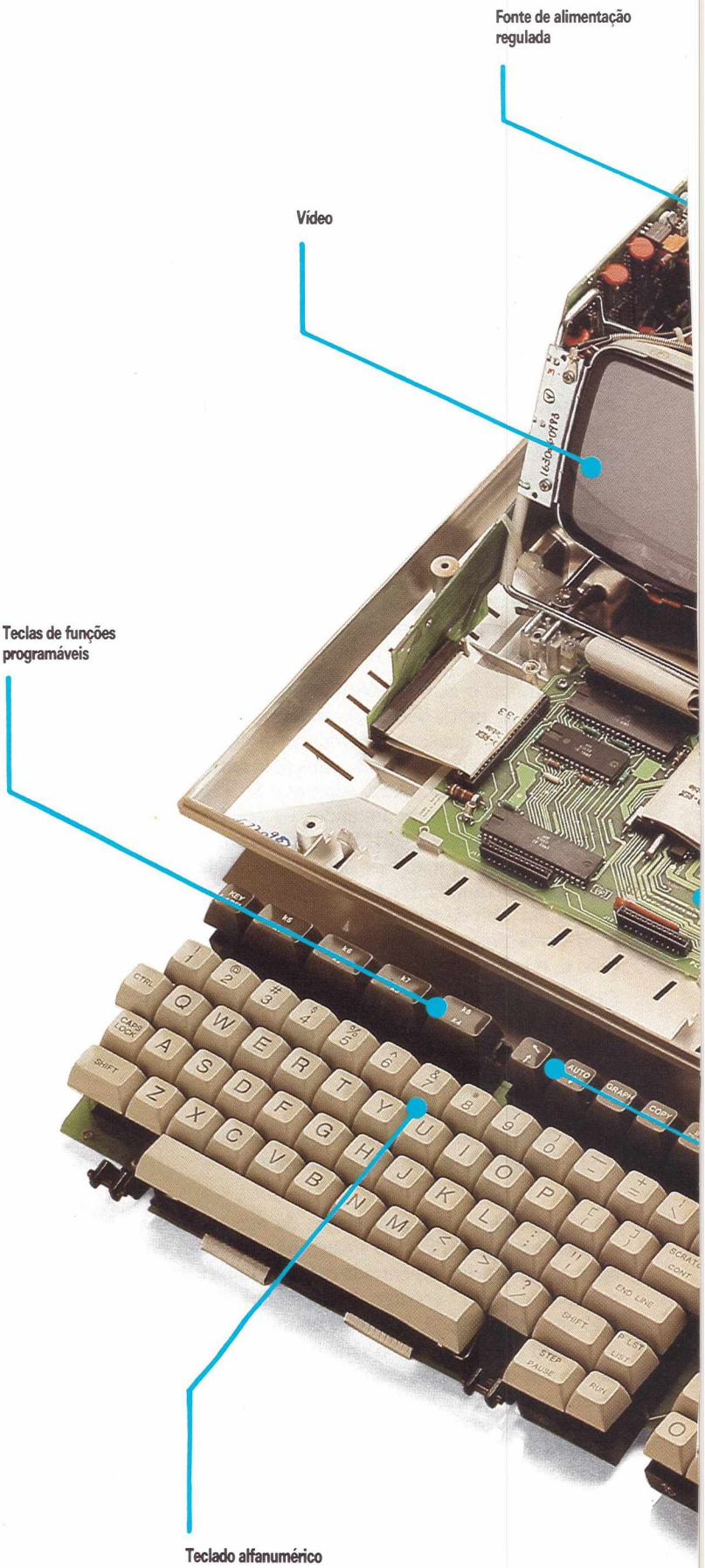
A Hewlett Packard, com o seu HP-85B, é a única empresa estrangeira autorizada pelo governo brasileiro a vender microcomputadores no país, porque há reserva de mercado destinada a proteger a indústria nacional. Existe apenas uma restrição: a máquina só pode ser comercializada para aplicações técnico-científicas, não para fins comerciais.

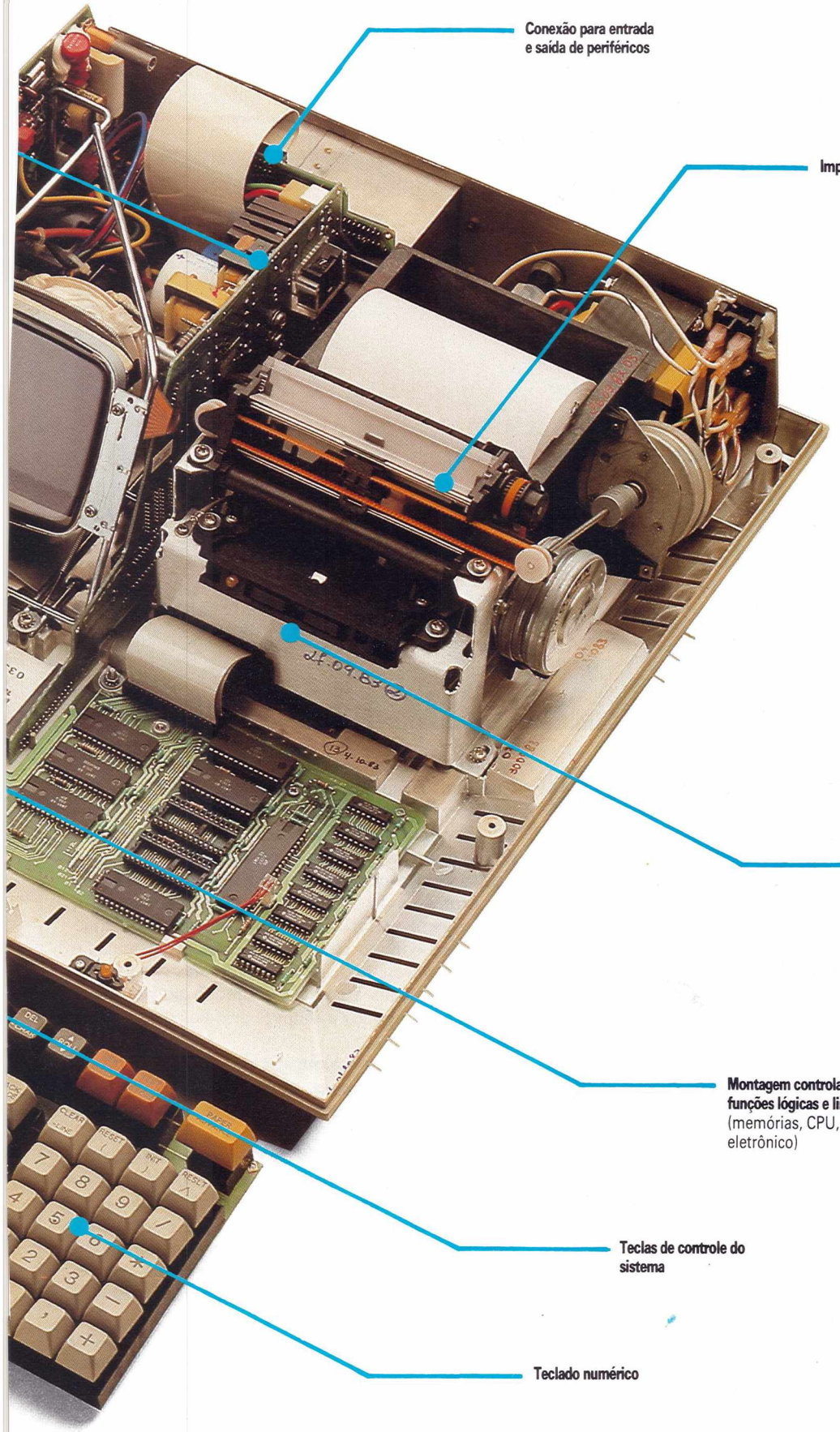
Em conseqüência, o HP-85 encontra-se disponível no mercado em sua configuração básica, sem saídas para unidade de vídeo maior, impressora de impacto ou unidade de disquete. Seu teclado, uma impressorazinha térmica, um pequeno vídeo e a unidade de fita magnética estão todos embutidos em um único gabinete.

Portátil, este micro, pesando apenas 9 kg, tem o tamanho aproximado de uma máquina de escrever elétrica. Sua tela de duplo modo torna as operações simples. O modo alfanumérico exibe os programas, dados, mensagens de erro, comentários da máquina e resultados. O modo gráfico permite ver as informações gráficas separadas das alfanuméricas. A troca se faz mediante um só comando. Cartuchos de fita magnética de alta densidade e capacidade são usados para armazenamento temporário ou definitivo de programas e dados. Cada um dos cartuchos pode manter até 42 arquivos separados (195 K de programas e 210 K de dados). Um simples comando indica o nome, tipo e tamanho de cada arquivo.

Embutido no gabinete, há também um disco eletrônico — um espaço da memória interna de 32 K — que atua como unidade de discos de alta velocidade. Nele, pode-se armazenar programas e dados temporariamente e recuperá-los com velocidade até 150 vezes maior que a da fita magnética. E essa capacidade é expansível a 160 K, por meio de um módulo de memória externa. O sistema operacional e a linguagem BASIC, incluindo o sistema avançado de gráficos, estão permanentemente armazenados dentro da máquina.








## HP-85B

**MICROPROCESSADOR**

HP

**CLOCK**

6 MHz

**MEMÓRIAS**

ROM padrão de 48 K e máxima de 96 K.
RAM do usuário de 32 K.
RAM no disco eletrônico padrão de 32 K e máxima de 160 K.
RAM de tela de 8 K.

**TECLADO**

Alfanumérico e numérico separados. Teclado de funções definíveis pelo usuário.

**PERIFÉRICOS**

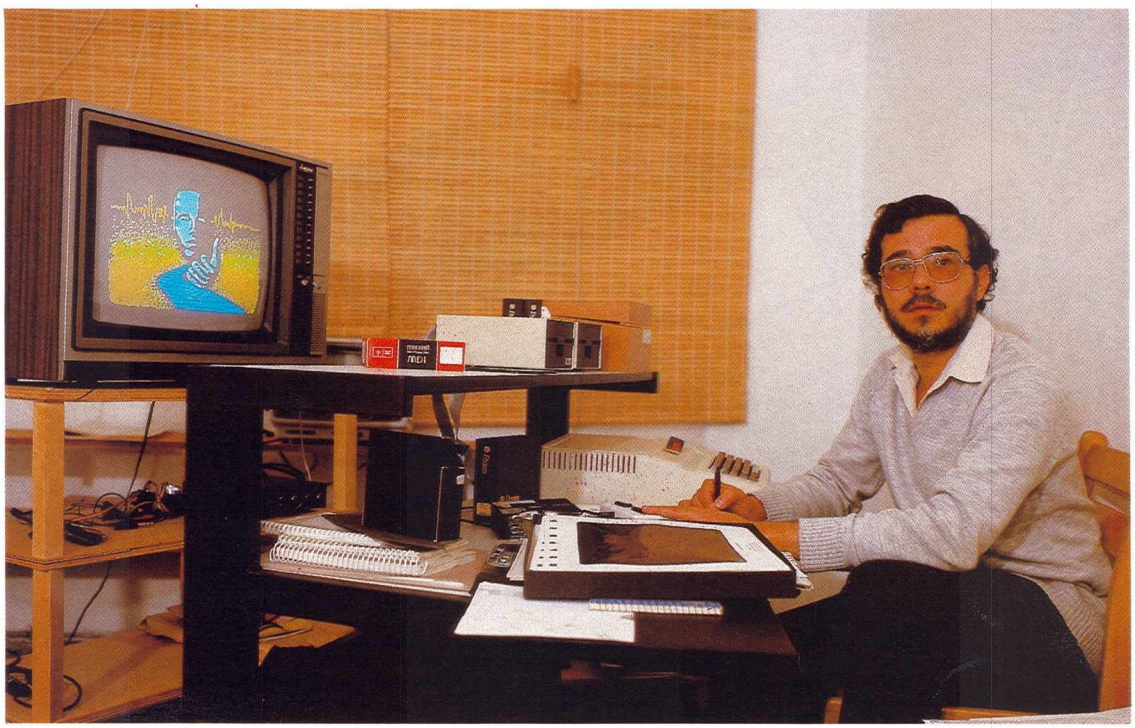
Plotter, unidade de disco, impressora serial de agulha. Todos necessitam de aprovação da Secretaria Especial de Informática (SEI).

**DOCUMENTAÇÃO**

Manual do usuário; conjunto de programas básicos; guia de consulta rápida; e cartão de referência básica.

# Micro e arte

**Pranchetas digitalizadoras, canetas ópticas, plotters e vídeos de terminais substituem a tinta, o pincel e as telas na arte computadorizada.**






**Desenho animado**
Eliandro Martins de Moraes, da ART Sistemas, de São Paulo, utiliza o micro para fazer animação de imagens. Primeiro, elabora e pinta quadro por quadro. Depois, com dois programas por ele desenvolvidos, organiza a seqüência e comanda os movimentos.

Sobretudo a partir do século XIX, as artes plásticas passaram por transformações radicais. Abandonou-se a idéia de que a pintura e a escultura destinavam-se a copiar formas colhidas diretamente do mundo exterior e a reproduzi-las da maneira mais fiel possível. Novas técnicas e novos materiais passaram a ser empregados. Modernamente, um quadro não precisa apresentar relação com óleo sobre tela emoldurada, representando, por exemplo, um nu feminino ou uma natureza-morta.

A informática também passou a representar um instrumento do artista, que dela tem lançado mão de diversas formas. Sua primeira utilização nesse campo foi desenvolvida, no Brasil, por Waldemar Cordeiro. Designer, artista plástico, arquiteto, paisagista e crítico de arte, Cordeiro nasceu em Roma (1925) e faleceu em São Paulo (1973). Anos antes de sua morte já utilizava o computador como instrumento para a criação de trabalhos arquitetônicos. Foi professor da Universidade de Campinas (SP), onde organizou um centro de processamento de imagens. Em contato com engenheiros de sistemas e programadores, pôde desenvolver suas pesquisas e executar trabalhos sobre o emprego de processos estocásticos ligados a outputs gráficos com aplicações cromáticas.

O grafismo por computador de Cordeiro resultava de extensos programas, cujas instruções definiam formas, traços e ângulos, após passarem por uma impressora.

Hoje já é possível utilizar o vídeo do micro para executar a animação de um desenho. Diz Fernando Albuquerque Lins, diretor da Link Informática (empresa que atua nas áreas de hardware e software), que há algum tempo os trabalhos gráficos eram feitos pelo pessoal de processamento de dados, que desenvolvia projetos em computadores de grande porte, com vídeos coloridos.

Utiliza-se também o computador para a criação artística por meio de pranchetas digitalizadoras (normalmente empregadas em desenho técnico), de plotters e do próprio vídeo do terminal.

A partir de 1980, com a explosão do uso dos mi-

cros no mercado nacional — inclusive os coloridos —, cresceu o interesse pela realização de trabalhos visuais na própria tela, ampliando-se o universo de usuários.

Diretor da ART Sistemas, softhouse que vem se consolidando no mercado como produtora de figuras animadas por microcomputador, Eliandro Martins de Moraes utiliza um software importado — acompanha o aplicativo Koala, um estojo digitalizador PAD, no qual se traça o desenho utilizando caneta óptica. Há um problema com o estojo sobre o qual a caneta desliza: como acontece com as pranchetas digitalizadoras, o usuário não trabalha diretamente no desenho. Os traços que ele executa no estojo formam a imagem que aparece no vídeo.

Com o Koala, Eliandro elabora quadro por quadro. Pinta, faz as alterações necessárias no desenho e depois armazena os dados relativos a cada quadro. Para a animação, ele desenvolveu dois programas. O primeiro organiza em seqüência os quadros na tela e o outro comanda os movimentos de cada um. Gravam-se na memória apenas as diferenças entre um quadro e o seguinte, pois de outra forma ela seria insuficiente para realizar a animação. A montagem dos desenhos, por sua vez, é realizada pela utilização de Assembler, pois uma linguagem de alto nível não alcançaria a velocidade necessária para a movimentação dos quadros.

Num disquete de 5 1/4 polegadas estão gravados 30 segundos de animação, com cerca de cem quadros relativamente diferentes entre si. A velocidade com que se passa de um quadro para outro depende sempre do número de alterações.

Sendo inviável trabalhar com o som proveniente do próprio micro, utilizam-se os recursos tradicionais de áudio. Depois a gravação é feita em videocassete.

Vice-presidente da área de informações e microinformática do Citybank e fotógrafo há vários anos, Mecenas Salles possui uma coleção de cerca de quarenta programas gráficos. Ao mesmo tempo, modifica os algoritmos criando comandos especiais para sofisticar seus desenhos, além de misturar cores e traços para caracterizar melhor seu estilo. Muitos de seus trabalhos são feitos em cima de fotos tiradas por ele mesmo. Assim, o micro serviu também à organização e catalogação de seu arquivo de 20.000 fotos.

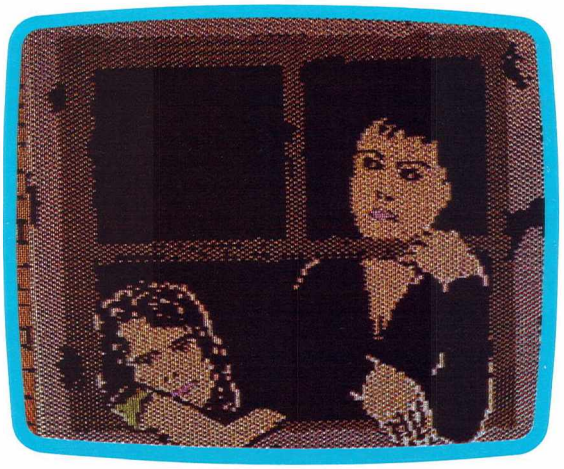
Eliandro e Mecenas começaram a fazer arte com o auxílio do micro por *hobby* e agora trabalham também para terceiros. Na opinião de ambos, convém utilizar um micro de 16 bits, que proporciona melhor resolução de imagem. Dessa forma, a semelhança entre a tela e um trabalho de tapeçaria deve desaparecer. Além disso, um micro de 8 bits possui apenas seis cores básicas (verde, magenta, branco, preto, laranja e azul), ao passo que um de 16 bits oferece variações de mais de 250 cores.

Mecenas pretende desenvolver seu trabalho pesquisando formas tridimensionais, uma especialidade da arquiteta, escultora e professora Vera Pallamin. Programas realizados por Vera em seu HP-85 permitem a elaboração de perspectivas de edificações. O programa é sempre aberto de maneira que ofereça várias opções para a visualização da organização do espaço.

A artista ressalta que, para seu trabalho, o conhecimento de matemática não é suficiente. Os cálculos se fazem necessários para a elaboração do programa — na determinação de formas em três dimensões, por exemplo. No entanto, é preciso um critério para hierarquizar e atribuir o valor devido a cada forma que o micro possa apresentar. Convém ainda avaliar a organização do espaço, tendo em vista a utilização pretendida.

Tais questões fogem ao campo específico da matemática, exigindo o conhecimento de outras disciplinas, inclusive a estética.

O desenho de perspectivas, em arquitetura, sempre exigiu enorme gasto de tempo e de esforço. Isso dificultava também a elaboração de diversas pranchas referentes ao mesmo espaço arquitetônico (para que, entre elas, fossem escolhidas as que melhor pudessem representá-lo tridimensionalmente). Também nesse campo o micro significou enorme ganho de velocidade, pois executa em segundos um desenho que antes demandava quase quatro dias para ser concluído.



**Pintando no vídeo**
Executivo do Citybank e fotógrafo há vários anos, Mecenas Salles criou comandos especiais para sofisticar seus desenhos. A maior parte dos trabalhos que desenvolveu foi feita em cima de fotos por ele mesmo tiradas.



**Projeto arquitetônico**
Depois de desenvolver em seu micro as duas formas impressas no papel (na foto), a arquiteta Vera Pallamin, com um visor estereoscópico, usou-as para montar o projeto tridimensional de um grande teatro.

# Descubra o código

**A elaboração de mensagens cifradas foi uma das primeiras aplicações dos computadores. Hoje, a criação e a decifração de códigos simples fazem parte das habilidades do programador BASIC.**

Todas as nossas comunicações com os outros são codificadas. Tanto a fala como a linguagem escrita são apresentadas de forma inteligível, se a pessoa receptora da mensagem for capaz de interpretar o código. Isso também vale em nossa "conversa" com os computadores. A maioria dos micros utiliza um dialeto de BASIC acessível ao usuário comum. Mas a máquina em si não recorre a essa linguagem para executar suas funções: deve primeiro converter as instruções BASIC na forma puramente numérica que pode então usar para estabelecer as seqüências de comutações definidas no programa. Assim, produz os resultados desejados. Códigos desse tipo — linguagens humanas e de programação — são bem acessíveis em nossa vida cotidiana. Qualquer pessoa aprende francês, alemão, BASIC ou FORTRAN, desde que empregue esforço e vontade.

## Compressão de dados

O usuário de computador que precisa armazenar grande quantidade de arquivos de texto está constantemente buscando métodos de comprimir os dados. Um modo de obter esse resultado é a simbolização. O símbolo pode ser substituído por uma palavra ou chave freqüentemente utilizada, de modo semelhante à maneira como os computadores TK85 ou CP 200 apresentam uma palavra em BASIC ao toque de uma única tecla.

Além disso, as técnicas de codificação são usadas para comprimir mais ainda os dados. Compact, um utilitário do sistema operacional Unix, é considerado capaz de comprimir arquivos de texto a uma média de 38%, e Clip, que processa sob o CP/M, obtém resultados ainda melhores. Compactor, que processa no Commodore 64, realiza a mesma função nos programas em BASIC, pela eliminação de todas as instruções REM, espaços desnecessários etc.

Mas há outro tipo de codificação (mais precisamente denominado criptografia), com objetivo oposto ao da comunicação: negar compreensão a todos, menos ao pequeno grupo ao qual a comunicação se destina. Antes da segunda metade do século XX, a transmissão de dados de forma não inteligível a todos restringia-se a governos e algumas empresas de grande porte. Porém, com o intenso emprego das linhas públicas de telefones — vulneráveis à escuta — para troca de informações com valor comercial, o uso da criptografia tornou-se mais comum.

As cifras e códigos variam do muito simples — a soma ou subtração de determinado valor para cada byte, por exemplo, ou a substituição formatada de um caractere por outro — até os esquemas complexos que estão sendo elaborados de acordo com os mais recentes progressos da teoria dos números. Essas cifras não contêm quaisquer elementos de repetição e, por isso, não são vulneráveis a métodos de decodificação de análise de freqüência.

A mais simples de todas as técnicas de criptografia de importância significativa é a Cifra de César (com certeza utilizada na época do Império Romano). A cifragem do código de César exige que se tenha apenas a mensagem e conhecimento do código; desse modo, não são necessários livros de código volumosos ou documentos a serem ocultos, nem máquinas sofisticadas. Eis uma mensagem simples, cifrada segundo essa técnica:

KGAPMAMKNSRYBMP ASPQM ZYQGAM

Podemos fazer algumas conjeturas sobre essas palavras codificadas a partir do modo pelo qual os grupos cifrados estão espacejados (embora isso pudesse ter sido feito apenas para nos confundir). A primeira coisa que salta aos olhos é o fato de que a mensagem consiste em três palavras: a primeira tem quinze letras; a segunda, cinco; e a terceira, seis. Também podemos admitir que a segunda palavra e a terceira terminam com a mesma letra. A letra final comum aqui (M) é também uma das duas letras na mensagem com ocorrência maior (a outra é A). Essa observação é de grande valor para o decifrador, desde que ele saiba com que língua está trabalhando.

Com uma amostra de tão poucos elementos como a nossa (um total de apenas 26 letras, que qualquer estatístico considerará uma amostragem insuficiente para basear a análise), nossos resultados provavelmente serão falhos. Mesmo assim, tentaremos uma substituição por freqüência para ver se os resultados têm algum significado. Substituímos primeiramente o M pela letra O:

KGAPoAoKNSRYBoP ASPQo ZYQGAo

A mensagem ainda não apresenta nenhum significado, mas há outras pistas. Qual a relação entre a letra original e aquela pela qual nós a substituímos? M está situada duas posições antes do O no alfabeto. O que acontecerá se fizermos a mesma substituição no restante da mensagem? Duas posições após o A está C. Assim, tentemos acrescentar esse dado:

KGcPocoKNSRYBoP cSPQo ZYQGco

Na terceira palavra temos agora um final com *co* que é uma construção válida em português. A segunda palavra começa com *c* e termina com *o*, que também pode nos indicar algo viável; desse modo, talvez estejamos na direção certa. Façamos o mesmo com o restante da mensagem. A letra situada duas posições

depois de K é M; duas após G é I; e assim a primeira palavra poderá ser MICRO...

A Cifra de César é, portanto, um código de substituição que se apóia no ''deslocamento'' do alfabeto, para diante ou para trás, num certo número de posições, com o objetivo de determinar o novo valor de cada caractere. Pode ser refinada depois pelo emprego de uma chave de transformação — 24225, por exemplo. Neste caso, a primeira letra estará deslocada duas posições, a segunda, quatro, a terceira, duas e assim sucessivamente. Quando o final da chave é atingido, retornamos ao início. Com o emprego dessa chave de transformação em nossa mensagem, o exemplo seria:

KEAPJAKKNPRWBMM AQPQJ ZWQGXM

Neste exemplo, a análise de freqüência será inútil, pois não há uniformidade na substituição — uma letra terá substitutas diferentes, de acordo com sua posição na mensagem completa. Outra cifragem independente torna a mensagem assim:

MOPDCOSIRCMUAO US AIOCOTRRBC

Examinando-a atentamente, verificamos que esta série de caracteres corresponde a um anagrama de MICROCOMPUTADOR — CURSO BÁSICO, com os dois espaços entre as palavras. Aqui, simplesmente procuramos determinar o algoritmo de codificação, dando amostras do texto cifrado e do não-cifrado — um procedimento bastante comum. Se a cifra tiver de ser compreensível ao receptor da mensagem, a combinação das letras deve ser de algum modo previsível. Esta cifra, em particular, conhecida como Cerca de Barras, por razões que logo se evidenciarão, também exige o conhecimento da chave pelo decodificador — e, neste caso, ela é 3. Tomemos os sete primeiros caracteres e vamos escrevê-los com três espaços entre si:

M***O***P***D***C***O***S

Você consegue perceber algo? Bem, então tente: escreva o texto não-codificado em três linhas, colocando letras acima e abaixo da linha central, da seguinte forma:

M O P D C O S
I R C M U A O
– U S – A I O
C O T R R B C

O sinal de subtração representa o espaço entre as palavras e o método de codificação é evidente.

Os exemplos citados até aqui são todos de cifras, entendidas como um método de escrita secreta que emprega a substituição ou a transformação de letras, de acordo com uma chave. Os códigos são diferentes porque tendem a substituir blocos inteiros por outros, em geral menores (possibilitando ainda condensar os dados). Mas apresentam um inconveniente: o receptor e o transmissor devem dispor de um livro de código antes que as mensagens sejam comunicadas. Um exemplo dessa técnica utiliza um romance, jornal ou qualquer outro texto acessível a ambos e indica *as palavras* que constituirão a mensagem apenas dando o número de seqüência em que ocorrem. Um texto como ''João gosta de brincar com seu microcomputador. Sua irmã prefere o curso de inglês. A mãe acha que o básico é que eles estudem.'' poderia ser a chave para o código 7, 5, 8. Talvez você possa deduzir a mensagem...

Computadores de qualquer tipo são de grande valor ao se tentar cifrar ou decifrar mensagens em código. Um requisito primário da Cifra de César, por exemplo, é a capacidade de deslocamento por uma série alfanumérica, acrescentando ou subtraindo uma constante ao valor ASCII de cada caractere a ser impresso. Pode-se alterar essa constante quando o programa for processado. O alfabeto, então, se inverte (onde a chave é 1, A resulta em Z).

Descobrindo a constante, você decodifica esta mensagem:

EXI E TVSBMQE!

## A Cifra de César

Este programa (desenvolvido em BASIC) codifica textos pela Cifra de César com o emprego de uma chave múltipla de cinco elementos. A mensagem aparece em texto normal, à medida que é fornecida e, ao ser pressionada a tecla RETURN, imprime-se a versão cifrada. A mensagem deve ser fornecida sem espaços nem pontuação.

```
10 INPUT "FORNECER UMA CHAVE DE CINCO
   ALGARISMOS";K$
20 INPUT "FORNECER A MENSAGEM";M$
30 FOR I=1 TO LEN(M$)
40 LET J=I — INT(I/5)*5+1
50 REM ***RODA O TEXTO POR MEIO DA CHAVE***
60 LET M=ASC(MID$(M$,I,1)) — VAL(MID$(K$,J,1))
70 IF M<65 THEN LET M=M+26
80 PRINT CHR$(M);
90 NEXT I
```




**Criptoanálise**
Uma das primeiras aplicações dos computadores foi a decifração de códigos muito complexos de substituição de múltiplas chaves utilizados na Segunda Guerra Mundial. Os alemães haviam desenvolvido a máquina chamada ENIGMA, que gerava suas próprias cifras. Os criptogramas resultantes fizeram com que os Aliados se empenhassem em sua interpretação. O êxito foi afinal obtido pelo grupo Colossus, na Inglaterra, do qual Alan Turing foi destacado membro.

# Questão de estilo

**Conhecidas as regras fundamentais da linguagem BASIC, podemos nos deter agora no estilo de programação e em novos comandos que aperfeiçoarão nossa técnica.**

O programa da agenda de endereços computadorizada, desenvolvido nos fascículos anteriores, emprega muitos recursos importantes da linguagem BASIC, mas não todos. Agora, na conclusão deste curso, examinaremos até onde o BASIC pode levá-lo, se você quer se tornar um programador de nível adiantado.

## Linguagem de máquina

A maioria das versões do BASIC permite incluir como parte do programa rotinas desenvolvidas em código de máquina. De maneira geral, há dois métodos de fazer isso. O mais simples consiste em utilizar os comandos PEEK e POKE. PEEK é uma instrução empregada para examinar endereços específicos de memória. Por exemplo, LET X = PEEK(1000) faz com que o valor armazenado na posição de endereço 1000 seja atribuído à variável X. A execução de PRINT X imprime, então, o valor que estava (e ainda deve estar) na posição 1000. O pequeno programa que apresentamos a seguir examina os conteúdos de dezesseis posições de memória, mediante o comando PEEK, e as imprime na tela:

```
10 INPUT "FORNECER ENDERECO DE INICIO PARA
   'PEEK'"; S
20 PRINT
30 FOR L = 1 TO 16
40 LET A = PEEK(S)
50 PRINT "A POSICAO"; S; "CONTEM: ";A
60 LET S = S + 1
70  NEXT L
80 PRINT "PRESSIONAR A BARRA DE ESPACO
   PARA EXAMINAR"
90 PRINT "AS PROXIMAS 16 POSICOES OU
   RETURN PARA TERMINAR"
100 FOR I = 1 TO 1
110 LET C$ = INKEY$
120 IF C$ < > CHR$(13) AND C$ < > " " THEN I = 0
130 NEXT I
140 IF C$ = CHR$(13) THEN GOTO 160
150 GOTO 30
160 END
```

O loop nas linhas de 100 a 130 verifica a entrada a partir do teclado e em seguida vai para o fim do programa, se o caractere digitado foi RETURN (13, em ASCII), ou volta ao início, saltando a instrução INPUT.

O caractere ASCII da posição de memória também pode ser impresso usando PRINT CHR$(A). Cuidado, porém, pois os valores ASCII menores que o decimal 32 (ASCII para o caractere de "espaço") não estão definidos de modo uniforme. Todos os valores ASCII de 0 a 31 representam caracteres não imprimíveis ou funções especiais, como controles de cursor. Um dos raros padrões comuns dos fabricantes de computadores consiste em ASCII 13 corresponder ao retorno do carro (RETURN) e em ASCII 7 produzir som ou um "bip" no alto-falante interno.

O comando POKE é o inverso de PEEK. Permite que você registre qualquer valor, de 0 até 255, em alguma posição da RAM. No entanto, deve-se empregar esse recurso com cautela, pois o registro numa parte da memória já utilizada pelo programa causa às vezes resultados catastróficos. As rotinas escritas em código de máquina são armazenadas por meio do comando POKE no endereço adequado e recuperadas, no processamento, pela instrução CALL.

O método para desenvolver programas em código de máquina vai além dos objetivos deste curso de BASIC. Basta lembrar que o código de máquina opera de maneira muito mais rápida do que as mais eficientes versões do BASIC. Quando a velocidade de execução ou uma grande precisão forem exigidas, o código de máquina revela-se a melhor opção.

## Movendo o cursor

Muitos dos microcomputadores permitem o endereçamento das posições na própria tela. Mesmo que seu equipamento não se enquadre nesse caso, é possível mover o cursor na tela (para a esquerda, para a direita, para cima e para baixo) com relativa facilidade. Em primeiro lugar, você precisa saber quais os códigos ASCII usados para representar as teclas de controle do cursor. Este pequeno programa solicita a digitação de uma tecla e indica o valor ASCII correspondente:

```
1 REM DESCOBRINDO OS CODIGOS ASCII PARA
  AS TECLAS DO CURSOR
10 PRINT "PRESSIONAR UMA TECLA";
20 FOR I = 1 TO 1
30 LET K$ = INKEY$
40 IF K$ = " " THEN I = 0
50 NEXT I
60 PRINT ASC(K$)
70 GOTO 10
80 END
```

Essa rotina também permite encontrar o código da tecla RETURN (em geral, 13), ESCape (quase sempre, 27) e a de espaço (geralmente, 32), além dos códigos para as teclas de controle do cursor. Existem micros que utilizam estes valores: 8, para deslocar o cursor para a esquerda; 28, para a direita; 29, para cima e 30, para baixo. Seu computador, com certeza, tem valores diferentes. Para substituí-los no programa citado, experimente o seguinte:

```
10 PRINT CHR$(12): REM UTILIZAR O CODIGO CLS
   OU O CODIGO CORRESPONDENTE
20 FOR L = 1 TO 39
30 PRINT "*";
40 NEXT L
50 FOR L = 1 TO 22
60 PRINT CHR$(8); :REM UTILIZAR O CODIGO
   "CURSOR ESQUERDO"
70 NEXT L
80 FOR L = 1 TO 4
90 PRINT "@";
100 NEXT L
110 END
```

Esse programa imprimirá na tela uma linha parecida com a que se segue:

```
****************@@@@****************
```

As linhas de 20 a 40 teriam imprimido apenas uma linha com 39 asteriscos. Contudo, as linhas de 50 a 70 ''imprimiram'' o ''caractere'' cursor para a esquerda 22 vezes; desse modo, o cursor se deslocou 22 posições para trás. As linhas de 80 a 100 imprimiram @ quatro vezes, e o programa se encerrou. Técnicas como essa permitem deslocar o cursor pela tela para impressão de caracteres em novas posições que podem não ser conhecidas até que seus valores sejam calculados pelo programa. Essa técnica tem a vantagem de possibilitar o emprego de caracteres comuns de tela para traçar gráficos simples, sem usar os recursos especiais do computador (se houver).

Para verificar de que modo esse tipo de controle de cursor pode ser feito na produção de gráficos como saída de seus programas, experimente a listagem abaixo:

```
10 PRINT "ESTE PROGRAMA IMPRIME UM GRAFICO
   DE BARRAS DE TRES VARIAVEIS"
20 INPUT "FORNECER OS TRES VALORES ";X,Y,Z
30 PRINT
40 FOR L = 1 TO 2
50 FOR A = 1 TO X
60 PRINT "*";
70 NEXT A
80 PRINT CHR$(13)
90 NEXT L
100 FOR L = 1 TO 2
110 FOR A = 1 TO Y
120 PRINT "+";
130 NEXT A
140 PRINT CHR$(13)
150 NEXT L
160 FOR L = 1 TO 2
170 FOR A = 1 TO Z
180 PRINT "#";
190 NEXT A
200 PRINT CHR$(13)
210 NEXT L
220 PRINT
230 END
```

Esse programa imprime um gráfico de barras das três variáveis. As barras são impressas em linhas horizontais, a partir da esquerda, e seguem o movimento ''natural'' do cursor. Observe que é necessária uma instrução PRINT CHR$(13) nas linhas 80, 140 e 200. Isso porque os sinais de ponto e vírgula no final das instruções PRINT suprimem os RETURN (13 é o código ASCII para <CR>).

## Mais sobre variáveis

Até aqui, tratamos das variáveis como se fossem apenas de dois tipos (numéricas e alfanuméricas). Na verdade, há muitos tipos de variáveis numéricas que o BASIC pode reconhecer. O bom programador sempre especifica o tipo correto, para economizar capacidade de memória e garantir a precisão.

Quando se declara uma variável em linguagem de programação, certa quantidade de espaço na memória é reservada para seu armazenamento. Se o programa souber que a variável sempre será inteira (por exemplo, LET CONTAGEM = TOTAL + BONIFICAÇÃO – MULTA), reservará menor capacidade de memória. Se tivermos uma variável que pode assumir infinitos valores reais (por exemplo, LET ÁREA = PI *RAIO* RAIO), o espaço de memória requerido será maior.

No desenvolvimento de nossa agenda de endereços computadorizada, adotamos a convenção de especificar variáveis alfanuméricas usando o sinal $ após o nome da variável (por exemplo, LET CHAVE$ = CAMPMOD$(TAMA)). As que aparecem sem o sinal de cifrão foram consideradas numéricas comuns. Porém, convenções semelhantes podem ser usadas depois de nomes de variáveis para especificar o tipo de variável numérica. Quando não há especificador, consideramos variáveis numéricas reais de precisão simples. Outros sinais admitidos pela maioria das linguagens BASIC são: % para especificação de variáveis inteiras; ! para as de precisão simples; e # para as de precisão dupla (isto é, as variáveis que podem armazenar o dobro de dígitos significativos). Eis um fragmento de programa hipotético que emprega esses três sinais:

```
70 LET JOGADOR = "JOAO": REM UMA
   VARIAVEL ALFANUMERICA
80 LET PLACAR% = 0: REM UMA
   VARIAVEL INTEIRA
90 LET PI! = 3.1416: REM UMA VARIAVEL
   DE PRECISAO SIMPLES
```

```
100 LET AREA# = PI*R*R: REM VARIAVEL
    DE PRECISAO DUPLA
110 LET JOGADAS = 6: REM ASSUME
    COMO SENDO REAL DE PRECISAO
    SIMPLES
```

Nem todas as versões da linguagem BASIC possuem todos esses tipos de variáveis. O Spectrum da Sinclair, por exemplo, não dispõe de variáveis inteiras. Os números inteiros são armazenados como reais de precisão simples. Também não possui números de dupla precisão. Contudo, no BASIC do Spectrum, números de precisão simples são calculados com até nove algarismos significativos, enquanto no BASIC da Microsoft a precisão resume-se a sete algarismos significativos. Vários micros admitem variáveis do tipo inteiro e números reais de precisão simples calculados até nove algarismos significativos. O BASIC da Microsoft admite variáveis de precisão dupla para dezesseis posições significativas.

Os computadores que efetivamente trabalham com variáveis inteiras utilizam quase sempre 2 bytes de memória para armazenar o número, que pode estar entre −32.768 e 32.767. Essa amplitude, em geral, mostra-se bem adequada para variáveis como resultados, quantidade de funcionários, contadores de loops FOR-NEXT e outros números que devem ter valores inteiros. Uma vez que se empregam apenas 2 bytes para armazenar cada número, o uso de variáveis inteiras, se houver, economiza tempo, embora para muitas versões de BASIC isso aconteça apenas com matrizes inteiras e não com variáveis individuais.

Exemplificamos abaixo as quatro funções restantes de nossa agenda computadorizada:

Opção 2 - Encontrar nomes (por trecho de um nome)
Opção 3 - Encontrar registros (de uma cidade)
Opção 4 - Encontrar registros (de uma inicial)
Opção 5 - Listar todos os registros

Para acrescentar o programa listado a seguir em nosso programa principal, devemos substituir as linhas de 4050 a 4080 do programa original para:

```
4050 IF ESCL=2 THEN GOSUB 16000: REM *ENCNOM*
4060 IF ESCL=3 THEN GOSUB 17000: REM *ENCCID*
4070 IF ESCL=4 THEN GOSUB 18000: REM *ENCINI*
4080 IF ESCL=5 THEN GOSUB 19000: REM *LISREG*
```

Por outro lado, todo o programa da agenda computadorizada foi desenvolvido tomando-se por base a versão BASIC da Microsoft. Caso seu equipamento tenha outra versão do BASIC, o programa pode ser convertido se você adaptá-lo a partir das indicações de nossos quadros "A propósito..." e também estudando o manual BASIC do fabricante de seu equipamento.

```
16000 REM SUB-ROTINA *ENCNOM*
16010 REM ESTA SUB-ROTINA ENCONTRA
16020 REM UM REGISTRO A PARTIR DE UM
16030 REM TRECHO DE PRIMEIRO NOME OU
16040 REM DE UM TRECHO DE SOBRENOME
16050 PRINT CHR$(12): REM LIMPA TELA
16060 PRINT
16070 PRINT
16080 PRINT
16090 PRINT "FORNECER UM TRECHO DE UM PRIMEIRO NOME"
16100 PRINT "OU TRECHO DE UM SOBRENOME A SER
      ENCONTRADO"
16110 TRECHO$=" "
16120 INPUT TRECHO$
16130 LET W=LEN(TRECHO$)
16140 REM CONVERTER TRECHO$
      EM MAIUSCULAS
16150 LET N$=TRECHO$
16160 GOSUB 20000: REM SUB-ROTINA
      *MAIUSCULAS*
16170 REM
16180 REM
16190 REM
16200 REM
16210 REM
16220 REM
16230 LET TRECHO$=P$
16240 LET P$=" ": LET N$=" "
16250 REM FAZER COMPARACAO DE TRECHO COM
16260 REM TODOS OS REGISTROS DA AGENDA
16270 REM ATE ENCONTRAR O TRECHO
16280 FOR L=1 TO TAMA-1
16290 FOR K=1 TO LEN(CAMPMOD$(L))-W
16300 TEMP$=MID$(CAMPMOD$(L),K,W)
16310 IF TRECHO$=TEMP$ THEN GOTO 16500
16320 NEXT K
16330 NEXT L
16340 REM REGISTRO NAO ENCONTRADO
16350 PRINT CHR$(12)
16360 PRINT
16370 PRINT
16380 PRINT "INFELIZMENTE NAO FOI ENCONTRADO"
16390 PRINT "UM REGISTRO CONTENDO O
      TRECHO: ";TRECHO$
16400 PRINT
16410 PRINT "(PRESSIONAR A BARRA DE ESPACO PARA
      CONTINUAR)"
16420 FOR I=1 TO 1
16430 IF INKEY$ < > " " THEN I=0
16440 NEXT I
16450 RETURN
16500 LET CORR=L
16510 GOSUB 13380: REM REGISTRO ENCONTRADO
16520 RETURN
17000 REM SUB-ROTINA *ENCCID*
17010 REM ESTA SUB-ROTINA ENCONTRA TODOS OS
17020 REM REGISTROS DE UMA MESMA CIDADE
17030 REM
17040 PRINT CHR$(12): REM LIMPA TELA
17050 PRINT
17060 PRINT
17070 PRINT
17080 PRINT "FORNECER O NOME COMPLETO DA CIDADE"
17090 PRINT "PARA ENCONTRAR OS REGISTROS"
17100 LET CIDADE$=" "
17110 INPUT CIDADE$
17120 REM CONVERTER CIDADE$ EM MAIUSCULAS
17130 LET N$=CIDADE$
17140 GOSUB 20000: REM SUB-ROTINA *MAIUSCULAS*
17150 LET CIDADE$=P$
17160 LET P$=" ": LET N$=" "
17170 REM
```

```
17180 REM
17190 REM
17200 FOR L=1 TO TAMA-1
17210 REM CONVERTER CAMPCID$ EM MAIUSCULAS
17220 LET CID$=" ": LET CID$=CAMPCID$(L)
17230 LET N$=CID$
17240 GOSUB 20000: REM SUB-ROTINA *MAIUSCULAS*
17250 LET CID$=P$
17260 LET P$=" ": LET N$=" "
17270 IF CIDADE$ < > CID$ THEN GOTO 17300
17280 CORR=L: GOSUB 13400: REM REGISTRO ENCONTRADO
17290 REM
17300 NEXT L
17310 PRINT
17320 PRINT "ISSO E TUDO QUE FOI ENCONTRADO"
17330 PRINT "PARTIR DA CIDADE: ";CIDADE$
17340 PRINT
17350 PRINT " (PRESSIONAR A BARRA DE ESPACO PARA
      CONTINUAR) "
17360 FOR I=1 TO 1
17370 IF INKEY$ < > " " THEN I=0
17380 NEXT I
17390 RETURN
18000 REM SUB-ROTINA *ENCINI*
18010 REM ESTA SUB-ROTINA ENCONTRA REGISTROS
18020 REM A PARTIR DAS INICIAIS DO NOME
18030 REM NO FORMATO INICIAL DO PRIMEIRO NOME
18040 REM E INICIAL DO SOBRENOME
18050 PRINT CHR$(12): REM LIMPA TELA
18060 PRINT
18070 PRINT
18080 PRINT
18090 PRINT "FORNECER AS INICIAIS DO REGISTRO"
18100 PRINT "A SER ENCONTRADO NO FORMATO DE"
18110 PRINT "INICIAL DO PRIMEIRO NOME E INICIAL"
18120 PRINT "DO SOBRENOME."
18130 PRINT "EXEMPLO: RA PARA RAUL AGUIAR"
18140 LET INICIAL$=" "
18150 INPUT INICIAL$
18160 REM CONVERTER INICIAL$ EM MAIUSCULAS
18170 LET N$=INICIAL$
18180 GOSUB 20000: REM SUB-ROTINA *MAIUSCULAS*
18190 LET INICIAL$=P$
18200 LET P$=" ": LET N$=" "
18210 REM
18220 FOR L=1 TO TAMA-1
18230 REM ENCONTRAR INICIAIS DE CAMPMOD$
18240 FOR K=1 TO LEN(CAMPMOD$(L))
18250 IF MID$(CAMPMOD$(L),K,1)=" " THEN GOTO 18270
18260 NEXT K
18270 LET INIC$=MID$(CAMPMOD$(L),
      K+1,1)+MID$(CAMPMOD$(L),1,1)
18280 IF INICIAL$ < > INIC$ THEN GOTO 18310
18290 LET CORR=L: GOSUB 13400
18300 REM
18310 NEXT L
18320 PRINT "ISSO E TUDO QUE FOI ENCONTRADO"
18330 PRINT "A PARTIR DAS INICIAIS: ";INICIAL$
18340 PRINT
18350 PRINT " (PRESSIONAR A BARRA DE ESPACO
      PARA CONTINUAR) "
18360 FOR I=1 TO 1
18370 IF INKEY$ < > " " THEN I=0
18380 NEXT I
18390 RETURN
19000 REM SUB-ROTINA *LISREG*
19010 REM ESTA SUB-ROTINA LISTA TODOS OS
19020 REM REGISTROS DA AGENDA COMPUTADORIZADA
19030 REM
19040 PRINT CHR$(12): REM LIMPA TELA
19050 PRINT
19060 PRINT
19070 PRINT
19080 PRINT "PRESSIONE: "
19090 PRINT "T PARA LISTAR TODOS OS REGISTROS NA TELA"
19100 PRINT "I PARA LISTAR TODOS OS REGISTROS NA
      IMPRESSORA"
19110 PRINT "A BARRA DE ESPACO PARA VOLTAR AO MENU
      PRINCIPAL"
19120 FOR I=1 TO 1
19130 LET A$=INKEY$
19140 IF A$=" " THEN I=0
19150 NEXT I
19160 IF A$="T" THEN GOTO 19300
19170 IF A$="I" THEN GOTO 19600
19180 IF A$=" " THEN RETURN
19190 GOTO 19120
19300 REM IMPRIMIR TODOS OS REGISTROS NA TELA
19310 PRINT CHR$(12): REM LIMPA TELA
19320 REM
19330 LET T=0
19340 FOR L=1 TO TAMA-1
19350 PRINT: LET T=T+1
19360 PRINT "REGISTRO NO. ";CAMPIND$(L)
19370 PRINT "NOME: ";CAMPNOM$(L)
19380 PRINT "RUA: ";CAMPRUA$(L)
19390 PRINT "CIDADE: ";CAMPCID$(L)
19400 PRINT "ESTADO: ";CAMPEST$(L)
19410 PRINT "TELEFONE: ";CAMPTEL$(L)
19420 IF T=3 THEN GOSUB 19500
19430 NEXT L
19440 PRINT
19450 PRINT "***FIM DO ARQUIVO***"
19460 PRINT
19470 GOTO 19080
19500 PRINT
19510 REM
19520 PRINT "PRESSIONE QUALQUER TECLA PARA CONTINUAR"
19530 FOR I=1 TO 1
19540 IF INKEY$=" " THEN I=0
19550 NEXT I
19560 LET T=0
19570 RETURN
19600 REM IMPRIMIR TODOS OS REGISTROS NA IMPRESSORA
19610 PRINT CHR$(12): REM LIMPA TELA
19620 PRINT
19630 PRINT
19640 PRINT "VERIFIQUE A IMPRESSORA E O PAPEL"
19650 PRINT
19660 PRINT "PRESSIONE QUALQUER TECLA PARA INICIAR A
      IMPRESSAO"
19670 FOR I=1 TO 1
19680 IF INKEY$=" " THEN I=0
19690 NEXT I
19700 FOR L=1 TO TAMA-1
19710 LET CORR=L
19720 GOSUB 13620: REM SUB-ROTINA *IMPCOR*
19730 NEXT L
19740 GOTO 19040
20000 REM SUB-ROTINA *MAIUSCULAS*
20010 REM ESTA SUB-ROTINA CONVERTE UMA VARIAVEL
20020 REM ALFANUMERICA QUALQUER EM MAIUSCULAS
20030 FOR M=1 TO LEN(N$)
20040 LET TEMP$=MID$(N$,M,1)
20050 LET T=ASC(TEMP$)
20060 IF T >= 97 THEN T=T-32
20070 LET TEMP$=CHR$(T)
20080 LET P$=P$+TEMP$
20090 NEXT M
20100 RETURN
```

# Desafio universitário

## O primeiro computador programável do mundo foi desenvolvido na Universidade de Manchester, na Inglaterra.

Terminada a Segunda Guerra Mundial, a Universidade de Manchester nomeou dois novos professores. Max Newman — depois de trabalhar em decifragem com o Colossus, o primeiro computador eletromecânico do mundo — tornou-se professor de matemática, e um engenheiro especializado em radares, F. C. Williams, passou a chefiar o Departamento de Engenharia Elétrica. Williams levou consigo seu jovem assistente, Tom Kilburn — familiarizado com os problemas dos dispositivos de memória eletrônica de pulso —, que conhecera durante a guerra, quando trabalhava com radares. Kilburn viria a ser o primeiro professor de computação na Universidade de Manchester.

Em 1946, durante uma visita a estabelecimentos que desenvolviam radares, nos Estados Unidos, Williams conheceu o protótipo do computador a válvula ENIAC (ver p. 46), e quando voltou à Inglaterra convenceu a Royal Society a investir 35.000 libras (mais de 50.000 dólares) num laboratório de máquinas de calcular, em Manchester. Essa instituição, contudo, não estava sozinha na corrida para construir um computador com programa armazenado. A Universidade da Pensilvânia (nos Estados Unidos) desenvolvia o EDVAC, a de Cambridge (americana) trabalhava no EDSAC, e prosseguia o projeto do ACE no Laboratório Físico Nacional (ver p. 88). Todos, no entanto, estavam usando armazenagem de memória em tubos de mercúrio. Apenas a equipe de Manchester optava pela máquina com dispositivo de memória (inventada por Williams) que utilizava um tubo de raios catódicos (TRC). Nesse trabalho, em fins de 1947, ele conseguira reter 2.048 bits por várias horas.

Baseado no que ficou conhecido como "válvula de Williams", o computador de Manchester, o Mark I, rodou com êxito um programa em junho de 1948, transformando-se assim no primeiro do gênero em todo o mundo. O Mark I podia executar uma instrução em 1,2 milissegundo. Usando um TRC para armazenar informações, a memória tinha a vantagem do acesso randômico. Além disso, seu conteúdo ou o registro de controle podia ser exibido visualmente.

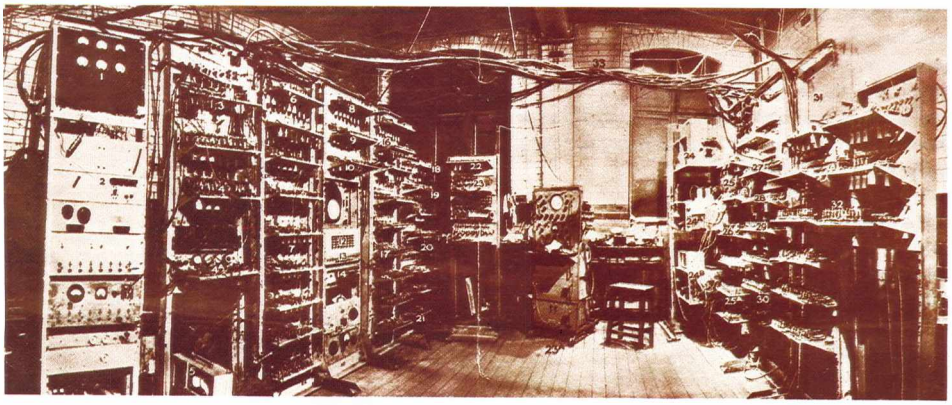
Estabelecida a viabilidade da válvula de Williams, construiu-se o Mark I aperfeiçoado, capaz de trabalhar com problemas de desenho óptico e geração de números primos. O principal responsável pelos assuntos de ciência do governo, Sir Ben Lockspeiser, ficou tão impressionado com o desempenho do computador que conseguiu convencer a Ferranti, uma empresa de Manchester, a construir a versão comercial do Mark I. O novo produto foi comercializado em fevereiro de 1951, precedendo o UNIVAC americano em cinco meses e tornando-se, portanto, o primeiro computador a chegar ao mercado.

## Pensando menos

Uma inovação importante do Ferranti Mark I era sua capacidade de modificar instruções enquanto as processava, recorrendo a outro armazenamento (denominado válvula B). Quando necessário, ele acrescentava o conteúdo desse segundo armazenamento ao registro de controle e modificava o código da instrução original. Assim, acelerava o processamento de programas. A IBM usou algumas das patentes de Manchester em seus primeiros computadores e, numa visita de Williams à sede em Nova Iorque — onde o lema da empresa, THINK (pense), estava afixado por toda a parte —, foi-lhe perguntado como a equipe inglesa conseguira construir um computador, ao passo que todos os recursos da grande corporação americana haviam falhado. Williams respondeu rapidamente: "É que nós não paramos muito para pensar!" Com a chegada de Alan Turing (ver p. 200) a Manchester, em 1948, intensificaram-se as atividades de pesquisa e projetos. Em 1950, ele produziu o primeiro manual de programação da universidade. Sua equipe teve a idéia de construir um computador mais compacto e econômico. Os planos puderam ser acelerados pela invenção do transistor. E, em novembro de 1953, o primeiro computador transistorizado do mundo tornava-se operacional.

Nos últimos anos da década de 50, os Estados Unidos assumiram a liderança tecnológica do setor de informática, levando o governo inglês a investir num projeto que ajudaria o país a recuperar o primeiro lugar. Para tanto, autorizou-se, em dezembro de 1962, a construção do computador Atlas, sob a direção de Tom Kilburn. Essa máquina usava um formato de palavra de 48 bits com endereço único, memória principal de 16 Kbytes e uma ROM de 8 Kbytes. Foram vendidos modelos para o Departamento de Pesquisa de Energia Atômica em Harwell e para a British Petroleum. Por muitos anos, o Atlas manteve a reputação de ser o computador mais avançado da época.

**O Mark I**
Após uma experiência bem-sucedida em junho de 1948, o Mark I, desenvolvido em Manchester, tornou-se o primeiro computador com programa armazenado do mundo. A Ferranti, uma empresa local, foi encarregada de desenvolver a versão comercial da máquina, lançada no mercado no começo de 1951.



# Risco calculado



**Uma tarde no Jóquei**
Prognósticos sobre corridas de cavalos constituem aplicação interessante de microcomputadores, mas especialistas constataram que as chances de se acertar o vencedor de um páreo continuam muito limitadas.

**Os computadores têm muitas aplicações no mundo dos jogos de azar. Existem programas de sistemas de apostas até para microcomputadores.**

Os jogos de azar giram em torno de probabilidades. Embora a maior parte dos jogadores contumazes prefira dizer que eles estão aí para serem ganhos, essa afirmação mostra-se infundada, pois quase todos os jogadores perdem com freqüência, e no total essas perdas são consideráveis. Isso acontece porque as chances estão contra eles e a favor do Jóquei Clube, do cassino, da Loto, da Loteria Esportiva. Para avaliar se os computadores poderiam ajudar a restabelecer o equilíbrio, precisamos, de início, considerar as chances. (*Chance*, aqui, entendida como probabilidade; por exemplo: a chance de se obter determinada face jogando um dado não viciado é de 1/6.)

Desguarnecidos das armadilhas externas, todos os jogos de azar se resumem em apostar no resultado de um evento fortuito. Em geral, este é gerado por algum dispositivo randômico (aleatório), como uma bola rodando numa roleta ou uma carta puxada de um baralho bem misturado. Se os parâmetros — digamos, o número de cartas — são conhecidos, o cálculo da probabilidade permite certas previsões sobre as chances de ocorrência do evento. A roleta mais usual, com 37 divisões numeradas de 0 a 36, exemplifica o cálculo das probabilidades: a bola pode cair em dezoito casas pares ou dezoito ímpares, mais o zero. A probabilidade de a bola cair numa casa com número ímpar pode ser expressa como 18/37, ou seja, cerca de 48,6%. Isso é um pouco menos que a probabilidade (50%) de uma moeda arremessada para o ar dar "cara"; a diferença, devido à casa do zero, representa a margem de lucro do cassino.

Essa margem faz com que os jogos de acaso sejam pouco gratificantes. Apesar de os fornecedores de sistemas para esses jogos afirmarem o contrário, um computador nada pode fazer para melhorar as chances básicas do apostador. De fato, grandes autoridades no assunto calculam que, a longo prazo, é impossível ganhar em qualquer jogo de cassino, exceto, talvez, em alguns jogos de cartas.

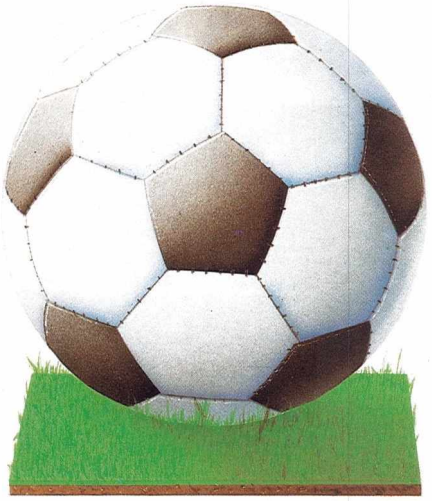
Essas objeções teóricas não desencorajaram inventores entusiasmados, e os proprietários de microcomputadores têm agora à disposição uma variedade de sistemas de jogo alegadamente seguros — e alguns deles parecem funcionar mesmo.

## Loteria Esportiva

Os sistemas de jogo dos cassinos quase sempre são variações de "dobrar a aposta", processo que leva a desvantagem de exigir quantia infinita de dinheiro para aposta, a fim de ser bem-sucedido. Há ainda outro problema: não se permite, nos cassinos, a utilização de qualquer tipo de computador, pois essas máquinas são consideradas de grande ajuda quando habilidade, memória ou estratégia estão envolvidas. O computador pode impor a quem joga a disciplina necessária e atuar como recurso de memória. No entanto, o valor dessa assistência tende a ser inversamente proporcional à perícia do jogador.

Por outro lado, o fator habilidade envolvido em quase todos os jogos de azar é mínimo. As apostas em futebol (como o concurso de prognósticos da Loteria Esportiva) são um caso à parte. O programa de previsão de apostas mais sofisticado é o famoso F4 Football Forecast, do professor Frank George, encontrado no mercado internacional em versões adequadas para a maioria dos microcomputadores. Baseado em dez anos de análise estatística, o programa atribui determinado valor ao desempenho médio de cada time. Quando ajustados pela aplicação de pesos relativos ao desempenho a longo prazo — extraídos de tabelas oficiais ou do noticiário esportivo —, ao desempenho a curto prazo e ao resultado da última partida, a comparação desses números possibilita ao programa predizer o resultado mais provável de determinada partida.

Com o Palmeiras jogando em casa contra o Ferroviário Ituano, por exemplo, prevê-se com facilidade o resultado, mas o programa mostra de fato seu valor quando prognostica o escore de uma partida entre times mais equilibrados. Isso não significa que o programa sempre acerte, mas a análise estatística indica que seu uso quase triplica a probabilidade de sucesso. "Reconheço que as chances *contra* ainda são enormes, mas é melhor jogar da forma mais inteligente possível, não é?", pergunta o autor.



**Sistemas de apostas**
Existem vários pacotes para microcomputadores que alegadamente melhoram as chances de ganhar no sistema de apostas do tipo da Loteria Esportiva brasileira. Grande número de programadores já tentou escrever seu próprio programa com essa finalidade. Os melhores sistemas desenvolvidos fazem uso de vasto banco de dados com informações sobre as partidas anteriores entre as equipes envolvidas e podem ser úteis na previsão dos resultados das partidas.

## Corridas de cavalos

Mesmo com essa ajuda, as probabilidades continuam bem desfavoráveis. Um dos maiores promotores ingleses de apostas concluiu que nenhum usuário de microcomputador jamais ganhou qualquer grande prêmio. "Se houvesse um sistema que funcionasse mesmo, eu seria o primeiro a saber", afirma ele. Embora as entradas sejam feitas por máquinas automáticas especiais, sua empresa de jogos só usa computadores para manter os registros.

As corridas de cavalos parecem oferecer variedade maior ao programador. O estudante inglês David Stewart criou um programa de microcomputador destinado a prever os vencedores dos páreos. Escrito para o Sinclair ZX-81 e adaptado para o Spectrum, o programa obteve êxito muitas vezes. Embora os palpites de David sejam divulgados por diversas estações de rádio, ele não amealhou grande fortuna.



**Rodando a roleta**
Numa roleta, é o número 0 que dá lucro ao cassino. A probabilidade de se acertar em cheio num dos outros 36 números é de 1/37. Praticamente nada pode ser feito para melhorar as chances básicas do jogador.

Significativamente, os profissionais de corridas de cavalos rejeitam o uso de computadores. Na Inglaterra, por exemplo, os prognósticos oficiais ainda são feitos a mão pelo Jóquei Clube (embora os dados fiquem armazenados no computador). *Timeform*, revista especializada de grande aceitação entre os apostadores, também compila a mão a maior parte de seus dados. "Só usamos o computador para calcular o tempo padrão de cada cavalo, levando em consideração a resistência do vento", explica um diretor da publicação. "Não existe nenhum sistema de prognósticos verdadeiramente computadorizado. Os computadores não conseguem dar conta dos resultados extraordinários que surgem todo dia."

O uso de computadores também aumentou do outro lado do balcão de apostas, mas não para calcular as chances. As equipes contratadas pelas grandes cadeias de lojas de *bookmakers* britânicos empregam calculadoras especiais para computar o retorno sobre as apostas. O setor de crédito do negócio está ficando cada vez mais computadorizado. O apostador que tenha conta numa das lojas pode fazer sua aposta por telefone, no centro de computação. Os detalhes são digitados e debita-se o valor da aposta na conta do cliente. Se o cavalo escolhido tiver o desempenho previsto, os ganhos são calculados e creditados na mesma conta.

Os *bookmakers*, na maioria, não acreditam em sistemas de prognósticos por computadores. "Ninguém jamais apareceu com um computador que ganhasse sempre. Caso contrário, não estaríamos aqui", zombam eles.

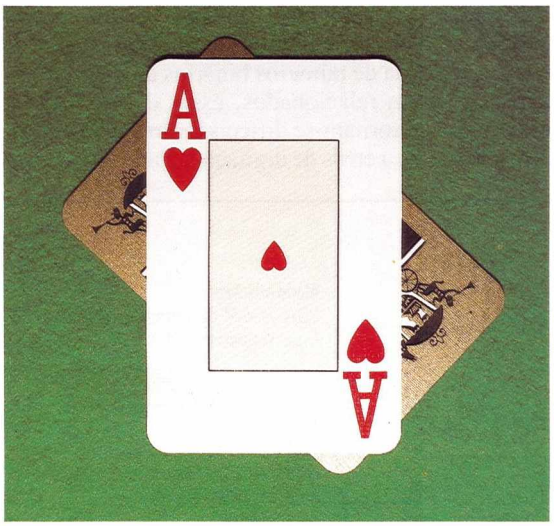
Uma empresa especializada em corridas de cavalos empreendeu extraordinária e controvertida simulação por computador. Registros pormenorizados do histórico dos ganhadores do Derby inglês foram incorporados num programa encomendado especificamente para esse fim, e convidaram-se leitores de jornais a dar palpites para os seis primeiros lugares. O resultado foi uma polêmica em torno da colocação do cavalo italiano Ribop, que nunca havia perdido uma corrida: a máquina colocou-o em quarto lugar!

Talvez o mais famoso "computador" para jogos de azar seja o ERNIE (Electronic Random Number Indicator Equipment, máquina eletrônica indicadora de números randômicos), que escolhe os ganhadores de uma loteria, na Inglaterra. Embora não seja programável, o ERNIE executa um programa. A máquina foi desenvolvida por Plessey em 1973 e sua função consiste em gerar randomicamente 200.000 números e gravá-los em fita magnética. Esses números são gerados a partir de uma série que começa pelo número de emissão mais baixo e termina com o mais alto. Então a fita é colocada num computador ICL de grande porte, que compara os números com uma listagem em fita dos números que já foram premiados. Eliminados estes números que não podem ser sorteados, o computador imprime os certificados dos prêmios e as cartas para os ganhadores. A probabilidade de um número ser sorteado na extração mensal é de apenas 1/15.000.

## Pretas X brancas

Outro jogo de probabilidade aparece em jornais diários britânicos. As chances de ganhar um dos prêmios de 1 milhão de libras oferecidos nas promoções de jornais são ainda mais remotas. Os números dos cartões distribuídos aos leitores contêm doze dígitos. Uma seqüência de doze dígitos que vai de 000000000000 a 999999999999 oferece 1 bilhão de combinações. Estatisticamente, a chance de sorteio de determinado número seria pouco melhor que 1/1 bilhão, pois o jornal envolvido publica dois números por dia. Nessa base, é bem duvidoso que o jornal tenha de pagar o grande prêmio, pois não vende 1 bilhão de exemplares diários.

A situação pode ser melhor visualizada se você imaginar um saco que contenha 2,5 milhões de bolas brancas, representando os competidores (o número de cartões em circulação) e 1 bilhão de bolas pretas para o número total de combinações possíveis. As chances de sortear uma bola branca na primeira vez são bastante remotas. E não melhoram muito quando são tiradas as bolas correspondentes a um ano de extração. Os estatísticos computam em 1/667 as chances de o jornal ter de pagar o prêmio de 1 milhão de libras.



**Lance de sorte**
Jogos de baralho proporcionam um dos melhores campos para o desenvolvimento de programas que possam ajudar a sorte.
A capacidade de memorizar as cartas já jogadas aumenta as chances de sucesso do apostador.
Os cassinos não permitem o uso de computadores nas mesas de jogo, mas em todas as grandes façanhas foram usados micros escondidos no corpo do apostador ou ligações radiofônicas com máquinas operadas fora do salão de jogo.

## Jogo de dados

A maior parte das versões de BASIC possui função de geração de números randômicos. Contudo, em muitos casos, os números assim gerados não são de fato randômicos, como demonstra este programa:

```
10 LET A = RND
20 LET B = RND
30 LET C = RND
40 PRINT A,B,C
```

Nas três primeiras linhas, um número supostamente randômico é designado para as variáveis A, B e C. Depois elas são impressas. Isso pode fornecer os seguintes resultados, entre outros:

```
,014007      ,964370      ,457397
```

Mas, se você tornar a executar o programa, a maioria dos microcomputadores mostrará de novo a mesma seqüência. Acontece que quando pedimos RND, o computador responde com o próximo número, numa seqüência fixa de números. Isso pode compreender 1 milhão de frações de seis dígitos entre 0,000000 e 0,999999, cada um ocorrendo uma vez no ciclo completo (mas não em seqüência).

Alguns tipos de BASIC usam sintaxe um pouco diferente, exigindo uma expressão em parênteses, chamada "argumento", que assume a forma LET A = RND(X). O efeito mostra-se bem semelhante: RND e RND(X) podem ser utilizadas da mesma forma que outras variáveis.

Versões diferentes de BASIC também apresentam a função RANDOMIZE, que faz a seqüência começar num ponto imprevisto. Inserindo o comando RANDOMIZE em qualquer programa onde o RND seja usado, assegura a geração de uma seqüência diferente de números cada vez que o programa for executado (RUN).

Para simular o arremesso de um dado, precisamos de números inteiros, que vão de 1 a 6. No entanto, devem-se eliminar as frações. Isso é feito por meio do uso da função INTeger (número inteiro). PRINT INT(6,99) produz o resultado 6 tão certamente quanto PRINT INT(6,01). Qualquer coisa que venha depois da vírgula é sempre descartada, por definição.

Como o maior número que RND pode gerar é 0,999999 (o qual, quando expresso como um número inteiro, assume o valor 0), uma pequena multiplicação se faz necessária. A fórmula tradicional é:

```
LET A = INT(6*RND)+1
```

Multiplicamos por 6 porque um dado tem seis faces. O "mais um" assegura que os resultados vão de 1 a 6, e não de 0 a 5.

# Linha de montagem

**Diversos são os modos pelos quais os programas podem ser expressos. Os códigos de máquina incluem desde a linguagem binária até a Assembly.**

Uma das dificuldades conceituais que os principiantes experimentam com relação à linguagem de máquina está nas várias formas pelas quais os programas podem se apresentar. Qualquer dado armazenado na memória do computador assume, em última análise, a forma de números binários de 8 bits. Contudo, ao serem relacionados, esses dados ocupam muito espaço, tornam-se difíceis de ler e de memorizar, e induzem a erros de digitação. Para evitar esses inconvenientes, empregamos quase sempre os números hexadecimais (hex), que permitem a representação dos conteúdos de qualquer byte como número de dois dígitos. Os hex possibilitam ainda que os endereços no limite da memória do computador (0 a 65535, em decimal) sejam representados por quatro dígitos.

Em geral, números hex escritos no papel são precedidos por um sinal $, para distingui-los de números decimais, embora esse sinal não fique retido na memória do computador. Além disso, quando um código de operação possui um operando de 2 bytes (por exemplo, LDA $3F80), eles são fornecidos ao equipamento em ordem inversa — ao último byte segue-se o primeiro. Assim, no exemplo apresentado, os 3 bytes seriam AD (que corresponde à representação do código de operação LDA na linguagem 6502) seguido por 80 e depois por 3F. Esse procedimento facilita as operações do processador, mas pode confundir o usuário.

O programa em código de máquina é quase sempre impresso como um "despejo hex" — longa lista de valores hexadecimais de dois dígitos. Além disso, obtém-se um endereço de início (em hex ou em decimal). O primeiro valor hex é carregado nessa posição; o segundo, na seguinte; e assim por diante. Obtém-se o carregamento por meio do comando POKE da linguagem BASIC. Se o endereço de início for $1000 (4096, em decimal) e o despejo hex for

AD (173, em decimal)
80 (128, em decimal)
3F (63, em decimal)

o programa poderá ser carregado com três instruções em BASIC:

```
POKE 4096,173
POKE 4097,128
POKE 4098,63
```

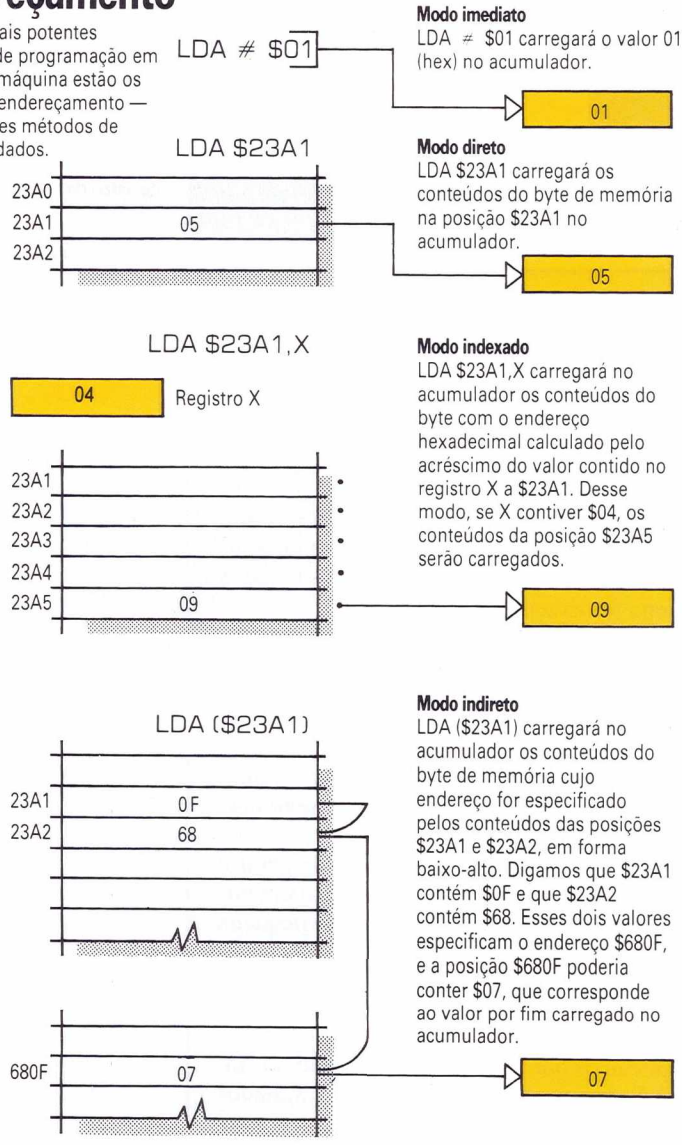
Observe que temos de converter todos os valores de hex para decimal, antes de utilizá-los na instrução POKE. No interior do equipamento eles serão armazenados em código binário.

Para despejos mais longos é freqüente o emprego de um pequeno programa em BASIC denominado "carregador de código de máquina". Esse procedimento requer o endereço de início e, em seguida, os valores hex. À medida que é fornecida, a pequena rotina em BASIC converte o valor hex em decimal e armazena-o por meio do comando POKE na posição seguinte. Como alternativa, o despejo hex pode ser lido pelo programa a partir das instruções DATA.

Uma vez fornecido o código de máquina, pode-se abandonar o programa carregador em BASIC. Con-

## Modos de endereçamento

Entre os mais potentes conceitos de programação em código de máquina estão os modos de endereçamento — os diferentes métodos de recuperar dados.



## Por extenso/ abreviado

O programa em código de máquina pode assumir diferentes formas. Em geral, desenvolve-se pelo programador como linguagem Assembly, que emprega os sinais mnemônicos para os códigos de operação e títulos para os operandos, assim:

```
LDA PESO
ADC COMBUSTÍVEL
STA PESO
```

Devemos, contudo, especificar os endereços desses títulos. Por exemplo:

```
COMBUSTÍVEL = $03EE
PESO = $031F
```

O pacote assembler transformará isso num despejo hex, pelo emprego de unidade de discos. A "pseudolinguagem assembly", usada abaixo, é de leitura mais difícil, mas com freqüência pode ser fornecida a um pacote denominado "spot assembler", que não necessita de discos.

```
LDA $031F
ADC $03EE
STA $031F
```

O despejo hex consiste num endereço de início (à esquerda) e na seqüência de valores hex de dois dígitos como aparecerão na memória. Observe que um operando como $031F é armazenado em ordem inversa (1F03) e que o valor hex correspondente substitui os códigos de operação:

19C4 AD 1F 03 6D EE 03 8D 1F 03

vém carregar o código de máquina num ponto da memória em que ele não seja "atropelado" pelo programa em BASIC, nem eliminado por instruções dessa linguagem, como NEW.

A maioria dos microcomputadores possui comando BASIC indicativo de quando se deve parar de operar nessa linguagem e começar a executar o programa em código de máquina. Uma forma desse comando se apresenta como SYS 4096 (RETURN), que corresponde a "transferir o controle ao sistema a partir da posição decimal 4096". CALL $E651 significa "chamar a rotina de código de máquina a partir da posição hex E651".

A sub-rotina ou programa em código de máquina executa esse sistema ou rotina (que pode ou não apresentar resultados visíveis, dependendo do tipo de programa). Se estiver corretamente desenvolvido e incorporar o procedimento apropriado de encerramento, o controle será transferido de novo para a linguagem BASIC. Incidentalmente, isso implica a possibilidade de chamar sub-rotinas de código de máquina a partir de vários pontos na operação do programa em BASIC, sempre que se fizer necessária a execução de função em alta velocidade.

Uma das dificuldades da programação em linguagem de máquina está em que, se você errar, o computador não responderá com uma simpática e útil mensagem tipo SYNTAX ERROR. A probabilidade maior é que o programa "engasgue": o equipamento deixa de responder às instruções que você digitar. Isso não danifica o computador, mas você terá de zerá-lo (ou desligá-lo e depois ligar outra vez). E isso, em geral, implica fornecer novo programa desde o princípio. Você não pode, portanto, experimentar em código de máquina como faz em BASIC — a operação do programa deve ser conferida por completo em papel antes de ser fornecida ao computador.

O "monitor do código de máquina" (que não tem nenhuma relação com a tela do monitor) auxilia bastante o fornecimento e a conferência do código de máquina. Ele está incorporado na ROM de alguns computadores, mas o mais comum é sua aquisição em pacotes tipo cartucho. O monitor do código de máquina constitui um sistema operacional simples, que apresenta na tela os conteúdos de qualquer setor da memória. Esses valores (hex) podem ser alterados ou substituídos por sobreposição; desse modo, o monitor é a melhor forma de requisitar um despejo hex. Além disso, quase sempre possibilita o carregamento e a gravação em cassete de programas em código de máquina, sem necessidade do programa carregador em BASIC. Os programas de recursos de código de máquina mais desenvolvidos — o código de máquina correspondente aos pacotes ferramentas em BASIC (ver p. 444) — mostram os conteúdos de cada registro interno do processador.

Os despejos hex constituem eficiente modo de representação do código de máquina, mas não são fáceis de ser lidos. A menos que você memorize os correspondentes hexadecimais de vários códigos de operação, será quase impossível distingui-los dos operandos. Assim, desenvolve-se a maior parte dos programas com o auxílio de sinais mnemônicos de três letras, que apresentamos na página 449. Esses auxiliares da memória são então traduzidos para hex com o emprego de códigos do manual do microprocessador.

Uma forma mais sofisticada de monitor de código de máquina — o "spot assembler" — permite a digitação do programa em sinais mnemônicos, executando automaticamente as conversões.

Isso nos leva à última forma pela qual o código de máquina pode ser representado — a linguagem Assembly, que não só utiliza os sinais mnemônicos para códigos de operação como também manipula nomes (ou títulos), em vez de números hex para os operandos. Desse modo, se a posição $07B2 contiver o número de mísseis disparados num jogo, podemos carregá-lo no acumulador com a instrução:

```
LDA MISSIL
```

No começo do programa temos de especificar a posição de MISSIL = $07B2 e que esta deve, de início, conter o valor $09 (nove mísseis).

Depois de desenvolver esse programa em linguagem Assembly (o "código fonte" do programa) processamos um programa utilitário denominado assembler (montador), que opera todo o código, substituindo os sinais mnemônicos e os títulos por seus correspondentes hexadecimais. Cria, desse modo, uma nova versão, o "código objeto". Este pode então ser fornecido à memória do computador e executado. O processo não é muito diferente da compilação (ver p. 84), embora nesse caso haja correspondência unívoca entre o código fonte e o código objeto.

Por constituir um nível de linguagem mais alto que o código de máquina, a linguagem Assembly apresenta maior operacionalidade sem perda de qualidade no desempenho. Contudo, os pacotes assembler em geral só funcionam com unidades de disco e, desse modo, não podem ser utilizados por todos os computadores.

## Códigos

Abaixo, alguns códigos de operação que se encontram em microprocessadores comuns.

### JSR

**Jump SubRoutine**
(sub-rotina de desvio) Equivale à função GOSUB do BASIC. JSR $354D alterará os conteúdos do contador do registro do programa (PC) para fazê-lo executar o código a partir de $354D.

### RTS

**ReTurn from Subroutine**
(retorno da sub-rotina) Ao encontrar RTS, o processador voltará à posição a partir da qual a sub-rotina foi chamada (isto corresponde a RETURN em BASIC). RTS não possui operando porque o endereço de retorno terá sido automaticamente armazenado numa área especial da memória chamada Stack.

### BMI

**Branch if MInus**
(desviar, se negativo) É uma das várias formas de desvio condicional em código de máquina (em BASIC, IF-THEN GOTO é um desvio condicional). Se o resultado da última operação for um valor negativo no acumulador, a execução do programa saltará para um endereço especificado. BPL especifica Branch if PLus (desviar, se positivo).

### LDX

**LoaD X register**
(carregar o registro X) X é outro registro de 1 byte no interior do processador; não executa procedimentos aritméticos do mesmo modo que o acumulador, mas é utilizado para "endereçamento indexado" (ver quadro da p. 464). LDX carrega um valor em X e STX (STore X, armazenar X) o armazena de novo na memória.

### INX

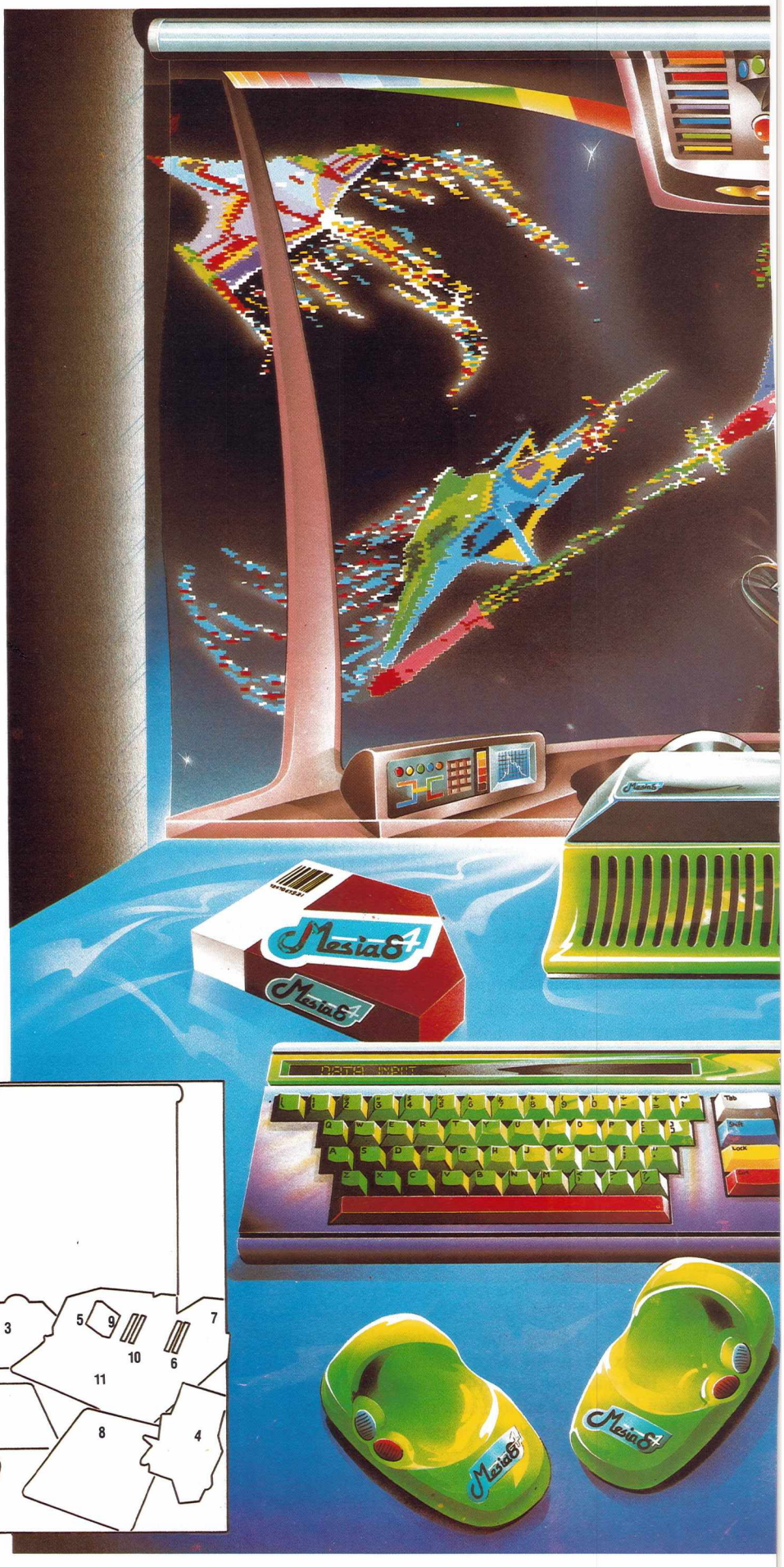
**INcrement X**
(aumentar X) Pelo acréscimo de uma unidade ao valor de X (DEX — DEcrement X, diminui X) e pela utilização do endereçamento indexado, pode-se passar gradativamente por uma série de posições na memória, executando o mesmo procedimento em cada uma.

# Futurologia

## Os microcomputadores tiveram extraordinário desenvolvimento nos últimos cinco anos, mas o que acontecerá na próxima década? Compare nossas idéias com suas previsões.

Como será o microcomputador da década de 90 e como funcionará? Vamos tentar responder a essas perguntas analisando alguns dos principais componentes e sistemas da máquina do futuro. Muitas dessas idéias fundamentam-se nas tecnologias que estão entrando no mercado (talvez em campos alheios à computação), enquanto outras representam os desenvolvimentos que consideramos prováveis.

Uma das características essenciais do nosso design hipotético é a modularidade. Depois de adquirir uma unidade base, o usuário disporá de grande variedade de opções para expandir a máquina. Na verdade, ele será virtualmente capaz de projetar seu próprio equipamento, selecionando, por exemplo, o módulo para gráficos ou o dispositivo de som que mais lhe aprouver. De uma coisa temos certeza: a velocidade das mudanças no mercado de computadores continuará acelerada por muitos anos.




### 1 Display

O microprocessador de 32 bits permitirá o display da informação em diversas formas simultaneamente. Por exemplo, a tela principal poderia mostrar o panorama visto de dentro da cabine, enquanto uma tela subsidiária, montada sobre o console de teclado/comando, estaria dando as informações de controle da cabine.

### 3 Monitor

Televisões de projeção existem desde o começo da década de 80, mas seu alcance é limitado pelo poder de emissão de luz do tubo de raios catódicos. As primeiras televisões de projeção precisavam de telas curvas, mas os modelos mais recentes já podem projetar sobre superfícies planas.

### 2 Teclado

Apesar da ineficiência inata do teclado QWERTY, é improvável que venham a ocorrer sérias tentativas de mudar seu layout. As teclas com molas, iguais às das máquinas de escrever, são, sem dúvida, as mais comuns, embora as teclas de efeito Hall, que usam ímãs, possam tornar-se populares. Os interruptores eletrônicos talvez sejam substituídos por um sistema baseado em chaves que interrompam uma matriz de raios laser.




### 4 Processadores alternativos

Além do processador principal de 32 bits, é provável que o micro dos anos 90 contenha processadores adicionais em forma de módulos acopláveis. Uma parte do trabalho — por exemplo, a operação de determinado periférico — pode ser "subcontratada" pelo processador principal para o subprocessador mais adequado. Módulos baratos emulariam os computadores clássicos dos anos 80, de modo que o software de qualquer máquina pudesse ser rodado.

### 5 RAM

O processador de 32 bits pode endereçar cerca de 4,3 bilhões de posições de memória, o que deixa longe o limite de 65.536 bytes imposto pelos processadores de 8 bits que trouxeram os micros para dentro das casas.

### 6 Comunicações

Ainda que as antenas em forma de prato para recepção de sinais de satélites venham a ser comuns nos anos 90, e que a maioria dos canais de telefones sejam digitalizados, em vez de analógicos, haverá necessidade de regular a velocidade de transmissão e recepção. Esses controladores desempenharão algumas das funções dos modems atuais.

### 7 Fonte de alimentação

O aumento de carga e a multiplicidade de dispositivos ligados ao microcomputador exigirão fornecimento de energia bem maior do que o atual. Haverá circuitos equalizadores e reforço de bateria recarregável, de forma que as oscilações da rede elétrica ou a falta de energia não provoquem perda ou distorção da informação.

### 8 Tela portátil

A tecnologia da tela plana, provavelmente envolvendo uma matriz de cristal líquido de ação rápida, e talvez ligada ao processador central por luz infravermelha (ou mesmo por microondas), poderá ser empregada para exibir textos e matérias gráficas. Se for sensível ao toque, também poderá assumir as funções de digitalizador.

### 9 ROM em disco

O Disco Compacto de ROM (DCROM), com raio laser para ler informação codificada opticamente, deverá substituir os cartuchos convencionais de ROM devido a sua maior capacidade — um DCROM típico tem capacidade de 4 megabytes.

### 10 Disquetes flexíveis

Até o fim da década de 80, os disquetes flexíveis competirão diretamente com os discos Winchester, tanto em velocidade quanto em densidade de armazenamento. Deverão também ter o diâmetro reduzido para menos de 3", o mínimo atual.

### 11 Painel frontal

Nos computadores antigos, os programas tinham de ser introduzidos em notação binária através do painel frontal — uma linha de luzes e interruptores que propiciava o controle sobre cada bit dos busses de endereço, dados e controle. Para os entusiastas, com experiência em código de máquina, um painel frontal ainda pode ser útil e, portanto, a idéia talvez ressurja no futuro.

### 12 Mouse infravermelho

O IBM PC-Junior já usa radiação infravermelha para transferir dados do teclado para o computador, sem ligação por cabo. Essa tecnologia poderá proporcionar a interligação de todos os periféricos, inclusive um mouse, eliminando assim o emaranhado de fios. Sem dúvida, também serão produzidos modelos para canhotos.

### 13 Microprocessador de 32 bits

Os primeiros micros pessoais baseados em microprocessadores de 32 bits apareceram em 1983, mas tiveram de operar com bus de dados de 16 ou até 8 bits para se manterem compatíveis com a memória e os chips dos periféricos, não podendo utilizar toda a potência que prometiam. Com a introdução de dispositivos como o chip 68032 da Motorola, que oferece processamento e transferência de dados de 32 bits, a velocidade e a capacidade de manipulação de dados desses componentes passarão a ser aceitas como padrão.

# As próximas gerações

**Com a introdução da tecnologia de VLSI, estamos a ponto de entrar na quarta geração de computadores. Mas os japoneses já se preparam para a quinta.**

Homens velhos não fazem revoluções, segundo o dito popular, e o diretor do projeto japonês para a criação da quinta geração de computadores parece ter levado isso ao pé da letra. Ao escolher quarenta cientistas nas dez maiores empresas privadas e nos laboratórios do governo para trabalharem com ele no Instituto de Tecnologia para Computadores de Nova Geração, em Tóquio, o dr. Kazuhiro Fuchi selecionou apenas colaboradores com idade inferior a 35 anos. O instituto, fundado em 14 de abril de 1982, com uma verba de 600 milhões de dólares (a ser despendida em dez anos), é uma *joint-venture* entre o governo e as indústrias. Empresas como a Fujitsu, Sharp e Toshiba participam desse ambicioso projeto, que pretende ultrapassar o estágio atual da tecnologia de computadores e criar máquinas muito mais avançadas.

A própria cunhagem do termo "quinta geração" tem sua origem nos progressos conseguidos no passado, levando em conta as futuras possibilidades de criação. A primeira geração de computadores caracterizou-se pelo uso de válvulas termiônicas, que se tornaram obsoletas após a invenção do transistor. Os computadores de segunda geração (os de transistores simples) foram, por sua vez, superados pelas máquinas que usavam tecnologia de Integração em Larga Escala (Large Scale Integration, LSI) e possibilitavam a montagem de muitos componentes dentro de um único chip. Atualmente (meados da década de 80), estamos no fim dessa terceira geração e até a década de 90 deveremos chegar à quarta de chips do tipo VLSI (Very Large Scale Integrated), que conterão 10 milhões de transistores por chip, ao passo que hoje o limite não passa de 250.000.

Comparado à IBM — International Business Machines, que gasta por ano quantia superior a 2 bilhões de dólares em pesquisa e desenvolvimento, o investimento japonês parece insignificante. Contudo, o desembolso de capital japonês não visa só ao lucro.

## Sociedade pós-industrial

O interesse da ciência mudou nos últimos cem anos, passando da utilização de energia em forma bruta (com a eletricidade e o motor de combustão interna) ao estudo da riqueza mais intangível — a informação. Terras, mão-de-obra, capital e indústria talvez tenham sido fontes de poder no passado, mas o futuro favorecerá aqueles que controlarem a informação. Conhecimento e processamento de informações serão as chaves da sociedade pós-industrial. Assim, essa nova sociedade precisará de um mecanismo com raciocínio automático aplicável a qualquer problema real ou área da atividade humana, com a precisão matemática de um computador. O mecanismo que está sendo construído pelos japoneses chama-se Knowledge and Information Processing System — KIPS (Sistema de Processamento de Informação e Conhecimento).

Os seres humanos têm muita habilidade para converter sinais sensoriais em formas cognitivas — por exemplo, percebem de relance a situação de uma partida de xadrez —, mas, quando se trata de tomar decisões que dependem de grande volume de dados, as limitações logo se evidenciam. As regras do xadrez são explicáveis em poucos minutos; contudo, o jogo pode se mostrar tão complexo que os grandes mestres só conseguem ver até uma dúzia de lances adiante. No entanto, em princípio, todo problema ao qual se aplica o raciocínio divide-se numa série de etapas simples, que por sua vez se resolvem mediante regras de inferência. Esse conjunto de regras, conhecido como lógica predicativa, aplica-se a todos os problemas, mas nas decisões cotidianas simples não temos consciência do processo.

**Linguagem lógica**

O PROLOG — uma abreviação de Programming Logic (Lógica de Programação) — foi desenvolvido no começo da década de 70 pelo grupo de Inteligência Artificial da Universidade de Marselha, na França. Com base em alguns dos princípios da lógica humana, essa linguagem provavelmente será usada nos computadores de quinta geração. Ela facilita o trabalho de criação e consulta de bancos de dados além de adequar-se a aplicações educacionais.

Para um especialista não basta um bom cérebro — no caso de um médico, são necessários muitos anos de experiência para acumular conhecimentos. Da mesma forma, um KIPS deve dispor de um banco de dados no qual as regras de inferência possam operar. Além do mais, o sistema precisa ser extremamente "user-friendly" (de fácil uso), para que sua operação não requeira uma equipe de especialistas. Uma máquina KIPS com a qual você possa manter uma conversa na linguagem de sua escolha deve resultar de pesquisas no campo da inteligência artificial — área de estudos bem controvertida.

As metas estabelecidas pelos japoneses englobam grande variedade de tecnologias relacionadas ao computador: hardware, software, interfaces, sistemas especializados (ver p. 72) e os problemas da inteligência artificial.

O objetivo do projeto japonês vai além dos avanços da tecnologia dos chips. Com o aumento da densidade dos transistores em circuitos integrados, os elétrons têm menos distância para percorrer entre os componentes e, portanto, os circuitos operam com

maior rapidez. Todavia, os japoneses sabem que só velocidade não basta, e esta é a razão de despenderem tanto esforço em software. Num jogo de xadrez, digamos, há tantas seqüências possíveis de movimentos (aproximadamente $10^{120}$) que o tempo necessário para explorá-las excede o período de vida que resta ao Sol. O projeto tem por objetivo a produção de uma máquina que faça 100 milhões de inferências lógicas por segundo, ou seja, 100 milhões de LIPS (Logical Inferences Per Second).

Outro meio de aumentar a velocidade seria colocar algumas funções de software no design de um chip, em vez de carregá-las na memória e processá-las através de um chip de uso geral. Essa mudança na diferença entre hardware e software é um dos aspectos mais interessantes do projeto japonês. Já existem memórias "associativas" que possuem circuitos de busca lógica embutidos nos elementos da memória. Esses dispositivos podem localizar parte dos dados apenas por seu significado, sem haver necessidade de especificar um endereço da memória.

Avanços desse tipo irão acelerar a interação dos processadores lógicos com os bancos de dados. A colocação de rotinas de programas no computador pelo processo de hard-wiring data da época do ENIAC (ver p. 140), mas as máquinas de quinta geração divergirão da arquitetura de Von Neumann num aspecto fundamental — elas vão apresentar muitos processadores diferentes trabalhando ao mesmo tempo (em paralelo), em vez de dispor apenas de uma unidade central de processamento. Isso exige maior cuidado na cronometragem e controle das operações internas, mas elimina a restrição da velocidade que a execução seqüencial de instruções impõe.

A linguagem interna escolhida para o KIPS é a PROLOG, desenvolvida na França e na Inglaterra e baseada na lógica predicativa. O KIPS terá a capacidade de comunicar-se em diversas línguas com os usuários.

Tradução de fala humana é outra meta do projeto, com o objetivo imediato de 95% de precisão. Hoje, a capacidade de reconhecer palavras soltas pronunciadas por diversos locutores perde de longe para o sucesso alcançado pela fala sintetizada. No entanto, a NEC Corporation of Japan já conseguiu criar a máquina que reconhece a voz de um indivíduo. Mas o projeto tem uma limitação: cada palavra deve ser previamente registrada por ele, a fim de que o computador possa "lembrar-se" do padrão da fala e reconhecê-lo.

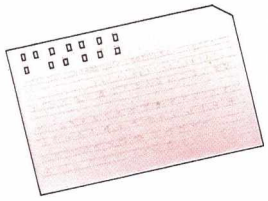
## Comunidade de informação

Quanto à palavra escrita, o projeto prevê um programa de dicionários japonês-inglês de 100.000 palavras, com margem de erro de apenas 10%.

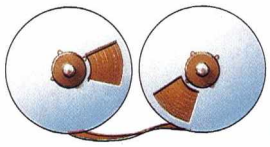
O Japão tem precedentes de sucesso em pesquisas de longo prazo: o projeto PIPS (Pattern Information Processing Systems) dos anos 70 mostrou-se útil no desenvolvimento de bancos de dados visuais e nas interfaces do tipo "user-friendly". Um KIPS precisa ser capaz de olhar para uma imagem e extrair as características e os contornos salientes para fazer uma avaliação preliminar. No metrô de Tóquio já existe uma máquina que faz o seguinte: controla os usuários através de uma câmara e produz um gráfico do fluxo de passageiros.

A tecnologia da informação movimentou, em 1983, 88 bilhões de dólares nos Estados Unidos e, com a tendência de declínio do emprego de mão-de-obra na indústria manufatureira, como na agricultura no começo do século (de 40% do total da força de trabalho para 3% hoje), a comunidade passará a ser uma sociedade de informação. Diante desse fato, o Japão faz de seu projeto de quinta geração algo muito ambicioso. O plano é otimista e inclui uma série de descobertas "programadas" que podem se concretizar ou não. Por exemplo: a descoberta da fissão nuclear controlada.

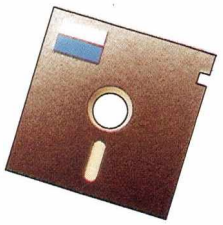
## Jogo de gerações



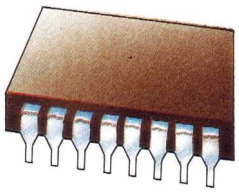
**A primeira geração** de computadores eletrônicos foi desenvolvida em torno da tecnologia da válvula termiônica. Eles possibilitavam pouca memória on-line, e os dados eram geralmente armazenados em cartões perfurados.



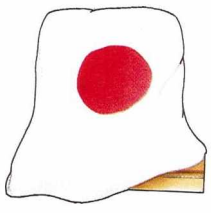
**A segunda geração** evoluiu a partir do transistor, que aumentou a capacidade de memória, embora o armazenamento off-line (em fita magnética) ainda fosse usado.



**Na terceira geração,** a invenção do circuito integrado aumentou extraordinariamente a capacidade do computador e foi responsável pelo surgimento do microcomputador — caracterizado pela unidade de disco flexível.



**Atualmente,** estamos passando da terceira para a quarta geração, com a tecnologia dos chips VLSI. A memória RAM será tão grande que o armazenamento off-line se tornará irrelevante.



**A quinta geração** de computadores, que está sendo desenvolvida no Japão, relaciona-se mais ao software que ao hardware. No entanto, baseia-se na suposição de que a memória disponível para o usuário será tão grande que o tamanho do programa perderá importância.

# BR1000

## Esse sistema modular, com a vantagem de ser multiusuário, adapta-se às necessidades da empresa.

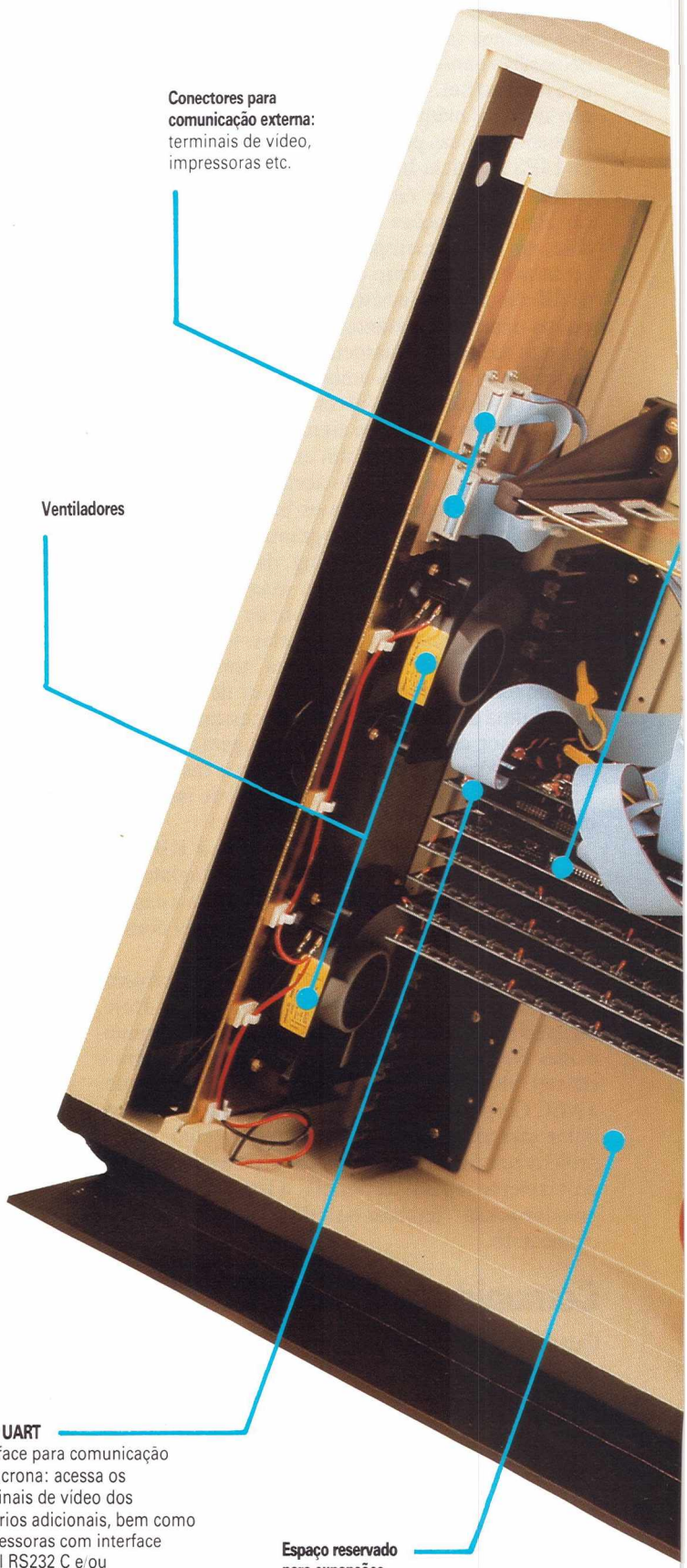
O Sistema BR1000, da Brascom Computadores Brasileiros, é um microcomputador apenas no nome e no tamanho. Voltado para aplicações comerciais e profissionais, faz quase tudo que um equipamento de maior porte executa. Trata-se de um multiusuário, ou seja, o sistema BR1000 viabiliza a utilização de seis programas diferentes ao mesmo tempo. Assim, seis departamentos ou setores distintos de uma empresa podem, por exemplo, rodar, de uma vez, programas de contabilidade, contas a receber/pagar, folha de pagamento, processamento de texto, controle de estoque etc.

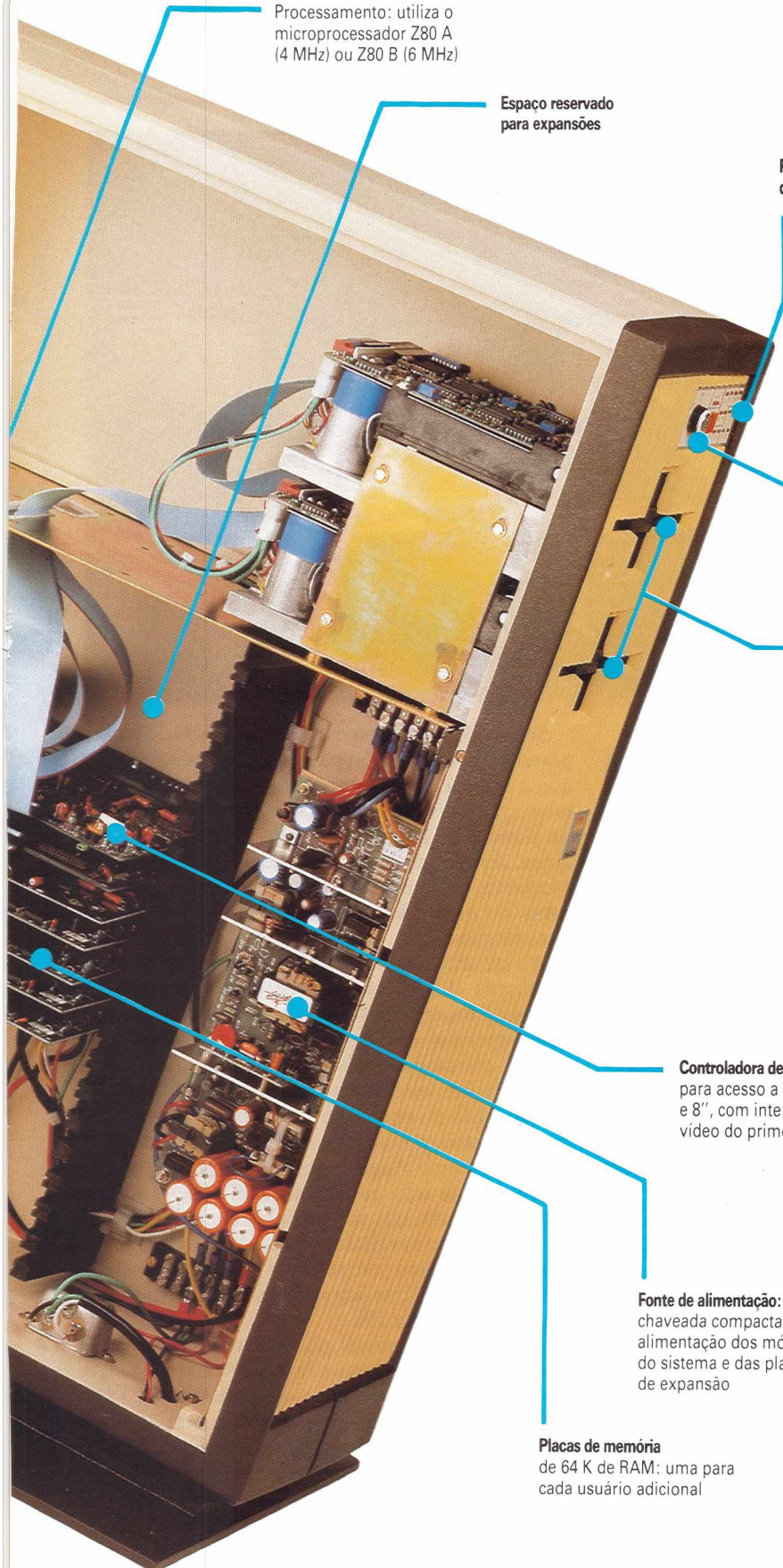
A característica principal do equipamento é a modularidade. Apresenta diversas possibilidades de expansão, conforme o volume e o tipo de informações a serem processadas.

Com memória mínima de 128 K, o BR1000 chega, na forma expandida, até as seguintes opções: seis terminais de vídeo/teclado (atingindo nesta configuração 512 K de memória); quatro unidades de disquete de 5 1/4 ou 8 polegadas; quatro unidades de disco rígido tipo Winchester de 5, 10, 16 ou 60 Mb; quatro unidades de disco magnético rígido tipo CMD de 32 ou 96 Mb; seis impressoras de diferentes velocidades e qualidades de impressão.

Dada sua modularidade e compatibilidade com o sistema operacional CP/M, o BR1000 pode ainda se ligar a terminais inteligentes, como os micros pessoais/profissionais FOXY, também produzidos pela Brascom. Este equipamento monousuário, com 64 K de memória, dispõe de uma ou duas unidades de disquete de 5 1/4 polegadas, com capacidade de 390 K cada. A ligação aos terminais de vídeo/teclado ou aos FOXY faz-se por cabos, ou por linha telefônica, quando a distância for superior a 100 m.







**Placa CPU**
Unidade Central de Processamento: utiliza o microprocessador Z80 A (4 MHz) ou Z80 B (6 MHz)

**Espaço reservado para expansões**

**Painel para monitoração do sistema**

**Chave liga/desliga**

**Unidades de drive**
para disquetes de 5 1/4", face dupla

**Controladora de disco flexível:**
para acesso a discos de 5 1/4" e 8", com interface para o vídeo do primeiro usuário

**Fonte de alimentação:** fonte chaveada compacta para alimentação dos módulos do sistema e das placas de expansão

**Placas de memória**
de 64 K de RAM: uma para cada usuário adicional

## BR1000

**MICROPROCESSADOR**

Z80 A ou Z80 B

**CLOCK**

4 MHz/6 MHz

**MEMÓRIA**

128 K de RAM estática

**VÍDEO/TECLADO**

Tubo de 12";
24 linhas x 80 colunas;
fosfatização verde;
total de 71 teclas,
com teclado numérico reduzido.

**LINGUAGENS**

BASIC compilado e interpretado; COBOL; FORTRAN IV; PASCAL; PL/I; "C"; Assembler Z80.

**PERIFÉRICOS**

Unidades de disquete de 5 1/4 ou 8";
disco rígido tipo Winchester; disco magnético tipo CMD; impressoras matriciais com velocidade de 100 até 340 cps;
impressoras lineares de 300, 600 e 900 lpm;
impressora margarida de 40 cps.

**DOCUMENTAÇÃO**

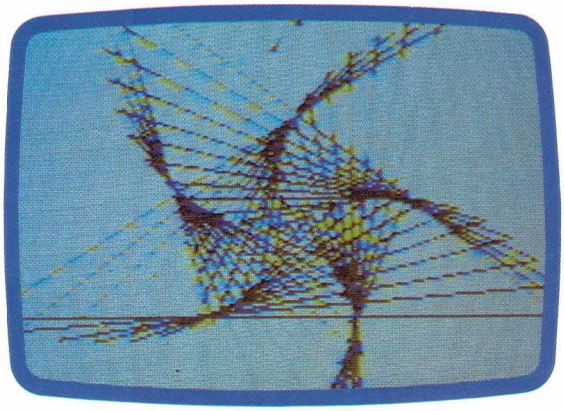
Manual de instruções sobre o sistema operacional;
manual de operação;
manuais dos gerenciadores de bancos de dados Friday ou dBase II.

# Passos da tartaruga

**O LOGO, além de ser uma poderosa linguagem de programação, contribui para mudar métodos pedagógicos tradicionais arraigados nas escolas.**







**Preparo inicial**
Antes de utilizar a linguagem LOGO para elaborar seus desenhos na tela do micro, os alunos da Escola Pequeno Príncipe participam de jogos e brincadeiras de treinamento.

É mais difícil definir a linguagem LOGO do que sentar-se diante de um microcomputador e fazer o cursor — carinhosamente chamado de tartaruga — deslocar-se para a frente, para trás e para os lados, produzindo desenhos quase mágicos. E para a criança, principal usuário dessa linguagem desenvolvida nos anos 60, a discussão não interessa. Ela encara a tartaruga como um novo jeito — muito mais divertido — de aprender.

No mundo adulto, o LOGO, além de linguagem de computação, assim como Assembler, FORTRAN, BASIC ou COBOL, vem sendo considerado a concretização de uma doutrina de ensino baseada em revolucionária ferramenta pedagógica, capaz de ousadias nunca experimentadas nos bancos escolares. E, dizem os especialistas, a criança que aprende com essa ferramenta adquire raciocínio lógico com incrível velocidade.

Segundo os professores que estão trocando lápis e borracha por microcomputadores, o LOGO permite que a criança não se torne mero receptor passivo da educação. Para eles, o LOGO desperta a criatividade do aluno, que descobre o mundo de maneira espontânea, sem as imposições da pedagogia tradicional. Incentiva, em suma, a pesquisa e a procura, precursoras de uma educação sadia, sem repressão ou constrangimento.

Ao ser criada no Instituto de Tecnologia de Massachusetts (MIT), sob a supervisão do professor Seymour Papert, essa linguagem veio dar imagem computacional aos modelos pedagógicos de Jean Piaget, psicólogo e epistemólogo suíço, que renovou os processos educacionais ao baseá-los mais na investigação do que na aprendizagem formal.

No entanto, o LOGO não se limita ao aspecto educacional. Como Piaget, essa linguagem inspirou-se no campo da inteligência artificial. Não é de admirar, portanto, que matemáticos como Andrea de Sessa, também nos Estados Unidos, tenham lançado mão da linguagem para resolver problemas de matemática avançada, registrados no livro *Turtle Geometry*, editado em 1981.

Como o LOGO possui poderosos recursos para a manipulação de palavras e listas, no campo pedagógico isso significa que, em contato inicial com a grafia, a criança pode saltar dos desenhos para o imenso leque de aplicações fornecidas por qualquer linguagem de programação completa. Sua utilização não se limita ao caminhar da tartaruga na tela, em busca de belíssimos desenhos que exigiriam de outra linguagem um estafante trabalho de programação — o

LOGO faz com que o educando entre em contato com recursos computacionais quase infinitos.

Orientado para programação modular e estruturada, o LOGO, em sua forma bruta, é representado pelos comandos, que se dividem em primitivos (os que vêm embutidos na linguagem) e procedimentos (escritos pelo usuário na memória da máquina). Com os comandos primitivos, por exemplo, pode-se traçar um quadrado na tela, dirigindo o cursor um número igual de vezes para a esquerda, para baixo, para a direita e para cima.

Executado esse trabalho, a criança pode lançar o primeiro procedimento para, por exemplo, repetir o quadrado quantas vezes desejar. Se essa for sua intenção, ela dá um nome à primeira seqüência, que passa a ser um novo comando. Assim, todas as vezes em que digitar esse nome, o programa desenhará o quadrado na tela. Elaborando mais um pouco, a criança logo tem condições de desenhar um caminhão de forma modular, que, ao sabor de sua criatividade, pode ser constituído por um retângulo (carroceria), dois círculos (rodas) e um quadrado (cabine), juntando tudo depois num único procedimento, cujo nome também fica a seu critério.

A criança mantém, assim, contato com noções de programação estruturada e princípios para a solução de problemas, partindo do pressuposto de que é mais fácil resolver várias questões pequenas do que uma grande e complexa, como seria o projeto de um caminhão na tela. Por meio do passo a passo da tartaruga, ela vai adquirindo noções importantes sobre ângulos, retas, distâncias, perspectivas etc.

A primeira tradução da linguagem LOGO no Brasil foi feita pelas professoras Heloísa Rocha Corrêa Silva e Maria Cecília Baranauskas, no Instituto de Matemática e Ciências da Computação da Universidade Estadual de Campinas, num trabalho integrado, desde 1975, com a Faculdade de Educação dessa universidade.

Com a propagação do uso de microcomputadores nas escolas do país, surgiram inúmeras software-houses que lançaram no mercado variada gama de traduções, todas com base nos princípios de Papert. Porém, a maior parte desses programas se limita à tradução dos comandos. Uma exceção é o LOGO desenvolvido pela Itautec (do grupo Itaú), que absorveu parte do trabalho da Universidade de Campinas e adaptou a linguagem para uma realidade brasileira, colocando à disposição das escolas um programa educacional com mais de 190 comandos e número ilimitado de procedimentos.

Nas grandes capitais, vários estabelecimentos de ensino já aplicam a linguagem, com particularidades e objetivos diversos em cada caso. Ao mesmo tempo, podendo ser manipulado desde os primeiros anos escolares, o LOGO permite a utilização do micro após uma série de brincadeiras e jogos desenvolvidos pelos alunos.

Na escola paulistana Pequeno Príncipe, por exemplo, as crianças são levadas ao pátio para a escolha de um animal que, a seguir, é imitado por algumas, enquanto as outras as observam. A partir daí, o grupo observador começa a fornecer comandos para o grupo imitador. Esses comandos, executados dentro de um retângulo (noção de tela), são ordens para caminhar certo número de passos para a esquerda, para a direita, e assim por diante. Depois do treinamento, os alunos se dirigem à sala de computação, onde, antes de dar comandos à tartaruga, aprendem que o computador é uma ferramenta e não propriamente um amigo, e como se comunicar com ela. Enquanto no pátio o comando era verbal, diante do micro a criança precisa de um interlocutor, ou seja, do teclado. A resposta a seu comando, agora digitada, passa a ser dada pela tela. Aí o computador se revela apenas uma ferramenta. A introdução da linguagem pode variar conforme o entendimento dos professores. Numa outra escola de São Paulo, a Graduada, que segue orientação baseada no modelo americano de ensino, coloca-se a introdução em prática com a professora fazendo o papel de tartaruga no centro de uma grande folha de papel estendida no chão, aí obedecendo aos comandos dos alunos.

Em comum todas as escolas têm a forma de colocar as crianças diante do micro, sempre destinando cada equipamento a duas ou três delas. Trata-se de um modo de não permitir que a criança se isole do contexto social da escola. Em grupos de duas ou três, elas não têm o convívio prejudicado, discutem entre si, e a máquina não as absorve. Logo após aprender os comandos primitivos e até o final do primeiro semestre, duas vezes por semana a criança manipula desenhos geométricos simples, podendo partir para recursos mais avançados, com a descoberta dos procedimentos. Nesse período importa a presença constante do professor junto aos educandos, adotando comportamento pedagógico orientado para suas necessidades.






**Doutrina LOGO**
Na Escola Graduada, são muitos os recursos à disposição das crianças educadas na doutrina LOGO. Um software com instruções na tela (foto superior) permite, por exemplo, traçar figuras e colori-las.

# Linguagem alternativa

**Encerrando nosso curso, faremos a avaliação crítica da linguagem BASIC e de algumas alternativas com relação a ela.**

Como conclusão de nosso curso de programação em BASIC, examinaremos rapidamente os pontos fortes e os fracos dessa linguagem, em comparação com outras de programação.

O BASIC é uma ramificação da linguagem em FORTRAN, uma das primeiras de programação. Ao contrário da maioria delas, o BASIC é interpretado. Isso significa que, quando um programa em BASIC é executado, um outro, especial, em algum ponto da memória do computador, interpreta o código linha por linha e converte as instruções do BASIC em código de máquina. Eis o que acontece em um pequeno programa desse tipo:

```
10 CLS
20 PRINT "DIGITE UM NUMERO"
30 INPUT X
40 PRINT "DIGITE UM SEGUNDO NUMERO"
50 INPUT Y
60 "PRINT "O PRODUTO DOS DOIS NUMEROS
   E: ";
70 PRINT X*Y
80 PRINT
90 PRINT "VOCE DESEJA OUTRO LANCE?"
100 PRINT "PRESSIONAR "S" PARA NOVA
    TENTATIVA"
110 PRINT "OU "N" PARA ENCERRAR"
120 FOR X = 1 TO 1
130 LET A$ = INKEY$
140 IF A$ <> "N" AND A$ <> "S" THEN X = 0
150 NEXT X
160 IF A$ = "S" THEN GOTO 10
170 END
```

Ao encontrar a linha 10, o interpretador BASIC opera o código de máquina necessário para limpar a tela. Na linha 20, o interpretador dá as instruções necessárias para enviar à tela a mensagem DIGITE UM NÚMERO. Na linha 30, determina o espaço de memória para armazenar um número real, aguarda o fornecimento de um item pelo teclado e a seguir converte o número digitado em código binário, armazenando-o no espaço reservado para a variável X. Todo esse procedimento se repete nas linhas de 40 a 60. Se o usuário quer repetir o programa por meio da digitação de S, o interpretador desvia de novo para a linha 10 e faz todos os cálculos e as computações novamente.

A maioria das outras linguagens é "compilada". Ou seja, após ter sido desenvolvido, o programa processa-se por um "compilador" antes de ser rodado. O compilador, um programa separado, passa pelo "código fonte" (o programa original) e produz uma segunda versão em código de máquina. Ao se processar, o programa compilado opera, possivelmente, de modo muito mais rápido que o programa interpretado, pois todas as traduções para código de máquina que exigem tempo já se terão realizado.

Uma vez que os programas compilados funcionam muito mais depressa que os interpretados, talvez você pergunte por que nem todas as linguagens empregam compiladores. Há várias vantagens no emprego de programas interpretados, como no BASIC, sobressaindo o aspecto interativo da linguagem. Isso significa que ela pode ser testada e seus erros eliminados "a partir do teclado", à medida que o programa se desenvolve. O BASIC, por exemplo, permite a inserção do comando STOP em qualquer ponto do programa. Ao encontrar essa instrução, o interpretador interrompe seu trabalho e passa a admitir a emissão de "comandos" a partir do teclado.

Comandos são instruções executáveis diretamente pelo interpretador, quando o programa não está sendo processado. O BASIC possui grande número desses comandos, que podem ser imprescindíveis na eliminação de erros. Após a execução de um programa em linguagem BASIC, isto é, quando o interpretador encontra a instrução END ou a instrução STOP, pode-se imprimir (PRINT) os valores de todas as variáveis. Experimente processar o programa da agenda de endereços, por exemplo, e digite o número 9 para desvio do programa. Se for processado por inteiro sem mensagens de erro, deverá encerrar-se com uma indicação em BASIC que, em geral, é OK, > ou *. Você então digita PRINT RMOD. O interpretador imprime o número 0 na tela (desde que você não tenha acrescentado mais algum registro!). A seguir, digite PRINT TAMA. O interpretador imprime na tela um número, uma unidade maior que o número de registros que você tiver no arquivo de dados.

## Vantagens do BASIC

Considera-se a linguagem BASIC ideal para o programador inexperiente, porque admite a eliminação dos erros por meio do teclado. Outra grande vantagem reside na facilidade de seu aprendizado. Note, por exemplo, que em nosso curso de programação em BASIC examinamos todos os pontos fundamentais da linguagem e muitos de seus aspectos mais avançados em apenas 86 páginas. Erros de sintaxe, como 40 PRINT A(12), quase sempre resultarão em mensagens de erro fáceis de ser entendidas, na execução do programa, como SYNTAX ERROR IN 40. Uma olhada na instrução referente ao número de linha mencionado é suficiente para mostrar onde se en-

contra o erro; e a retificação, comumente, consiste em digitar EDIT 40 (seguido por alguns comandos de correção) ou refazer a linha, digitando-a de modo correto. Ao encontrar um erro lógico ou de sintaxe, o interpretador BASIC interrompe a execução do programa e indica o erro. A correção de erros consiste em experimentar uma nova linha em substituição à errada e digitar RUN de novo.

## Desvantagens do BASIC

A programação em BASIC apresenta uma série de inconvenientes, alguns sutis, outros mais evidentes. Por se tratar de uma linguagem interpretada (embora existam algumas versões compiladas do BASIC), seu processamento é de lenta execução. Se rapidez não for essencial (como num programa para cálculo de conta bancária, por exemplo), a lentidão do BASIC interpretado não traz inconvenientes. Mas se a rapidez for importante (como num programa para animação de tela que emprega gráficos, ou num "relógio" para cronometrar as reações em experiência de laboratório), é muito provável que o BASIC interpretado se mostre excessivamente lento.

Se você precisar acelerar seus programas, há duas alternativas: a programação em código de máquina ou em linguagem Assembly (ver p. 448), processo difícil, que consome tempo, ou programação em linguagem compilada, como PASCAL ou FORTH. As linguagens compiladas não se mostram difíceis de aprender, mas é muito provável que o código fonte (o programa original) contenha erros, detectáveis ao se compilar o programa. As retificações não são fáceis, em comparação com as do BASIC. Depois de feitas as correções do código fonte, deve-se compilar o programa outra vez por inteiro. A maior parte dos compiladores executa duas ou três passagens pelo código fonte e cada uma delas na certa resulta em mensagens de erro, que devem ser corrigidas antes da recompilação do programa.

A produção de um programa compilado de maneira correta pode requerer muito mais tempo que a de um em BASIC interpretado. Todavia, o BASIC tem maior possibilidade de afastar o programador principiante da objetividade e da concisão, estimulando técnicas de programação viciosas, que linguagens bem estruturadas como PASCAL não admitiriam. O BASIC permite ao usuário escrever programas com pouco cuidado, cheios de GOTOs, por exemplo — são maus hábitos, que dificultam a transição para linguagens mais avançadas.

## E o que mais, após o BASIC?

A linguagem BASIC mostra-se flexível e fácil. Tem excelentes recursos de manipulação de séries, mas é lenta e não tira proveito máximo da capacidade do microcomputador. Linguagens mais modernas, como PASCAL e FORTH, apresentam bons recursos de programação, difíceis ou impossíveis no BASIC.

A linguagem PASCAL também foi criada para aprendizado e projetada especificamente para incentivar o desenvolvimento de programas "estruturados" e bem construídos. Constitui linguagem compilada, ou seja, o usuário encontrará numerosos erros detectados pelo compilador (após o código fonte ter se desenvolvido e antes que o "código objeto" compilado se processe), e isso pode ser muito frustrante. Os programadores principiantes em PASCAL consideram as restrições da linguagem — a necessidade de declarar todas as variáveis no início do programa e indicar a que tipo pertencem: reais, inteiras etc. — como uma limitação à programação livre e flexível.

Em PASCAL, o programador tem de refletir cuidadosamente sobre a estrutura lógica do programa antes de desenvolvê-lo. Nos programas em PASCAL, há grande possibilidade de numerosos erros de sintaxe no código fonte. Mas esses programas são mais bem estruturados e têm menor probabilidade de conter erros básicos de lógica.

A linguagem FORTH vem se tornando uma alternativa muito popular ao BASIC, para microcomputadores. Embora seu aprendizado não se mostre difícil como Assembly ou a linguagem de código de máquina, devemos salientar que é muito menos "intuitiva" que o BASIC ou PASCAL. Mesmo assim, a linguagem FORTH possui muitas qualidades exclusivas que a tornam séria candidata a segunda linguagem do programador.

Embora de alto nível, a linguagem FORTH processa com rapidez muito próxima à do código de máquina, em decorrência do modo excepcional com que opera. Enquanto linguagens como o BASIC têm número fixo de instruções e comandos, os usuários da FORTH podem definir seu próprio vocabulário.

A palavra-chave PRINT, em BASIC, significa que os caracteres que a seguem, colocados entre aspas, serão apresentados na tela. Na linguagem FORTH, PRINT pode ser definido para apresentar na tela, digamos, uma relação dos equivalentes hexadecimais dos códigos ASCII relativos aos caracteres de determinada variável, impressos numa coluna vertical.

A linguagem FORTH permite que o programador dê a qualquer palavra a significação que queira e apresente os resultados desejados sempre que utilizada daí por diante. FORTH não é apenas muito flexível nesse sentido; também apresenta programas que podem ser compilados para código objeto (ver p. 184), os quais são quase tão compactos e rápidos no processamento quanto os de linguagem de máquina.

Embora haja muitas linguagens de programação, a maioria dos amadores que deixam o BASIC prefere escolher entre Assembly, PASCAL e FORTH. Estas são, em resumo, as vantagens e desvantagens de cada uma delas:

**BASIC**
Aprendizado fácil
Memorização fácil
Correção de erros fácil
Execução lenta
Grande utilização de espaço na memória
Desincentivo à programação estruturada

**Linguagem Assembly**
Aprendizado não muito fácil
Memorização não muito fácil
Correção difícil
Execução muito rápida
Permissão de controle total do microprocessador

**PASCAL**
Aprendizado relativamente fácil

Memorização moderadamente fácil
Eliminação de erros mais difícil que em BASIC
Incentivo a melhores técnicas de programação
Execução mais rápida que o BASIC; mais lenta que o Assembly
Necessidade de ser compilada, o que toma tempo; uma vez compilada, processa quase tão rapidamente quanto Assembly
Possibilidade de controle relativo sobre o microprocessador, porém menos que Assembly; a manipulação de variáveis alfanuméricas não é tão fácil como em BASIC

**FORTH**
Aprendizado não muito fácil; mais fácil para os principiantes, não tão fácil para os programadores em BASIC
Memorização relativamente fácil
Correção de erros no modo interpretado muito fácil
Compilação possível; executa quase tão rapidamente quanto a linguagem Assembly
Permissão de controle total do microprocessador
Consumo de pouca capacidade de memória
Aprendizado mais fácil que a linguagem Assembly, embora menos "intuitiva" que o BASIC

## A propósito...

| Comando/Instrução/Função | Ação/Resultado | MS BASIC | Applesoft | TRS-80 | Sinclair |
|---|---|---|---|---|---|
| POINT (x,y) | Fornece a cor do ponto especificado | + | | + | |
| POKE n,m | Aloca m na posição de memória n | + | + | + | + |
| POP | Esquecer a posição de retorno do último GOSUB executado | | + | | |
| POS (n) | Fornece a coluna da posição atual do cursor | + | + | + | |
| PR # slot | Seleciona o número do slot para a saída | | + | | |
| PRINT @ n;expressão | Inicia a impressão na tela a partir da posição n | | | + | |
| PRINT TAB(n);expressão | Move o cursor para a posição n da tela e imprime | | | + | + |
| PRINT USING x$;expressão | Especifica o formato de saída da impressão | | | + | |
| PRINT lista de expressões | Imprime na tela uma lista de variáveis ou expressões | + | + | + | + |
| PRINT #-1,lista de variáveis | Grava dados em fita cassete | | | + | |
| PRINT"arquivo",lista de expr | Grava dados em um arquivo seqüencial | + | | | |
| PUT"arquivo",número | Grava um registro de um buffer para um arquivo | + | | | |
| RAND n | Atribui o valor n à variável do sistema para gerar RND | | | | + |
| RANDOM | Aciona o gerador de números aleatórios | | | + | |
| RANDOMIZE n | Aloca o valor n ao sistema para gerar um n.º aleatório | + | | | |
| READ "nomearquivo" | Dá o nome do arquivo para pegar dados com GET e INPUT | | + | | |
| READ lista de variáveis | Lê os valores da instrução DATA e os aloca às variáveis | + | + | + | |
| RECALL variável | Traz os valores da variável de uma fita cassete | | + | | |
| REM comentário | Insere comentários em um programa | + | + | + | + |
| RENUM novo, velho, incremento | Renumera as linhas do programa | + | | | |
| RESET | Reinicia informações do disquete | + | | | |
| RESET (x,y) | Desativa a posição (x,y) da tela | + | | + | |
| RESTORE | Endereça o pointer do DATA no início da lista | | + | + | |
| RESTORE linha | Permite a execução do DATA a partir da linha indicada | + | | | |
| RESUME | Faz voltar a execução onde ocorreu um erro | | + | | |
| RESUME linha | Continua a execução do programa na linha após erro | + | | + | |
| RETURN | Retorna a execução do programa para o último GOSUB | + | + | + | + |
| RETURN linha | Retorna a execução do programa para a linha desejada | + | | | |
| RIGHT$(x$,n) | Fornece o enésimo caractere mais à direita de x$ | + | + | + | |
| RND | Fornece um número aleatório entre 0 e 1 | | | | + |
| RND (0) | Fornece um número aleatório entre 0 e 1 | + | | + | |
| RND (x) | Fornece um número aleatório entre 0 e 1 | + | + | + | |
| ROT=x | Muda a orientação do desenho em alta resolução | | + | | |
| RUN "arquivo" | Executa um programa | + | + | | |

| Comando/Instrução/Função | Ação/Resultado | MS BASIC | Applesoft | TRS-80 | Sinclair |
|---|---|---|---|---|---|
| RUN linha | Executa o programa em memória a partir da linha indicada | + | + | + | + |
| SAVE "arquivo" | Grava o programa em memória com o nome definido | + | + | | + |
| SCALE=x | Fornece a escala para gráficos em alta resolução | | + | | |
| SCREEN(linha,coluna,z) | Dá o código ASCII do caractere na posição da tela | + | | | |
| SCRN(x,y) | Dá a cor do ponto (x,y) em gráfico de baixa resolução | | + | | |
| SCROLL | Empurra a imagem da tela uma linha para cima | | | | + |
| SET (x,y) | Ativa a posição (x,y) da tela | + | | + | |
| SGN (x) | Fornece o sinal de x(1,0 ou -1) | + | + | + | + |
| SHLOAD | Carrega um desenho em alta resolução da fita cassete | | + | | |
| SIN (x) | Fornece o cálculo do seno de x, x em radianos | + | + | + | + |
| SLOW | Apresenta o display de forma contínua | | | | + |
| SOUND freq,duração | Gera um som com freqüência e duração indicadas | + | | | |
| SPACE$(n) | Fornece uma variável alfanumérica com n espaços | + | | | |
| SPC(n) | Pula n espaços em uma instrução PRINT | + | + | | |
| SPEED=x | Altera a freqüência de saída dos caracteres | | + | | |
| SQR(x) | Fornece o valor da raiz quadrada de x | + | + | + | + |
| STOP | Termina a execução de um programa | + | + | + | + |
| STORE variável | Grava uma matriz em fita cassete | | + | | |
| STR$(x) | Fornece uma representação alfanumérica do valor de x | + | + | + | + |
| STRING$(n,m) | Fornece uma var alfanum de n caract de código ASCII m | + | | + | |
| STRING$(n,x$) | Idem anterior apenas com n caracteres do 1.º de x$ | + | | | |
| SWAP variável1,variável2 | Troca os valores de duas variáveis entre si | + | | | |
| SYSTEM | Retorna ao sistema operacional | + | | + | |
| TAB(n) | Posiciona na tela para a posição n | + | + | | |
| TAN(x) | Fornece o valor da tangente de x, x em radianos | + | + | + | + |
| TEXT | Retorna a tela para modo texto depois de gráfico | | + | | |
| TIME$ | Fornece data e hora | + | | + | |
| TRACE | Mostra o número da linha que está sendo executada | | + | | |
| TROFF | Desliga a marcação das linhas executadas | + | | + | |
| TRON | Liga a marcação das linhas executadas | + | | + | |
| UNLOCK "nomearquivo" | Desbloqueia um arquivo em disco | | + | | |
| UNPLOT x,y | Limpa o ponto da tela x,y mostrado em PLOTx,y | | | | + |
| USR x | Vai para sub-rotina em linguagem de máquina | | + | + | + |
| USRn (x) | Chama a sub-rotina indicada em linguagem de máquina | + | | | |
| VAL (x$) | Dá o valor numérico da variável x$ | + | + | + | + |
| VARPTR(variável) | Dá o endereço onde estão nome, valor e ind de uma var | | | + | |
| VLIN x1,x2,AT y | Desenha linha vertical na tela em baixa resolução | | + | | |
| VTAB x | Posiciona o cursor na linha x da coluna atual | | + | | |
| WAIT port,n,m | Suspende a execução do programa enquanto monitora a porta | + | + | | |
| WEND | Encerra um WHILE expressão | + | | | |
| WHILE expressão | Executa instruções de um loop enquanto expr for verdadeira | + | | | |
| WIDTH tamanho | Define o tamanho de saída da linha em n.º de caracteres | + | | | |
| WRITE "nomearquivo" | Dá o nome do arquivo em disco para as próximas saídas | | + | | |
| WRITE lista de expressões | Dá saída de dados na tela | + | | | |
| WRITE #arquivo,lista de expr | Grava dados em um arquivo seqüencial | + | | | |
| XDRAW expressão AT x,y | Desenha uma forma gráfica em alta resolução | | + | | |

# Bases sólidas

**Na história do microcomputador, os desenvolvimentos de hardware e software estão entrelaçados, e as personagens são tão importantes quanto os produtos.**

Em vários episódios da história, o ritmo das mudanças tecnológicas deixa as pessoas confusas. Mas, até hoje, nada — nem mesmo o progresso da aviação, desde Santos Dumont até o pouso na Lua — pôde se igualar à velocidade da revolução microeletrônica. O progresso, dos microprocessadores primitivos até os projetos de 16 bits de hoje, dos primeiros microssistemas até os *mainframes* de mesa, levou apenas um decênio. E a velocidade do desenvolvimento continua crescente.

Por volta de 1971, diversas novas empresas fabricantes de chips da Califórnia concluíram que um computador poderia ser alojado num pedacinho de silício. Nessa época, não havia planos grandiosos para uma revolução, e não se falava em "tecnologia da informação". A idéia era produzir um computador pequeno e barato que pudesse ser usado no controle de máquinas industriais ou elevadores, e os primeiros microprocessadores desempenharam essa tarefa a contento.

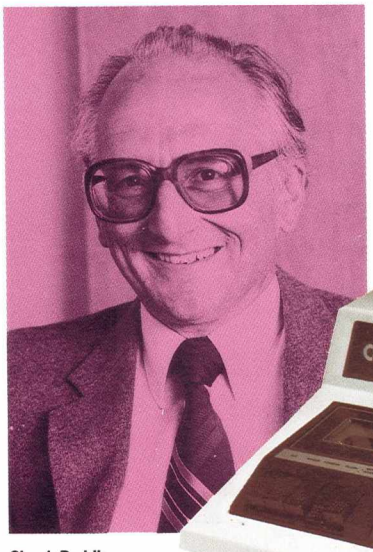
Uma das indústrias de chips, a Intel, é reconhecida como a produtora do primeiro microprocessador, denominado 4004. Os "quatro" no número referem-se a sua capacidade: era um processador de 4 bits que manipulava dados em blocos de quatro dígitos binários. Só podia usar pequenas quantidades de memória — o suficiente para um programa de controle de elevador, por exemplo.

Em 1972, a Intel desenvolveu o chip 8008 — um processador de 8 bits — e alguns hobistas começaram a pensar em construir seus próprios computadores com o novo chip. As revistas americanas especializadas em montagens eletrônicas passaram a publicar projetos dessas máquinas. Embora elas não dispusessem de monitor com tela, teclado apropriado ou outros dispositivos sofisticados, podem ser consideradas os primeiros computadores domésticos. Foi de um desses projetos que nasceu o primeiro microcomputador comercial, o Altair 8800 — vendido somente em forma de kit.

No ano seguinte, surgiria o primeiro microprocessador "de verdade", o 8080, também da Intel. Operava com blocos de dados de 8 bits e podia manipular até 64 Kbytes de memória para programas maiores. Por essa época, outros fabricantes de chips começavam a concorrer com a Intel. O 6800 da Motorola, por exemplo, fazia o mesmo que o 8080. Tinha características semelhantes de hardware, mas precisava de instruções diferentes para funcionar. Foi nesse ponto que começaram os problemas de compatibilidade de software: os programas escritos para o 8080 não podiam ser processados no 6800 e vice-versa.

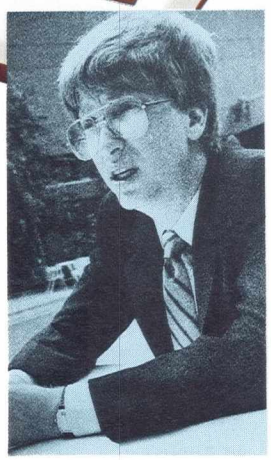
Ao mesmo tempo, outras empresas desenvolviam processadores semelhantes, entre elas a National Semiconductor, a Signestics e a Advanced Micro Devices. Mas o passo mais importante foi dado pela MOS Technology, onde trabalhava uma das principais personagens da história da computação, Chuck Peddle (ver p. 180). Ele estava na MOS quando a empresa desenvolveu um processador muito parecido com o Motorola 6800, chamado 6500. A primeira versão era tão parecida que teve de sofrer algumas modificações, e o chip revisado recebeu afinal o nome de 6502.

**Os fundadores**
Chuck Peddle projetou o Commodore PET e a sua base, o microprocessador 6502, mas a contribuição de Bill Gates como autor do BASIC da Microsoft embutido na ROM do PET foi igualmente importante.



Chuck Peddle



Bill Gates

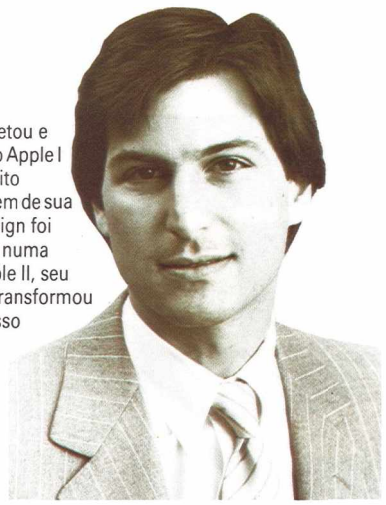
Peddle, com o *know-how* adquirido, passou-se para a Commodore — conhecida no Canadá por suas máquinas de escritório e calculadoras eletrônicas. Ele entrou na empresa com a idéia de desenvolver um computador pessoal completo, com tela, teclado, cassetes para armazenamento de programas e demais recursos que um verdadeiro computador deveria ter — logicamente, tudo construído em torno do processador 6502. A máquina surgiu em 1976 com o nome de PET 2001, um nome simpático (significa mimo), escolhido para transmitir a idéia de que o computador não era avançado demais para o usuário doméstico.

Enquanto se lançava o primeiro PET, dois outros inovadores preparavam-se para comercializar uma máquina construída numa garagem, na Califórnia. Steve Wozniak (ver p. 155) sempre quis ter um computador e, entrando para o Homebrew Computer Club, viu que seu sonho poderia se realizar. Ele pro-

jetou um computador numa única placa de circuito impresso e, com seu amigo Steve Jobs, começou a fabricar esses equipamentos e vendê-los. Chamaram a placa de Apple I. Alojada numa caixa com um teclado, a máquina acabou transformando-se no mundialmente famoso Apple II. Surgiu logo após o PET de Peddle e propiciou o aparecimento de microempresas periféricas, fabricantes de hardware e software.

A Tandy Corporation, do Texas, tinha idéias próprias para o pequeno mercado de computadores. Ela era, e continua sendo, fabricante de grande variedade de equipamentos eletrônicos, como aparelhagens de som, sintetizadores e rádios, vendendo-os em sua cadeia de lojas, a Radio Shack. O microcomputador representava uma extensão natural dessa linha de produtos. O resultado foi o TRS-80 Model 1, outro sucesso no mercado dos Estados Unidos. TRS é a abreviação de Tandy Radio Shack, mas o 80 refere-se ao microprocessador usado — o Zilog Z80. A Zilog era mais uma nova empresa fabricante de chips, e tinha produzido um processador semelhante ao Intel 8080 com melhoramentos substanciais.

Com o TRS-80 Model 1 tendo um microprocessador Z80, e o Apple II e Commodore PET um 6502, os microcomputadores começaram a apresentar diversidade em hardware. Mas com essa primeira possibilidade de escolha para o consumidor, vieram os problemas associados à incompatibilidade de máquinas e ao software não padronizado. O tipo de microprocessador usado nas primeiras máquinas é significativo porque o chip determina a escolha do software proveniente de terceiros. Enquanto o hardware se desenvolvia, também se estabeleciam padrões de software.



**A alma da empresa**
Steve Wozniak projetou e construiu o primeiro Apple I (uma placa de circuito impresso) na garagem de sua casa. Quando o design foi modificado e posto numa caixa criando o Apple II, seu amigo Steve Jobs transformou o produto em sucesso comercial.

**Steve Jobs**

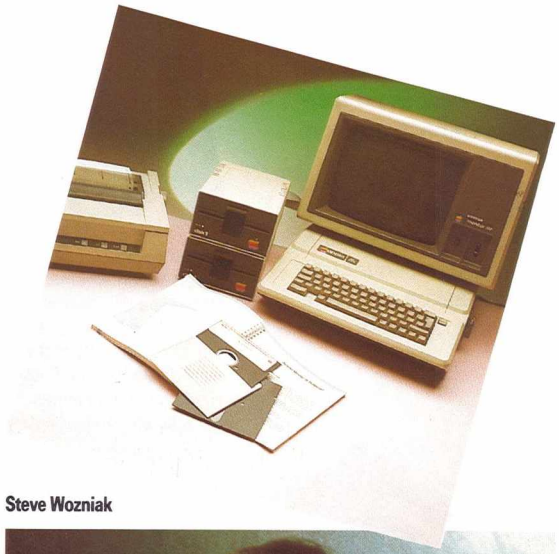
Em 1972, um jovem chamado Gary Kildall era consultor da Intel. Sua empresa, a Microprocessor Application Associates, trabalhava numa linguagem de computador que os engenheiros da Intel poderiam usar para escrever software destinado aos novos chips. Kildall achou possível ligar um microprocessador com memória a uma unidade de disco flexível de 8 polegadas e a um teletipo, a fim de dar a cada engenheiro seu próprio computador. Mas a Intel preferiu continuar sua prática de partilhar uma máquina de grande porte entre seus engenheiros.

Kildall e seu amigo John Torode, em outra garagem da Califórnia, decidiram então montar por conta própria um sistema. Torode construiu o hardware para o disco flexível poder trabalhar com o processador e Kildall escreveu o software que capacitava o processador a controlar o disco. O programa foi chamado CP/M (Control Program/Microcomputers), nome derivado do trabalho de Kildall com a linguagem de programação da Intel, a PL/M (Programming Language/Microcomputers).

O primeiro sistema operacional de disco para micros foi logo adotado pelos fabricantes de harware, que queriam instalar unidades de disco em suas máquinas. O software também influenciou o design: o CP/M só podia ser rodado pelos processadores 8080 e 8085 da Intel e pelo modelo parecido (porém mais avançado) da Zilog, o Z80. Em conseqüência, o Z80 tornou-se o chip padrão para qualquer máquina do tipo CP/M, e a compatibilidade do CP/M passou a ser a meta de todos os produtores de software.



**Gary Kildall**
Os sistemas operacionais mais recentes são desenvolvidos por grandes equipes de programadores, mas o CP/M foi escrito por Gary Kildall sozinho.
Até algumas das versões posteriores refletiam sua destinação original a um hardware grosseiro.



**Steve Wozniak**



Além de operar sistemas, os microcomputadores precisavam de uma linguagem na qual as pessoas pudessem escrever seus programas. O BASIC, desenvolvido no Dartmouth College, nos EUA, considerado uma linguagem fácil de ser aprendida, foi uma escolha óbvia.

Bill Gates, formado pela Universidade de Seattle, produziu um interpretador de BASIC para microcom-



**Adam Osborne**
Descrito por alguns como "um guarda-caça que virou ladrão de caça", Adam Osborne foi por muitos anos jornalista especializado em microcomputadores, antes de fundar sua própria empresa e produzir o primeiro computador portátil do mundo.

putadores, um programa de tradução que cabia num chip de memória limitada e podia ser incorporado a uma máquina doméstica. A empresa de Gates, a Microsoft, tornou-se expoente na produção de linguagens, tanto quanto a Digital Research no desenvolvimento de sistemas operacionais — e seu futuro ficou assegurado.

Com esse desenvolvimento, os avanços em hardware e software aplicativo seguiram o mesmo ritmo. Dan Bricklin e Bob Frankston produziram o primeiro programa de folha eletrônica, o VisiCalc, em sua empresa Software Arts. Distribuído pela Personal Software no Apple II, tornou-se o pacote gerador de aplicações mais vendido de todos os tempos, e para enfatizar sua ligação com o produto, a Personal Software mudou seu nome para VisiCorp. O WordStar, produzido pela MicroPro de Seymour Rubinstein, foi o maior sucesso de vendas no mercado de processadores de palavras compatíveis com o CP/M.

O hardware em que esses pacotes eram executados tornou-se mais barato e mais potente. Adam Osborne, que começou como escritor técnico, jornalista e editor de software, depois de se mudar da Inglaterra para os Estados Unidos lançou um computador de uso comercial de grande sucesso, com enorme quantidade de software caro, incluído no preço bastante competitivo do sistema. O inglês Sir Clive Sinclair estabeleceu novos níveis de preço com os ZX80, ZX81 e ZX Spectrum, possibilitando que milhões de usuários tivessem acesso pela primeira vez a microcomputadores.

A partir de 1982, os padrões em matéria de micros foram estabelecidos pela IBM, com o IBM PC. Essa máquina tem obtido grande sucesso junto ao público. Quase todas as softhouses e os fabricantes de periféricos estão produzindo material para o PC, o que, por sua vez, incentiva mais e mais pessoas a escolherem o equipamento.

O IBM PC reúne vários implementos pioneiros dos primórdios da indústria de microcomputadores. O microprocessador provém da Intel, que criou essa tecnologia; os sistemas operacionais são da Microsoft de Bill Gates, diversificando as linguagens, e da Digital Research de Gary Kildall; e dois dos primeiros pacotes de software colocados na máquina foram o VisiCalc e o WordStar.

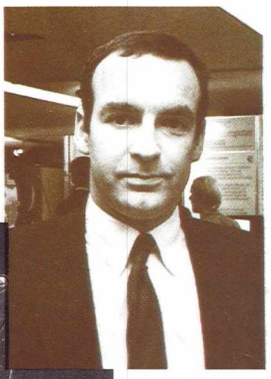


**Origem modesta**
Surpreendentemente, a tecnologia dos micros desenvolveu-se mais a partir das calculadoras programáveis (como esta Hewlett-Packard HP65) do que da antiga geração de minicomputadores.



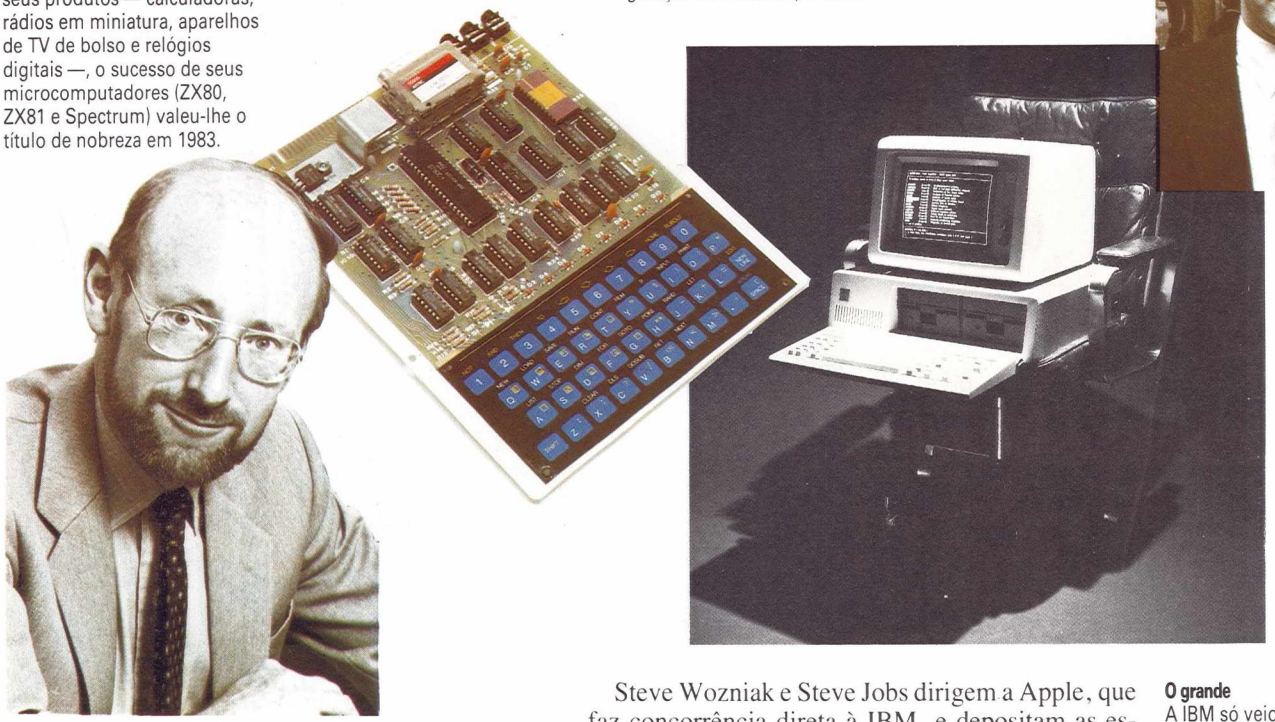
**Herman Hauser**

**Os pequenos Acorn**
A contribuição de Chris Curry e Herman Hauser (como designers e diretores dos computadores Acorn) foi valorosa. Seus micros, como o BBC, são considerados marcos fundamentais na Inglaterra.



**Chris Curry**

**Sir Clive Sinclair**
Seguindo a linha inovadora de seus produtos — calculadoras, rádios em miniatura, aparelhos de TV de bolso e relógios digitais —, o sucesso de seus microcomputadores (ZX80, ZX81 e Spectrum) valeu-lhe o título de nobreza em 1983.



Steve Wozniak e Steve Jobs dirigem a Apple, que faz concorrência direta à IBM, e depositam as esperanças da empresa na tecnologia revolucionária do Lisa e do Macintosh (uma versão reduzida do Lisa com preço em torno de 2.500 dólares). Chuck Peddle fundou sua própria empresa, a Sirius, apoderando-se de grande fatia do mercado inglês antes da chegada da IBM, mas depois disso a empresa passou a enfrentar dificuldades financeiras.

Mas, com certeza, Peddle voltará. A curta história do microcomputador mostra que seus criadores são também os sobreviventes — mesmo quando as multinacionais tentam assumir a direção do jogo.

**O grande**
A IBM só veio a aceitar a viabilidade do microcomputador em 1982. Mas quando começou a produzi-lo, tomou conta do mercado. Quase todos os novos micros de uso comercial são compatíveis com o IBM PC, para capitalizar sua enorme base de software.

# O direito ao lazer

**A grande variedade de jogos disponíveis no mercado — desde os de aventura até os de reflexão — transforma o micro em importante aparelho de entretenimento da vida moderna.**

Naves espaciais, robôs, mísseis, raio laser partindo em várias direções. Alienígenas ameaçam os habitantes da Terra. Você deve evitar que a invasão ocorra. Ou então você está num escuro labirinto subterrâneo habitado por um ogro. Não entre em pânico, procure rapidamente a saída e aperte a tecla apropriada!

Esse é o mundo dos jogos de informática, povoado por extraterrestres, monstros, dificuldades e perigos. Mas há também grandes recompensas para os vitoriosos: vencendo o computador, os jogadores tornam-se heróis.

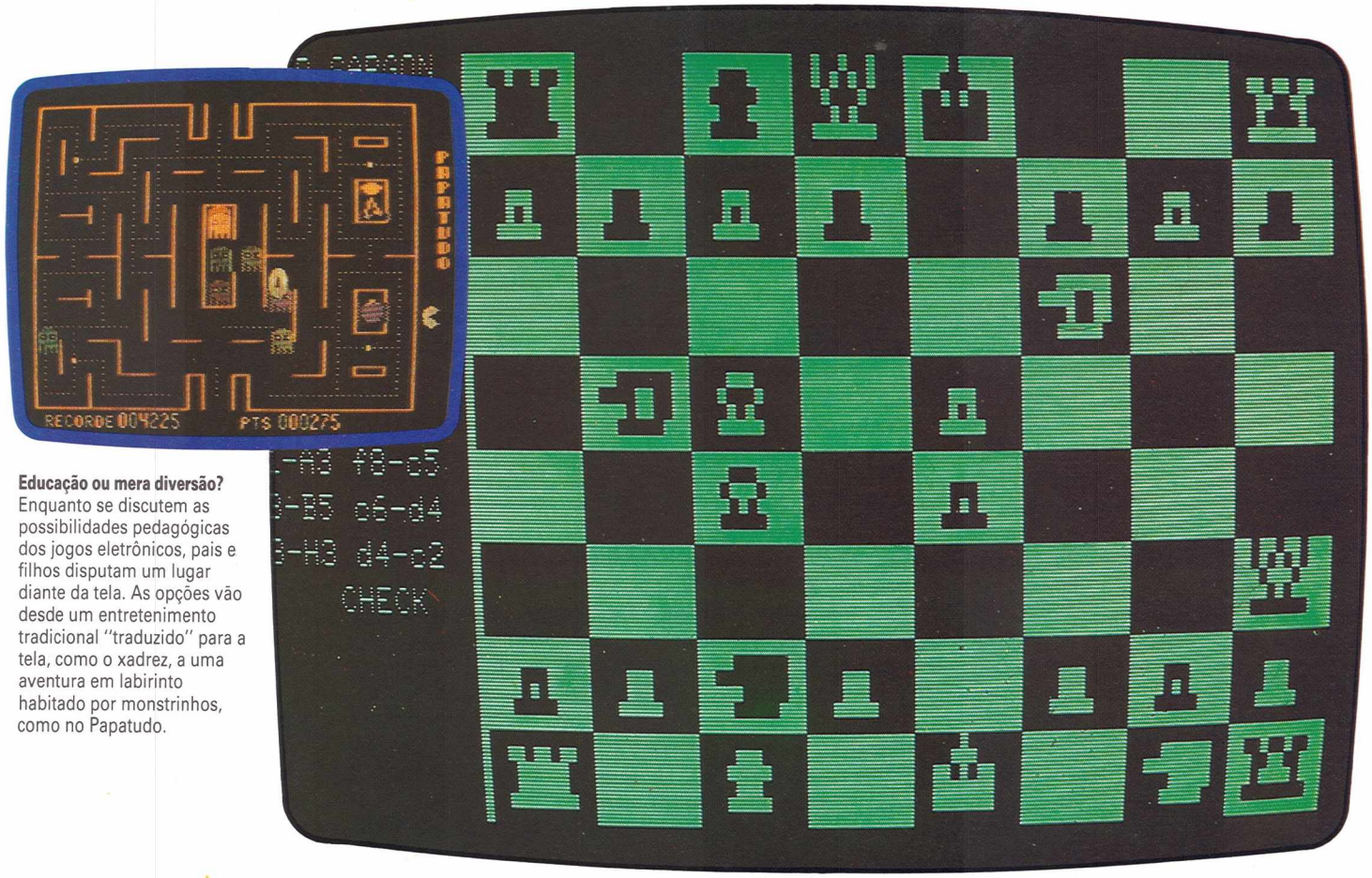
Nesse campo, os adultos competem com as crianças, mostrando-se verdadeiros entusiastas da nova coqueluche, em particular dos jogos mais complexos. A geração posterior à conquista da Lua convive tão bem com os computadores e suas brincadeiras que, sem ficar devendo nada aos pais, assimila em pouquíssimo tempo as novidades da informática.

Para público tão variado, o mercado oferece um pouco de tudo, procurando atender às solicitações educativas ou de distração. A maior parte dos produtos encontrados pertence ao que os especialistas classificam de jogos de aventura, de raciocínio ou reflexão e de animação. A partir dessas denominações genéricas, eles podem ser mais ou menos futuristas, atuais ou "históricos" (quando localizam a ação em época passada). Da mesma forma, distinguem-se pela existência ou não de recursos sonoros e visuais e por sua apresentação em fitas ou disquetes (estes, de preços bem mais altos).

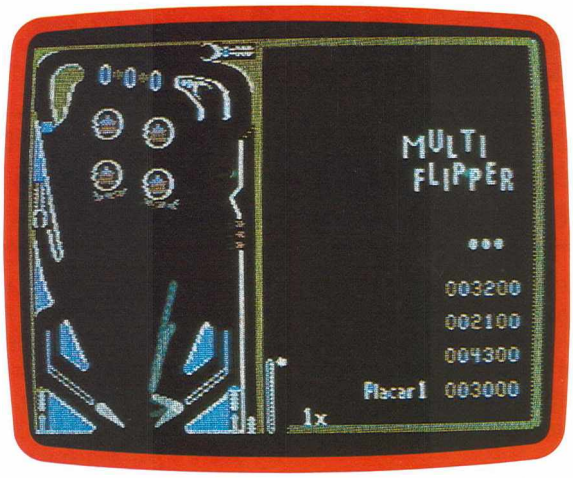
Os jogos costumam ser compatíveis com determinada categoria de computador. Em geral, encontram-se nas lojas os apropriados para Apple (Micro Engenho, Unitron etc.); TRS-80 (CP 500, Sysdata) e Sinclair (TK83, TK85 e CP 200).

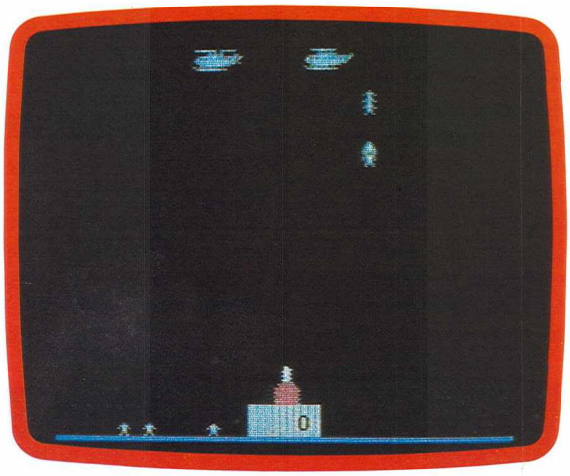
## Jogos de reflexão

Na maioria das vezes, os chamados jogos de reflexão, de raciocínio ou inteligentes são "traduções" para o computador de entretenimentos tradicionais, como xadrez, dama e jogo-da-velha. Nessa linha, existe no mercado brasileiro uma infinidade de versões. Uma delas é a do Cubo Mágico, título de um jogo da Microsoft do grupo Microdigital. Destinado




**Educação ou mera diversão?** Enquanto se discutem as possibilidades pedagógicas dos jogos eletrônicos, pais e filhos disputam um lugar diante da tela. As opções vão desde um entretenimento tradicional "traduzido" para a tela, como o xadrez, a uma aventura em labirinto habitado por monstrinhos, como no Papatudo.



**Visuais e sonoros**
Empregam-se recursos cada vez mais sofisticados nos jogos de aventura, como Sabotagem, e de animação, como o Multiflipper. Alguns reproduzem sons de naves espaciais e vozes humanas. A nova geração de jogos inclui versões em plano tridimensional.

a TK83 e TK85, em fita, esse programa permite alinhar e mover o cubo em qualquer direção, possibilitando a visualização das posições em três dimensões. O mesmo jogo recebeu uma adaptação, em fita, para CP 300 e CP 500. Visando à linha Apple, há no mercado o programa de cubo mágico em disquete.

Outro clássico do raciocínio é o gamão. Traduzido para os microcomputadores TK83, TK85, em fita, ou para Apple, em disquete, tem como objetivo testar a habilidade e a sorte do jogador.

Talvez o mais tradicional dos jogos de reflexão ainda seja o xadrez, que pode ser encontrado em fita (para CP 200, TK83 e TK85) ou fita e disquete (para CP 300, CP 500 e TK2000 Color). Em fita, a Microsoft desenvolveu o Tkadrez I e o Tkadrez II, cada um com 16 K de memória e recursos que possibilitam a análise de partidas. O Tkadrez II, mais complexo, apresenta sete níveis de dificuldades e, além de disputar a partida, mostra todos os lances efetuados, recomenda uma jogada e armazena a posição das peças, em fita, caso se queira prosseguir o jogo mais tarde.

Fazem parte ainda desse grupo a tabuada e os jogos de forca, da velha (tridimensional), de senha e de loto, entre outros, às vezes com nomes diferentes, mas com a estrutura e o conteúdo das brincadeiras dos tempos do papel e lápis.

## Jogos de aventura e animação

Você é espião dos americanos na Segunda Guerra Mundial. Sua missão: entrar no Castelo de Wolfenstein, procurar os mapas que mostram as posições alemãs e sair com eles do território inimigo. Mas tome cuidado, pois os guardas nazistas estão atentos!

Esse é um típico jogo de aventura. E, como a maioria deles, apresenta recursos visuais e sonoros sofisticados. Nesse Castelo de Wolfenstein, em disquete para Apple, conseguiu-se a reprodução quase perfeita da voz humana (os guardas, no caso, dão ordem de prisão ao invasor).

No mesmo gênero, a Play Soft desenvolveu, também em disquete para Apple, todos sonoros (com indicações em inglês), jogos como Ilhas Misteriosas, Ataque e outros em que o jogador deve vencer invasores extraterrestres, procurar um tesouro, fugir de índios ou de outras ameaças.

Os jogos de aventura e os animados muitas vezes se confundem; as diferenças são, em alguns casos, muito sutis. Jornada nas Estrelas, por exemplo, enquadra-se em ambos os gêneros. Alguns fabricantes, para simplificar, estabeleceram apenas as denominações "animados" e "de raciocínio". Um jogo animado seria o Ases do Volante para TK83 e TK85, em fita. Nessa brincadeira, você pilota um auto de corrida e precisa percorrer 2.500 km. Não pode bater nos carros a sua frente e muito menos sair da pista.

Para Apple, em disquete, existe no gênero o Simulador de Vôo II. Utilizando os recursos disponíveis, como um mapa dos Estados Unidos com todos os seus aeroportos, painel de instrumentação etc., você deve levantar vôo e pousar em segurança. Pode simular, com ajuda do programa, uma pista mais próxima ou mais distante.

Com recursos bastante avançados, reproduzindo sons de naves espaciais e vozes humanas, o Galaxy (fita e disquete) é compatível com CP 500. Já o Lode Runner, com 150 níveis de dificuldades, destina-se à linha Apple. Apresentado em disquete, permite que o usuário crie as etapas do jogo, editando com o próprio teclado. O objetivo é transportar vários barris de um lado para outro.

No gênero existem também Tiro ao Alvo, Invasores, Batalha Aérea, Caça-Níqueis (em fitas, para CP 300 e CP 500), Comando UFO, Batalha Naval, Vôo Simulado (em fitas, para CP 200), Caça ao Tesouro, Desafio Espacial, Grand Prix, Minotauro, Tubarão (para TK83 e TK85, em fitas) e Ataque, Auto-Estrada, Piratas no Espaço, Ilha das Aranhas (em fitas e disquetes para TK2000 Color).

As possibilidades dos jogos de informática são muitas e estão apenas começando. Educadores discutem o valor pedagógico do computador e de seus programas de jogos. As crianças e os adultos adeptos desses jogos, em geral indiferentes a tais debates, descobrem no dia-a-dia com a máquina novas opções de lazer. E a tendência é um aperfeiçoamento em equipamento e software. Os primeiros passos já foram dados: com o aumento da capacidade de memória dos micros, desenvolve-se uma nova geração de jogos com alta resolução gráfica e em plano tridimensional.

# Sumário

## VOLUME 2

### Chips & bytes


Jogando pelo correio ..... 266
Comunidade "ligada" ..... 301
Conforto no trabalho ..... 321
Atendendo pacientes ..... 358
Micros na advocacia ..... 374
Ficção e realidade ..... 381
Mestre-de-obras ..... 392
Micro e finanças ..... 426
Guerra na paz ..... 441
Micro e arte ..... 452
Passos da tartaruga ..... 472
O direito ao lazer ..... 481


### Conexões


Traços eletrônicos ..... 258
Claro como cristal ..... 278
Rato eletrônico ..... 296
Mordomo eletrônico ..... 314
Bastões ligados ..... 332
Plena carga ..... 352
Imprimindo a jato ..... 372
Senso comum ..... 394
Mão única ..... 414
Show de laser ..... 434


### Fundamentos


O visual dos caracteres ..... 252
Questão de segurança ..... 253
Trabalho de detetive ..... 298
Controle editorial ..... 308
Registro de trilhas ..... 324
Passo a passo ..... 348
O mapa da mina ..... 364
Autor original ..... 384
Fim específico ..... 388
Código de ordenação ..... 413
Máquina abstrata ..... 424
Novilíngua ..... 428
Código de máquina ..... 448
Linha de montagem ..... 464
As próximas gerações ..... 468


### Hardware


Memórias do passado ..... 304
Expansão dos limites ..... 326
Fora do espectro ..... 386


### Os precursores


Gottfried Leibniz ..... 260
Norbert Wiener ..... 300
Uma casa de chá ..... 320
Konrad Zuse ..... 340
Leonardo Torres ..... 360
Concorrência criativa ..... 380
Vannevar Bush ..... 400
Ma Bell ..... 420
Grace Hopper ..... 440
Desafio universitário ..... 460
Bases sólidas ..... 478


### Perspectivas


Construa seus jogos ..... 241
Controle seu percurso ..... 243
Tempo de observação ..... 248
Janelas para o mundo ..... 264
Seu fiel servidor ..... 281
Viajando ..... 341
Observando os astros ..... 346
Lance de mestre ..... 361
A melhor opção ..... 368
Coisa de criança? ..... 401
Linha de visão ..... 421
Voz de comando ..... 446
Futurologia ..... 466


### Por dentro do hardware


DGT-1000 ..... 250
Apple IIe ..... 269
Ego ..... 290
Epson HX-20 ..... 309
Commodore Vic-20 ..... 330
JR Sysdata ..... 349

# VOLUME 2


Cobra 210 ..... 370
SID 3000 ..... 390
Labo 8221 ..... 410
PC16 ..... 430
HP-85 ..... 450
BR1000 ..... 470


## Programação BASIC


Campos e registros ..... 254
Novas entradas ..... 272
Respostas aos exercícios ..... 280
Elaboração do programa ..... 292
Ampliação de arquivos ..... 316
Trocando de lugar ..... 336
Montagem de programas ..... 354
Valores fictícios ..... 376
Tempo e movimento ..... 396
Mandado de busca ..... 416
Recursos extras ..... 436
Questão de estilo ..... 456
Linguagem alternativa ..... 474


## Software


Nomes encadeados ..... 244
Um livro de figuras ..... 261
Comportamento simulado ..... 267
A ordem da jogada ..... 286
Procurando caminhos ..... 288
Quadro de avisos ..... 306
A toda velocidade ..... 328
Idiomas diferentes ..... 344
Faz de conta ..... 366
Intérprete de papéis ..... 389
Revisão eletrônica ..... 404
Gerador de aplicações ..... 406
Texto e computação ..... 408
Elementos subversivos ..... 432
Kits de ferramentas ..... 444
Descubra o código ..... 454
Risco calculado ..... 461


## Som e luz


Apresentando o som... ..... 246
... e a luz ..... 246
Dicas sobre o som ..... 276
Como criar imagens ..... 276
O ressoar do Vic ..... 284
Esclarecendo o Dragon ..... 285
Recursos modestos ..... 312
Imagens primárias ..... 312
O som ideal ..... 334
Luz-guia ..... 334

# Teste

Ao final da série MICROCOMPUTADOR — CURSO BÁSICO convém que você teste o aproveitamento de sua leitura e recorde noções fundamentais sobre o assunto. Relacionamos nesta página algumas perguntas básicas; nas seguintes estão as respostas bem como a indicação da(s) respectiva(s) página(s) de consulta para esclarecimento mais profundo.

## I Computação e micros em geral

**I.1** O que é hardware, software e firmware?

**I.2** O que é uma CPU, memórias RAM, ROM e EPROM?

**I.3** O que é um bit, um byte e o sistema hexadecimal?

**I.4** A memória principal do micro é representada por um esquema geral chamado mapa da memória. Quais seus principais elementos? O que é um buffer?

**I.5** O que é DOS? Como se divide logicamente um disco flexível?

**I.6** Quais são os principais tipos de dispositivos de armazenamento de dados utilizados pelos micros? Quais são e para que servem os periféricos em geral?

**I.7** A comunicação entre micros e seus periféricos, micros e outros micros e até mesmo micros e mainframes (grandes computadores) é muito importante. Como se dá a comunicação serial? E a paralela? O que são redes de micros e quais seus principais tipos?

**I.8** O que são sistemas analógicos e sistemas digitais? Qual sua importância?

**I.9** O que é código de barras?

**I.10** Que tipos de profissionais encontram-se na área de computação mais comumente?

**I.11** O que significa criptografia? Para que serve?

## II Software

**II.1** Quais os tipos de software mais encontrados para microcomputadores?

**II.2** O que é um programa interpretado e um programa compilado? O que significa programa fonte e programa objeto?

**II.3** Em computador, como se constitui um arquivo de dados? O que é classificação de dados? Para que serve? Cite duas técnicas de classificação.

**II.4** As técnicas de simulação encontram no microcomputador um grande aliado. O que é a simulação? Cite um exemplo.

**II.5** O que são fluxogramas?

**II.6** O que são linguagens de programação? Cite três tipos.

**II.7** Qual a vantagem e a desvantagem de utilizar a linguagem de máquina? Para que servem os comandos PEEK e POKE?

## III História dos computadores

**III.1** O que são as chamadas gerações de computadores, e quais suas principais características?

**III.2** Cite duas figuras ilustres da história da invenção do computador e duas personagens conhecidas da história recente dos microcomputadores.

## IV BASIC

Retorne aos exercícios propostos nas páginas 136, 137, 148, 149, 175, 197 e 235. As respectivas respostas estão nas páginas 149, 175, 197, 215 e 280. Se você acertar mais de 80% dos exercícios propostos, passe à interpretação do programa final da agenda computadorizada das páginas 438 e 439. Quem consegue entender a lógica do programa e a função das instruções BASIC pode se considerar um programador BASIC apto a passar de amador para profissional, com um pouco mais de fundamentos avançados e muita prática. Caso você não tenha ido bem nos exercícios já citados, recomendamos uma revisão geral do curso de BASIC, até chegar ao final da agenda computadorizada.

Desejamos boa sorte em futuras incursões aos "aprovados" e um pouco mais de trabalho aos que ainda não atingiram esse estágio.

# Respostas

## I Computação e micros em geral

**I.1** *Hardware* é o termo utilizado para designar o equipamento físico, a parte eletrônica do computador. O *software* define a parte lógica do computador, em geral representada pelo conjunto de programas e procedimentos que determinam o que o computador deve fazer. *Firmware* representa o armazenamento de software num suporte físico permanente (um chip ROM, por exemplo). Se você estranhou alguma resposta, releia as páginas de 1 a 8, e as noções ficarão mais claras.

**I.2** *CPU* (Unidade Central de Processamento) é, como o nome indica, o dispositivo central do computador, onde se executam instruções, operações e controles do processamento. As memórias eletrônicas do computador são representadas pelos tipos *RAM* (Random Access Memory), de acesso aleatório e volátil (seu conteúdo se apaga ao se desligar o computador); *ROM* (Read Only Memory), cujo conteúdo é permanente (não se apaga ao se desligar o computador e é gravado no momento de sua fabricação); *EPROM* (Erasable Programmable Read Only Memory), que pode ser gravada e apagada por meio de equipamento específico, mesmo depois de sua fabricação. Nas páginas 96 e 97 você encontrará mais esclarecimentos sobre as memórias; na 138 e na 139 há outras informações sobre o funcionamento da *CPU*.

**I.3** Um *bit* (BInary digiT) corresponde à menor unidade de informação tratada pelo computador e representada por um estado entre dois possíveis (Sim ou Não, 0 ou 1, ligado ou desligado). A representação comercial dos diferentes sinais conhecidos pelo homem é feita pelo *byte*, combinação de oito bits que permite a representação de 256 diferentes tipos de sinais. O *sistema hexadecimal* representa números na base 16. É muito utilizado por programadores por permitir a conversão de binário em hexadecimal e vice-versa com mais facilidade que o sistema decimal. Além disso, permite a manipulação de números mais extensos com menor risco de erros que em binário ou decimal. Há mais esclarecimentos a respeito de bit e byte nas páginas 32 e 33. Sobre números hexadecimais, consulte a página 179.

**I.4** O *mapa da memória* representa o conteúdo da mesma, onde se encontra o sistema operacional da máquina, a memória de tela, os dados alfanuméricos, as variáveis numéricas, o programa (em BASIC, por exemplo), o sistema de dados interno e o espaço de memória disponível para o programador ou usuário. O *buffer* corresponde a uma memória intermediária entre dois dispositivos que se comunicam. Serve para harmonizar eventuais diferenças de velocidade e processamento entre os dispositivos. Mais detalhes sobre buffers estão nas páginas 236 e 237; referências aos mapas de memória encontram-se nas páginas 329, 364 e 365.

**I.5** A sigla *DOS* representa Disk Operating System (Sistema Operacional de Disco), que é o software essencial para um microcomputador funcionar utilizando uma unidade de disco flexível ou rígido. *Discos flexíveis* dividem-se em trilhas e setores, formando assim um endereço para localização de determinado arquivo no disco. Detalhes sobre DOS e discos flexíveis estão nas páginas 324 e 325.

**I.6** O *armazenamento de dados* pode se dar em memórias de chips tipo RAM, ROM (ver pp. 96 e 97), em unidades de discos flexíveis (ver pp. 114 e 115), em unidades de fita cassete (ver pp. 94 e 95), em cartuchos ROM (ver p. 5), em unidades de disco rígido tipo Winchester (ver pp. 352 e 353) ou até mesmo em unidades de disco tipo laser (ver pp. 434 e 435). Entre os *periféricos* utilizados por microcomputadores, além dos já citados para armazenamento de dados em massa, podemos destacar os de entrada representados por teclados (ver pp. 36 e 37), joysticks e track balls (ver pp. 56 e 57), a caneta óptica (ver pp. 156 e 157), o digitalizador (ver pp. 258 e 259) e o mouse (ver pp. 296 e 297). Quanto a periféricos de saída, citam-se as impressoras tipo matricial, margarida ou de jato de tinta (ver pp. 74 e 75), o vídeo (ver pp. 132 e 133), o plotter (ver pp. 198 e 199) e o display de cristal líquido (ver pp. 278 e 279). Como periféricos e equipamentos de comunicação, é importante conhecer os modems (ver p. 108), os acopladores acústicos (ver pp. 216 e 217), os cabos co-axiais e as fitas ópticas — meios de transporte de dados (ver pp. de 301 a 303).

**I.7** A comunicação *serial* se dá pela transmissão e recepção de um bit por vez. Assim, o byte enviado e recebido deve se constituir de um conjunto de bits, que são reconhecidos por um sinal (bit) de início e de fim de um byte. A comunicação *paralela* ocorre pela transmissão simultânea de 8 bits constituindo um byte por meio de oito linhas paralelas. Os tipos mais comuns de interfaces são a RS232 (para comunicação serial) e a Centronics (para comunicação paralela). Mais detalhes sobre essas noções podem ser encontrados da página 206 à 208. Interligação entre micros constitui uma *rede de micros*, que pode

ser local, quando todos os equipamentos estão ligados numa mesma área física (um prédio, por exemplo) ou numa rede remota em teleprocessamento. Neste último caso, utiliza linhas de transmissão e sistemas de gerenciamento de redes públicas ou privadas. Os principais tipos, quanto à arquitetura de rede, são estrela, anel ou bus. Nas páginas 218 e 219 você encontra mais explicações sobre redes de microcomputadores e suas aplicações.

**I.8** *Sistemas analógicos* são aqueles que possuem mecanismos "análogos" aos dos sistemas reais que pretendem representar. A balança de dois pratos com ponteiro fisicamente conectado a seu braço constitui exemplo clássico de sistema analógico de representação do peso de objetos. Um *sistema digital* converte informações para unidade digital, passando a tratar o fenômeno que representa sob a forma digital, como nos microcomputadores que conhecemos, convertendo e reconvertendo esse modo de informação tanto na entrada quanto na saída, para que possa ser entendido pelo homem (números, textos, vozes, imagens) ou por máquinas (ação de ligar ou desligar dispositivos etc.). Nas páginas 238 e 239 você encontra mais exemplos e esclarecimentos.

**I.9** O *código de barras* é, como o nome indica, um código de representação de números e letras baseado num conjunto de listras legível por dispositivos ópticos que conseguem captá-lo seja qual for a posição relativa do conjunto de listras. Mostra-se, portanto, ideal para identificar mercadorias no caixa e estoque de supermercados. Seu uso tornou-se obrigatório nas embalagens de muitos produtos, em vários países. Na página 21 você encontra interessante artigo sobre o assunto.

**I.10** Os *profissionais* encontrados com mais freqüência nas empresas que se utilizam de computadores são analistas, programadores, digitadores e operadores. Nas empresas que fabricam computadores, são comuns engenheiros e técnicos de desenvolvimento e engenheiros e técnicos de manutenção. (Ver pp. de 101 a 103.)

**I.11** *Criptografia* é a técnica que combina a arte e a ciência de cifrar e decifrar mensagens e informações. Muito utilizada em comunicações militares e diplomáticas, vem sendo cada vez mais empregada em teleprocessamento, processamento de dados e tratamento de informação em geral por motivos de sigilo e segurança na transmissão de informações. Alguns detalhes e exemplos das técnicas mais usuais estão nas páginas 454 e 455.

## II Software

**II.1** Os *tipos de software* mais usados em microcomputadores são:

a) Processadores de texto, que servem para manipular textos longos (contratos ou estudos, por exemplo) ou textos repetitivos (cartas de convite "personalizadas").
b) Folhas eletrônicas, representadas por pacotes tipo Calc, utilizados com muita eficácia no cálculo de matrizes como projeções de demonstrativos financeiros. Em geral, o pacote inclui um recurso gráfico.
c) Gerenciadores de bancos de dados, muito utilizados em tarefas que exigem a manipulação de arquivos (de clientes, fornecedores, estoques etc.).
d) Pacotes comerciais, representados por programas de contabilidade, folha de pagamentos, faturamento, controle de estoque, contas a receber, contas a pagar e outros.
e) Programas de jogos, lazer e educacionais, cada vez mais vendidos, para uso doméstico e escolar, destinam-se tanto ao lazer quanto à educação de crianças e adultos.

Nas páginas de 6 a 8 desta série você encontra mais informações sobre os diferentes tipos de software. Vários dos artigos tratam de aplicações do computador com diversos tipos de software. Sobre editores e processadores de texto, você encontra outros informes nas páginas de 61 a 63, 308, 404 e 405. Quanto a folhas eletrônicas, consulte as páginas de 158 a 160.

**II.2** *Programa interpretado* é aquele que o computador processa a partir de um programa escrito na linguagem original de programação. Necessita, portanto, de uma "interpretação" para ser entendido pelo computador. A este tipo de programa dá-se o nome de programa fonte. Os programas originais podem ser "compilados" por meio de compiladores especiais — um tipo de software que transforma um programa fonte em programa escrito em linguagem de máquina. Este pode ser processado diretamente pelo computador, sem necessidade de interpretação. O *programa compilado* recebe o nome de programa objeto. Nas páginas 184 e 185 você tem mais referências sobre interpretadores e compiladores e sobre as vantagens e as desvantagens do processo.

**II.3** Um *arquivo de dados* é constituído de registros. Estes se formam por uma reunião de campos integrados por um conjunto de dígitos ou posições. *Classificação de dados* consiste na técnica de ordenação de dados de um arquivo por determinado critério (ordem alfabética crescente, por exemplo). Os critérios facilitam a manipulação dos dados desse ar-

quivo (para busca ou inclusão de novo elemento, por exemplo). Algumas das técnicas de ordenação mais conhecidas: a classificação bolha *(bubble sort)*, a classificação por inserção e a classificação shell *(shell sort)*, cujos detalhes podem ser vistos nas páginas 286, 287 e 413. Quanto à utilidade desta técnica, consulte as páginas 204 e 205.

**II.4** *Simulação* é a técnica de representar eventos reais em modelos reduzidos, reproduzindo seu funcionamento. Pela simulação podemos investigar fenômenos da realidade sem correr riscos de catástrofes caso o resultado não seja aquele imaginado. Um exemplo clássico e já muito usado em microcomputadores são os simuladores de vôo, com os quais pilotos podem "viajar" por meio do vídeo, decolando e aterrissando em aeroportos, e até mesmo derrubando aviões — sem que haja mortos ou feridos. Mais referências à simulação nas páginas 267 e 268.

**II.5** *Fluxogramas* correspondem a representações lógicas de um sistema, que ajudam a criar um programa com estrutura lógica. A técnica de elaboração de fluxogramas já está consagrada e tem uma simbologia quase universal para representar entradas, saídas, operações lógicas etc. Nas páginas 104 e 105 você encontra um exemplo de fluxograma e sua aplicação.

**II.6** *Linguagens de programação* são as que permitem a comunicação homem-máquina (computador, no caso), pela qual se aliam a inteligência do homem e a capacidade de processamento do computador para realizar determinadas tarefas. Existem várias linguagens de programação, cada uma desenvolvida com vistas a certo tipo de aplicação. Por exemplo: o COBOL (COmmon Business Oriented Language) é uma linguagem de programação desenvolvida para utilização em aplicações comerciais. A linguagem FORTRAN (FORmula TRANslation system) destina-se a aplicações científicas. O BASIC (Beginners All-purpose Simbolic Instruction Code) foi desenvolvido para facilitar o acesso ao computador por parte de principiantes, com uma linguagem de entendimento e manipulação fáceis para leigos em computação. MICROCOMPUTADOR — CURSO BÁSICO dedicou mais de noventa páginas ao BASIC. LOGO é outra linguagem desenvolvida pensando em tornar o uso do computador acessível a crianças. Nas páginas 26, 27, 164 e 165 encontram-se exemplos da linguagem LOGO. Sobre linguagens de programação em geral, ver as páginas 344, 345, 475 e 476.

**II. 7** A utilização da linguagem de máquina no desenvolvimento de um programa traz como *vantagem* principal a aceleração do processamento quando da execução do programa, além de permitir maior detalhamento e mais eficiência no uso da memória do computador, na formatação do vídeo etc. A principal *desvantagem* é que, para programas relativamente complexos, a tarefa de programação em linguagem de máquina pode resultar demorada e penosa. Os *comandos* PEEK e POKE são utilizados para acesso direto à memória do computador, guardando valores em determinadas posições da memória (POKE), ou recuperando valor de posições da memória (PEEK). Sobre linguagem de máquina, ver as páginas 448, 449, 464 e 465; a respeito dos comandos PEEK e POKE você encontra maiores detalhes na página 188.

## III História dos computadores

**III.1** Apesar de sua história ser recente (iniciou-se na década de 40), os computadores já passaram por diferentes *gerações:*

— *Primeira geração*: tecnologia das válvulas e dos cartões perfurados;
— *Segunda geração*: tecnologia dos transistores e das fitas magnéticas;
— *Terceira geração:* tecnologia do circuito integrado e dos discos — é onde começa a história do microcomputador;
— *Quarta geração:* tecnologia dos chips integrados em larga escala;
— *Quinta geração:* avanços do software integrado ao hardware, possibilitando dar mais "inteligência" e "capacidade de aprendizado" aos computadores.

Nas páginas 468 e 469 você encontra mais detalhes sobre as gerações de computadores.

**III.2** Blaise Pascal com sua "Pascalina" de 1642 é uma personagem importante da história "antiga" do computador. Herman Hollerith também se salientou na história da computação, pelo uso do cartão perfurado. John von Neumann, pelo desenvolvimento do projeto ENIAC em 1946-47, tornou-se elemento de destaque nesta história. Várias outras personagens — como Napier, Von Leibniz, Babbage, Boole, Turing, Shannon — mostraram-se fundamentais. Na história dos microcomputadores, Chuck Peddle com sua participação no projeto do microprocessador 6502 e no lançamento do micro PET da Commodore; Steve Jobs e Steve Wozniac com seu famoso Apple; Clive Sinclair com seus micros da linha Sinclair; Bill Gates como autor do BASIC da Microsoft e Gary Kildall como principal autor do famoso sistema operacional CP/M são nomes diretamente ligados ao fabuloso sucesso dos microcomputadores por todo o mundo. Dedicamos diversas páginas desta série aos precursores da computação em geral e da microcomputação em particular. Você encontra um resumo desta história nas páginas de 86 a 88 e de 478 a 480.

**As páginas A, B, C, e D — com o teste e as respostas — não fazem parte do volume encadernado.**

# Índice geral


## A

Ábaco 86, 179(1)
Acoplador acústico **216-217**(1)
Adaptadores de interface de periféricos
  ver PIA
Advocacia **374-375**(2)
Aiken, Howard 440(2)
ALU (Arithmetic and Logic Unit) 139(1)
Analisador diferencial 400(2)
Analistas 102-103(1)
Animação **152-154, 181-183**(1)
APPLE II PLUS 14(1), **269-271**(2)
APPLE-TRONIC 14(1)
Armamentos 243(2)
Arquitetura 453(2)
Arquivos
  ver Banco de dados
Arte **34-35, 44-45,** 141-143(1), 381-383, **452-453**(2)
Assembler 84-85, 103, 152(1), **464-465,** 475(2)
Assembly (ver Linguagem de máquina)
Astronomia **346-347**(2)
Automóvel **65**(1)

## B

Babbage, Charles 86-87, **220**(1), 381(2)
Bacon, Francis 86(1)
BAM (Block Availability Map) 324-325(2)
Banco de dados 7-8, **124-125,** 204-205(1), 264-265, 306-307, 316-317, 336-339(2)
Bardeen, John 47(1)
Barth, John 383(2)
BASIC 3, 16, 19, 84-85, 103, 184-185(1), 344, **474-477**(2)
  Comandos 474(2)
    DUMP 384(2)
    FOR-NEXT **38-40**(1)
    GOSUB **77-78,** 212(1)
    GOTO 17-19, 212(1)
    HELP 384(2)
    IF-THEN-ELSE 212(1)
    INPUT 17(1)
    LET 17(1)
    LIST 16(1)
    PEEK 178, **188**(1), 456-457(2)
    POKE 178, **188**(1), 276-277, 456-457(2)
    PRINT 17, 52(1)
    PRINT USING 53(1)
    REM 17(1)
    REPEAT-UNTIL 213(1)
    SAVE 77(1)
    TRACE 384(2)
    WHILE-DO **212-213**(1)
  Extensão **444-445**(2)
  Funções **146-148**(1)
    ASC 214(1)
    COS 147(1)
    DATA 195(1)
    DIM 194-195(1)
    INSTR 148(1)
    LEN 147(1)
    LOG 147(1)
    RND **172-175**(1), 362(2)
    SQR **146**(1)
    SGN **146-147**(1)
    SIN 147(1)
    TAN 147(1)
Baud, Émile 428(2)
Bell, Alexander Graham 420(2)
Big trak 27(1)
Biotecnologia 305(2)
BITS **32-33**(1), 428(2)
  paridade par **253,** 298(2)
Bomba
  lógica 429(2)
  relógio 429(2)
Boole, George **128-129**(1)
Boot 428(2)
BR1000 **470-471**(2)
Braço-robô 281-283, **314-315,** 447(2)
Bricklin, Dam 480(2)
Brinquedos **401-403**(2)
  educativos 401(2)
Brunner, John 382(2)
Buffer **236-237**(1), 364(2)
Bulletin Board Services (Serviço de Quadros de Aviso) 306(2)
Bus de endereços
  ver Dados - armazenamento
Bush, Vannevar 87(1), **400**(2)
Bushnell, Nolan 221(1)
Buzzwords **428-429**(2)
BYTES **32-33**(1), 253, 429(2)

## C

Cabos co-axiais 302-303(2)
Caixa automático 60(1)
Caixa-forte 92-93(1)
Calculadoras 166-167(1)
  analítica 86-87(1)
  mecânica 86-87, 92(1), 260(2)
Caneta óptica **156-157**(1)
Capek, Karel 281(2)
Caracteres
  Gerador de **252,** 365(2)
Cartões perfurados 240, 380(2)
Cartucho 5(1)
Cassete
  Fita 5-6, **94-95,** 114(1)
  Gravador 5-6, 20, 94-95, 114(1)
Cavalo de Tróia 429(2)
CCD (Charge Coupled Device) 283(2)
Chapa gráfica
  ver Digitalizadores
Chips 2, 28, 69, **121-122**(1), 388, 478(2)
Chomsky, Noam 404(2)
Cibernética 300(2)
Ciência 28, 72-73(1), 248-249, 346-347(2)
Cifra de César 454-455(2)
Cinema 142-143, 166(1), 381-383(2)
Circuito
  analógico 239(1)
  biestável (Flip Flop) 305(2)
  integrado 88, **122-123**(1)
Clarke, Arthur C. 381(2)
Classificação
  bolha 286(2)
  por inserção 287(2)
  Shell **413**(2)
COBOL 185(1), 440(2)
COBRA 210 **370-371**(2)
Código
  de barras **21,** 176(1)
  Baudot 95(1)
  cifrado **454-455**(2)
  Gray 348(2)
  Hamming **298-299**(2)
  de máquina ver Linguagem de máquina
  Morse 80(1)
COMAL 344(2)
Comércio 6-8, **41-43,** 60, 65, 158-160(1), **320**(2)
COMMODORE 64 **189-191**(1)
Computador **1-4,** 28-29, 46-47, 72-73, 86-88, 128, 155(1), 321-323(2)
  analógico **238-239**(1)
  aplicação
    Advocacia **374-375**(2)
    Arte **34-35, 44-45,** 141-143(1), 381-383, **452-453**(2)
    Ciência 28, 72-73(1), 248-249, 346-347(2)
    Comércio 6-8, **41-43,** 60, 65, 158-160(1), **320**(2)
    Doméstica **106-107**(1), 314-315, 394-395(2)
    Educação **25-27, 81-83,** 164-165(1), 472-473(2)
    Engenharia **392-393**(2)
    Indústria 65(1), 281-283(2)
    Jornalismo 408-409(2)
    Lazer 7-8, 22-23, 26, 56-57, 161-163(1), 241-242, 334-335, 361-363, 441-443(2)
    Literatura 408-409, 381-383(2)

Medicina 72-73, **126-127**(1), **358-359**(2)
Mercado financeiro **426-427**(2)
Militar 243(2)
Processador de palavras 6, 61-63, 84, 167(1), 263, 308, 374-375, 404-405, 414-415(2)
Serviços à comunidade 125(1), 264-265, 301-303, 306-307, 341-343(2)
Categorias profissionais **101-103**(1)
Digital 112-113(1), 238-239(1)
Gerações **468-469**(2)
Gíria profissional 428-429(2)
História **46-47**, **86-88**(1), 238-239, 380, 400, 420, 468-469, **478-480**(2)
Perspectiva **466-469**(2)
Programação 2-3, **66-67**, 84-85(1)
a válvula 46-47, 87-88(1)
Cordeiro, Waldemar 452(2)
Correio eletrônico 306, 446-447(2)
Corrida de cavalos 461-462(2)
CP 200 14, 50(1)
CP 300 14, **449-451**(1)
CP 500 **9-11**, 15(1)
CPU 4, 12, 112-113, **138-139**, **144-145**(1)
Criança 25-27, 81-83(1), 401-403, 472-473(2)
Criptografia 454-455(2)
Cristal líquido 278-279(2)
Curry, Chris 480(2)


# D


D-8100 **130-131**(1)
DACTRON E 15(1)
Dados
armazenamento **204-205**(1), **304-305**, 316-317(2)
endereços **144-145**(1), 448(2)
banco de 7-8, **124-125**, 204-205(1), 264-265, 306-307, 316-317(2)
classificação **286-287**(2), 413(2)
índice **244-245**(2)
recuperação **336-339**(2)
busca binária **416-419**(2)
Descartes, René 281(2)
Desenho
animado **181-183**(1)
industrial **421-423**(2)
DGT-1000 15(1), **250-251**(2)
Diagramas
isobáricos 249(2)
de Venn 129(1)
Digitalizadores 21, 45, 183(1), **258-259**(2)
Dígito binário
ver BITS/BYTES, sistema numérico - binário
Diodo de emissão de luz
ver LED
Direito autoral 193(1)
Diretório 325(2)
Disco
áudio compacto 434(2)
flexível 5-6, 20, **114-115**(1), 467(2)
óptico a laser **434-435**(2)
rígido 261, **352-353**(2)
sistema operacional de **324-325**(2)
unidade de 20(1)
vídeo 434(2)
Display 12(1), **278-279**, 466(2)
de cristal líquido ver LCD
Dispositivo de carga acoplada
ver CCD
Distel 307(2)
Dodgson, Charles 128(1)
Dongles 193(1)
DOS (Disk Operating System) **324-325**(2)
DRAGON 32 **285**(2)


# E


EAN (European Article Numbering) 21(1)
Editor **308**(2)
Eckert, Presper 440(2)
Educação **25-27**, **81-83**, 164-165(1), 472-473(2)
Brinquedos educativos **401-403**(2)
EGO **290-291**(2)
ELPPA II PLUS 15(1)
ELPPA JR. **150-151**(1)
Engenharia **392-393**(2)
EPROM (Erasable Programmable Read Only Memory) 96(1)
EPSON HX-20 **309-311**(2)
Ergonomia **321-323**(2)
ERNIE (Electronic Random Number Indicator Equipment) 462-463(2)
Erro 298-299, **384-385**, **432-433**(2)
código de Hamming **298-299**(2)
ortográfico 404-405(2)
paridade par **253**, 298(2)
teste de soma (checksum) 298(2)


# F


Fac-símile 108(1)
Fala 127, **186-187**(1), **446-447**(2)
Fibra óptica 302-303(2)
Ficção científica **381-383**(2)
Filtros 334-335(2)
Firmware 6(1)
Fita
cassete 5-6, **94-95**, 114(1)
magnética **224-225**(1), 304(2)
Flip Flop
ver Circuito biestável
Fliperama 23, 152-154(1), **241-242**(2)
Fluxograma **104-105**(1)
símbolos 104(1)
Folha eletrônica 7, **158-160**(1), 262-263, 480(2)
Forest, Lee de 87(1)
FORTH 16(1), 345, 475-476(2)
Frankston, Bob 480(2)
Fuchi, Kazuhiro 468(2)
Função randômica
ver BASIC - funções RND, números aleatórios


# G


Garbage 429(2)
Gates, Bill 479-480(2)
Geradores
de aplicações **406-407**(2)
de caracteres **252**, 365(2)
de programas 406-407(2)
Gödel, Kurt 200(1)
Gráficos **34-35**, **44-45**(1), 285, **334-335**, 452-453(2)
animação **152-154**, **181-183**(1)
Gravador cassete 20, 94-95(1)


# H


Hamming, R. W. 298(2)
Handshake 429(2)
Hardware 6, 12(1), 428(2)
Hardware complementar
ver Periféricos
Hauser, Herman 480(2)
Heinlein, Robert 381(2)
Hollerith, Herman 87, **240**(1), 380(2)
Hopper, Grace **440**(2)
HP-85 **450-451**(2)
Huxley, Aldous 383(2)


# I


I-7000 **169-171**(1)
Ícone 242, 262-263(2)
Impressora 20, **74-76**(1), **372-373**(2)
Input/Output
ver I/O
Índice 62(1), **244-245**(2)
Indústria 281-283(2)
Informação
Tecnologia da 468-469(2)
Interfaces 12, 76, 108, 113(1), **206-208**, 365(2)
paralelas 76, 113(1), 208
serial 113, **208**(1)
Interpretadores **184-185**(1)
I/O 4, **112-113**(1)
ISBN (International Standard Book Number) 21(1), 253(2)
ITAUTEC JR. 14(1)


# J


Jacquard, Joseph 86(1)
Jobs, Steve 155(1), 479-480(2)
Jogos 7-8, **22-23**, 26, **161-163**, 202-203, 221-223(1), 334-335, 361-363, 462-463(2)
de azar **461-463**(2)
de cartas 463(2)
de dados 463(2)
de estratégia **441-443**(2)
fliperama 25, 152-154(1), **241-242**(2)
históricos **442-443**(2)

de guerra **441-443**(2)
de labirinto **288-289**(2)
postais **266**(2)
de probabilidade 461-463(2)
"space invaders" **221-223**(1)
xadrez 22-23(1), **361-363**(2)
Jornalismo **408-409**(2)
Joysticks 20, **56-57**(1), **332-333**(2)
JR SYSDATA 15(1), **349-351**(2)

## K

Kelvin, Lord 238(1), 400(2)
Kemeny, John 19(1)
Kernel 365(2)
Kidder, Tracy 383(2)
Kilburn, Tom 460(2)
Kilby, Jack 123(1)
Kildall, Gary 479-480(2)
KIPS (Knowledge and Information Processing System) 468-469(2)
Kits de ferramentas 385, **444-445**(2)
Kurtz, Thomas 19(1)

## L

LABO 8221 **410-412**(2)
Laser óptico 305, **434-435**(2)
Lazer 7-8, 22-23, 26, 56-57, 161-163(1), 242-243, 334-335, 361-363, 441-443(2)
LCD (Liquid Crystal Displays) **278-279**(2)
Le Stik 332-333(2)
LED (Light Emitting Diodes) 112(1), 278(2)
Leibniz, Gottfried Wilhelm 86, 92, 119(1), **260**(2)
Leitora óptica 21(1)
Linguagem 2-3, 16(1), **344-345**(2)
de composição 142(1)
de máquina 2-3, 6, 66-67, **84-85**, 184-185(1), **364-365**, **448-449**, **464-465** (2)
de programação 6, 16, 19, 84-85, 103(1), 440, 472-473(2)
lógica 468-469(2)
Linha de retardamento de mercúrio 304(2)
LISA **261-263**(2)
LISP 345(2)
Listas 244-245(2)
Literatura **381-383**, **408-409**(2)
Lógica booleana **128-129**(1)
LOGO 16, 26, **164-165**, 178(1), 345, **472-473**(2)
Loteria esportiva 461-463(2)
Loto 461-463(2)
Lovelace, Ada 87(1)

## M

Mapa de Disponibilidade de Blocos
ver BAM
Maptel 307(2)
Máquina de Turing **424-425**(2)
Markkula, Mike 155(1)
Mauchly, John 440(2)
Maxwell, James Clerk 400(2)
MAXXI 14(1)
MBASIC 172(1)
Medicina 72-73, **126-127**(1), **358-359**(2)
Memória 2, **58-59**, 114-115(1), 304-305, 329, 364-365(2)
de bolha 243, 305(2)
criogênica 305(2)
dinâmica 305(2)
intermediária 236-237(2)
laser óptico 305(2)
de núcleo 304-305(2)
RAM 4, 12, 58-59, **96-97**(1), 365(2)
ROM 4-6, 12, 58-59, **96-97**(1), 365(2)
Mercado financeiro **426-427**(2)
Meteorologia **248-249**(2)
Micro **12-15**, 42, **166-168**, 218-219, **226-229**(1)
modelos
APPLE II PLUS 14(1), **269-271**(2)
APPLE-TRONIC 14(1)
BR1000 **470-471**(2)
COBRA 210 **370-371**(2)
COMMODORE 64 **189-191**(1)
CP 200 14, 50(1)
CP 300 14, **49-51**(1)
CP 500 **9-11**, 15(1)
D-8100 **130-131**(1)
DACTRON E 15(1)
DGT-1000 15(1), **250-251**(2)
DRAGON 32 **285**(2)
EGO **290-291**(2)
ELPPA II PLUS 15(1)
ELPPA JR. **150-151**(1)
EPSON HX-20 **309-311**(2)
HP-85 **450-451**(2)
I-7000 **169-171**(1)
ITAUTEC JR. 14(1)
JR SYSDATA 15(1), **349-351**(2)
LABO 8221 410-412(2)
LISA **261-263**(2)
MAXXI 14(1)
MICRO ENGENHO 2 14, **210-211**(1)
NEXUS 1600 14, **89-91**(1)
PC16 **430-431**(2)
SID 3000 **390-391**(2)
SINCLAIR QL **230-231**(1)
SINCLAIR ZX 81 **326-327**(2)
SPECTRUM 312, **386-387**(2)
SYSDATA JR. ver JR SYSDATA
TK85 15, **30-31**(1)
TK2000 15, **109-111**(1)
UNITRON AP II 15, **70-71**(1)
VIC-20 284, 312, **330-331**(2)
Redes **218-219**(1), 301-303(2)
MICRO ENGENHO 2 14, **210-211**(1)
Microdrive 63(1), **224-225**(2)
Microeletrônica **121-123**(1)
Microprocessador 123(1)
Microwriter **414-415**(2)
Mísseis **243**(2)
Mitchell, Edward Page 381(2)
MLOGO 165(1)
Modelos de simulação 267-268(2)
Modem **108**, 216(1), 306-307, 467(2)
Modulação 334(2)
Modulador de RF 4(1)
Monitor **132-133**(1), 321-322, 465-466(2)
de código de máquina 465(2)
Morse, Samuel 80(1)
Mouse 37(1), 261-262, **296-297**, 467(2)
Música **141-143**(1), 284(2)

## N

Napier, John 86(1)
Neuman, Max 460(2)
Neumann, John von 88, **140**, 209(1)
NEXUS 1600 14, **89-91**(1)
Numeração européia de produtos
ver EAN
Números
aleatórios **209**(1), 361-363(2)
hexadecimais **179**(1)
randômicos ver números aleatórios

## O

Operadores 102(1)
Orwell, George 383(2)
Osborne, Adam 479-480(2)
Otimização 368-369(2)

## P

Padrão Internacional de Codificação de Livros
ver ISBN
Pallamin, Vera 453(2)
Papert, Seymour 164(1), 472-473(2)
Paridade par 253, 298(2)
PASCAL 16(1), 344, 475(2)
Pascal, Blaise 86(1)
PC16 **430-431**(2)
Peddle, Chuck **180**(1), 478, 480(2)
Peopleware 428(2)
Periféricos 12, **20**, 56-57(1), 112-113, 132-133, 156-157, 198-199, 206-208(1), 365(2)
PIA (Peripheral Interface Adaptors) 365(2)
Piaget, Jean 472-473(2)
PIPS (Pattern Information Processing System) 469(2)
Pirataria **192-193**(1)
Pixel 44, 182(1)
Plotter **198-199**(1), 372(2)
Pompeu, Renato 408(2)
Portas lógicas 388(2)
AND **68-69**, 92-93, 128-129(1)
NOT **68-69**, 80, 92-93(1)
OR **68-69**, 92-93, 128-129(1)
Potenciômetro 57(1)
Poulsen, Valdemar 87(1)
Powers, James 380(2)

PPI (Programmable Peripheral Interface) 113(1)
Praxioscópio 181(1)
Processadores 467(2)
  alternativos 467(2)
  de palavra 6, **61-63**, 84, 167(1), 263, 308, 374-375, 404-405, 414-415(2)
Programadores 66, 101-103(1)
Programas 2-3(1)
  Arte **44-45**(1)
  Classificação de dados **286-287**(2)
  Comércio 21, 60(1)
  Lazer 22(1), **334-335**, 361
  Utilitários
    ver Kits de ferramentas
Projeto Ciranda 306(2)
PROLOG 468-469(2)

# Q

Quadros de aviso 306-307(2)

# R

RAM (Random Access Memory) 4, 12, 58-59, **96-97**(1), 467(2)
Rato eletrônico
  ver Mouse
Redes de comunicação
  comunitária **301-303**(2)
  local **218-219**(1), 301-303(2)
Relé 47(1)
Revisão 62(1), **404-405**(2)
Rewtel 307(2)
Robô
  doméstico **314-315**(2)
  industrial **281-283**(2)
  tartaruga **176-178**(1)
Roleta 461-462(2)
ROM (Read Only Memory) 4-6, 12, 58-59, **96-97**(1), 467(2)
Rubinstein, Seymour 480(2)
Rushent, Martin 143(1)

# S

Schultz, Klaus 141(1)
Script 142(1)
Sensor óptico 21(1)
Shannon, Claude 46, 87(1)
Shockley, William 47(1)
SID (Sound Interface Device) 334-335(2)
SID 3000 **390-391**(2)
Silício 121-123(1)
Símbolo
  de decisão 104(1)
  de procedimento 104(1)
SIMON'S BASIC **444-445**(2)
Simulação **389**
  fisiológica 389(2)
  jogos de 202-203(1), 441-443(2)
  modelos de **267-268**(2)
  programas de **366-367**, 389(2)
  de ventos 389(2)
  de vôo **201-203**(1), 389(2)
SINCLAIR QL **230-231**(1)
SINCLAIR ZX 81 **326-327**(2)
Sinclair, Clive **120**(1), 480(2)
Sintetizadores 141-143(1)
  analógicos 141(1)
  digitais 141-143(1)
  eletrônicos 141(1)
  de voz **186-187**(1)
Sistema numérico 32-33, **54-55**(1), 348(2)
  Babilônico 55(1)
  Binário 32-33, **54-55**, 68-69, **79-80**, 92-93(1)
    adição 92-93(1)
    multiplicação **119**(1)
    subtração **79-80**
  Decimal 54-55(1)
  Hexadecimal **179**(1), 464-465(2)
  Hindu 55(1)
  Romano 54(1)
Sistema de controle central **394-395**(2)
Software **3-8**, 66-67(1), 261, 428(2)
  Formas físicas de armazenamento **4-6**(1)
    Cartucho 5(1)
    Cassete 5-6, 94-95(1)
    Disco flexível 5-6, **114-115**(1)
    Disco óptico a laser **434-435**(2)
    Disco rígido 261, **352-353**(2)
    Fita magnética **224-225**(1), 304(2)
    ROM 4-6(1), 12, 58-59, 96-97(1), 467(2)
Som 141-143, **186-187**(1), 284, 334-335, 446-447(2)
  digital 141(1)
SPECTRUM 312, **386-387**(2)
Sprite graphics 152-154, 181-183(1)
Stack 364(2)
Stewart, David 462(2)
Strings 364(2)
Stibitz, George 420(2)
SYSDATA JR. ver JR SYSDATA

# T

Tabelas de validação 69(1)
Tablets ver Digitalizadores
Tabuladores **380**(2)
Tartaruga 26, 164-165, **176-178**(1), 472-473(2)
Teclado 12, **36-37**(1), **322-323**, 466(2)
Tela portátil 467(2)
Telecom Gold 306(2)
Telesoftware 265(2)
Teletexto 264(2)
Televisor 20, 132-133(1), 283, 321-322(2)
Terminais de comunicação 301-303(2)
Texto 408-409(2)
  índice 62(1)
  processamento de 6, **61-63**, 84, 167(1), 263, 308, 374-375, 404-405, 414-415(2)
  revisão 62(1), **404-405**(2)
  videotexto 125(1)
TK85 15, **30-31**(1)
TK2000 15, **109-111**(1)
Torode, John 479(2)
Torres, Leonardo **360**(2)
Trabalho 101-103(1)
Tracejador **258-259**(2)
Track ball 20, **57**(1)
Transistor 47, 88(1), 420(2)
Transporte **341-343**(2)
Trickstick **332-333**(2)
Turing, Alan 87-88, 200(1), 361, 424-425(2)
Turismo eletrônico 265(2)
Turnkey 428(2)
TV por cabo **301-303**(2)

# U

ULA (Uncommitted Logic Array) **388**(2)
Unidade Aritmética e Lógica
  ver ALU
Unidade Central de Processamento
  ver CPU
Unidade de disco 20(1)
Unidade de vídeo com teclado
  ver VDU
UNITRON AP II 15, **70-71**(1)
UNIVAC (Universal Automatic Computer) 304(2)

# V

Válvula 46-47(1)
Variáveis alfanuméricas
  ver Strings
VDU (Visual Display Unit) 296(2)
VIC-20 **284**, 312, **330-331**(2)
Vídeo 4(1), 296, 321-322(2)
Videotexto 125(1), **264-265**, 301-303(2)
Vôo simulado **201-203**(1), 389(2)
Voz 127, **186-187**(1), **446-447**(2)

# W

Wiener, Norbert **300**(2)
Williams, F. C. 304, 460(2)
Williams, Samuel B. 420(2)
Wozniac, Steve 88, **155**(1), 478-480(2)

# X

Xadrez 22-23(1), **361-363**(2)

# Z

Zuse, Konrad 87(1), **340**(2)