MANUAL DE OPERAÇÃO E LINGUAGEM

EDITELE

MANUAL DE OPERAÇÃO E LINGUAGEM

CP 400

COLOR

EDITELE

EDITELE

Editora Técnica Eletrônica Ltda.
Rua Casa do Ator, 1060
04546 — São Paulo — SP — Brasil
Cx. Postal 30141

2.ª EDIÇÃO
2.ª IMPRESSÃO

PREFÁCIO

O seu CP 400 Color é um Computador Pessoal extremamente versátil e poderoso, mas nem por isso você paga mais por ele. Na sua configuração básica, requer apenas um aparelho de TV em cores comum, para exibição de imagens e reprodução de sons, evitando assim o uso de monitores caros e, muitas vezes, desnecessários. A *performance* do CP 400 é notável: ligado ao seu televisor, ele é capaz de gerar imagens compostas de textos ou figuras, usando nove cores diferentes; reproduzir 255 tons musicais distintos com duração controlável; realizar cálculos matemáticos complexos; processar textos e muitas outras coisas que, com o tempo, você irá descobrir. Além disso, ele permite controlar arquivos de dados gravados em fitas cassete comuns. Basta para isso um gravador ligado ao micro e algumas instruções simples em COLOR BASIC. Assim você terá uma cópia permanente de seus programas e dados importantes.

Mas seu Computador não pára por aí. Ele ainda permite muito mais, como o uso de **cartuchos** pré-programados para a realização de jogos de vídeo (*videogames*), programas de aplicação e até novas linguagens de programação. A maior parte dos cartuchos requer o uso de *joysticks*, que são controladores de movimento compostos por uma alavanca e um botão, muito usados principalmente nos *videogames*. O seu CP 400 permite o acoplamento de até dois joysticks ao mesmo tempo.

Como você vê, o seu Computador CP 400 Color pode ser utilizado para os mais diversos fins, e mesmo que com o tempo você venha a achar que os 16 kB de memória são insuficientes para as aplicações que você deseja, eles podem ser expandidos para 32 kB, duplicando sua capacidade. Se, mesmo assim, você ainda preferir um sistema mais sofisticado, o seu CP 400 pode ser conectado a uma unidade de disco, com até dois drives.

Em resumo, o seu Computador CP 400 Color cresce com você, de acordo com suas necessidades. Você pode começar com a unidade básica e, aos poucos e à medida que domina a máquina, ir complementando seu sistema de Computação com unidades de disco, impressora etc. Portanto, prepare-se para conhecer todos os poderosos recursos de seu novo Computador. Leia atentamente a Seção I deste livro, onde são apresentados os detalhes de instalação e operação do CP 400.

Como usar este livro

Este livro está dividido em quatro seções básicas para que você possa entender melhor seus recursos. A primeira seção descreve o Computador em seus módulos básicos, mostra como instalá-lo corretamente e como ligá-lo a outros equipamentos. Descreve ainda a maneira correta de operá-lo e como regular imagem e som no seu televisor. Leia esta seção mesmo que você já conheça a linguagem COLOR BASIC, pois o CP 400 Color pode ser prejudicado por uma instalação ou operação incorreta.

A segunda e a terceira seções falam sobre a linguagem COLOR BASIC. Como você já deve saber, a linguagem BASIC é a mais simples e por isso a mais difundida linguagem de programação entre os computadores pessoais. O seu CP 400 "fala" essa linguagem e a Seção II mostra todos os recursos normais dessa poderosa linguagem. Já a Seção III aborda os recursos especiais do CP 400 Color, explicando as fantásticas instruções que estão a sua disposição no seu Computador e que não são padrão no mercado.

A Seção IV constitui-se de vários apêndices, onde toda a informação sobre o CP 400 está condensada e distribuída de uma forma racional. No final do livro você encontrará um Índice Remissivo para localizar facilmente qualquer informação que lhe interesse.

SUMÁRIO

SEÇÃO I — OPERAÇÃO

SEÇÃO II — NOÇÕES DE PROGRAMAÇÃO

SEÇÃO III — PROGRAMAÇÃO AVANÇADA

SEÇÃO IV — APÊNDICES

SEÇÃO

I

OPERAÇÃO

Leia com muita atenção esta parte do livro. O seu CP 400 Color é um equipamento sofisticado que requer alguns cuidados especiais com relação à instalação e operação. Não deixe de ler esta Seção, mesmo que você já conheça a linguagem COLOR BASIC. Você encontrará no final do Capítulo 1, dois programas para ajustar as cores e o som de seu televisor.

CAPÍTULO 1

apresentação

O seu CP 400 consiste em:

- Unidade central com um teclado de 55 teclas para introdução de programas e dados.
- Uma interface para TV que permite ligar o Computador a qualquer aparelho de TV em cores padrão PAL-M, para exibição de imagem e reprodução de sons.
- Um microprocessador 6809E responsável por todo o controle e processamento interno.
- Um Interpretador de Linguagem BASIC residente em ROM (*Read Only Memory* — Memória Apenas para Leitura).
- Uma Memória de Acesso Direto ou RAM (*Random Access Memory*) para armazenar programas e dados no Computador (16 ou 32 quilobytes).

Um quilobyte (kB) equivale a 1 024 "caracteres" ou bytes de informação.

- Um conector para *cartuchos de programas* com carregamento instantâneo de: jogos, finanças domésticas, educação e outros (os cartuchos são adquiridos separadamente).
- Dois conectores para *joysticks* (alavancas de controle) para divertimento ou aplicações especiais (os joysticks são vendidos separadamente).
- Uma interface para gravador de fita cassete. A fita pode ser utilizada para armazenar permanentemente seus programas ou dados importantes. Pode ser utilizado qualquer gravador comum de fita.
- Uma interface para impressora que permite cópias impressas de seus relatórios ou programas (a impressora é opcional).

INSTALAÇÃO E CONEXÕES

Retire o seu CP 400 da embalagem, cuidadosamente. Remova todo o material de apoio na embalagem e guarde-o para o caso de precisar transportar o equipamento mais tarde. Separe também todos os cabos que acompanham o equipamento.
Coloque o Computador sobre uma mesa perto do televisor que você pretende usar. Uma tomada elétrica apropriada deve estar próxima, assim se evita o uso de extensões.

Não ligue ainda o Computador à rede elétrica!

CONEXÃO COM A TV

A conexão do seu Computador à TV é realmente muito simples, como mostra a figura 1. O cabo de TV que acompanha o equipamento deve ser ligado à tomada indicada com TV, na traseira do

seu CP 400. A outra extremidade deve ser ligada aos terminais VHF (antena) de seu televisor.

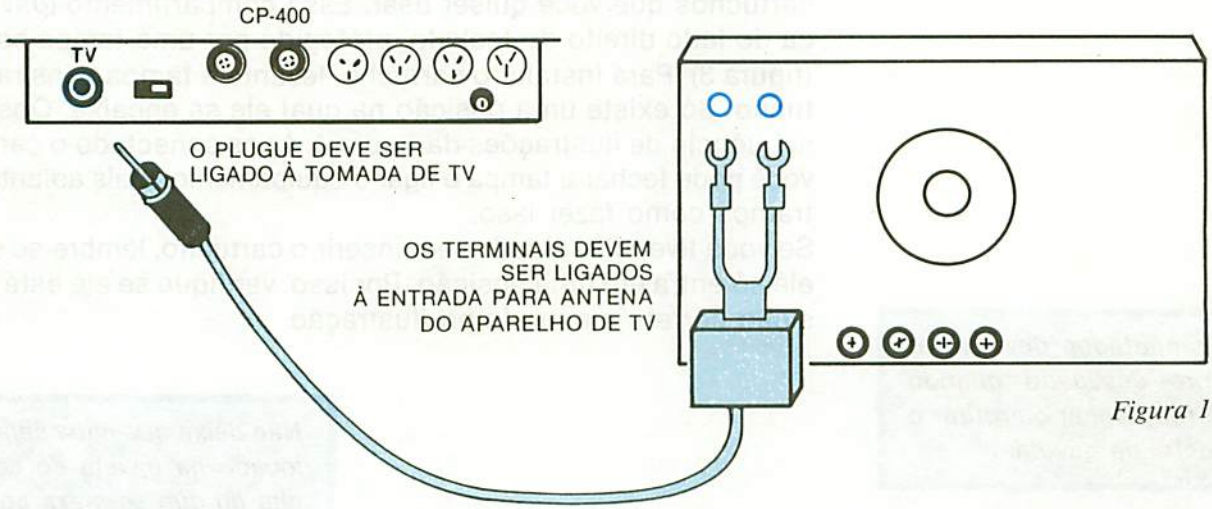


Figura 1

Deve ser desligada a antena normal do aparelho de TV (interna ou externa) para que não haja interferência na imagem.

CONEXÃO COM UM GRAVADOR

O cabo de conexão com o gravador que acompanha o equipamento possui em uma das extremidades três plugues. É essa extremidade que será ligada ao gravador.

1. Conecte o cabo na tomada K-7 na traseira do Computador.
2. Conecte o plugue menor da outra extremidade à tomada REMOTE do gravador.
3. Conecte o plugue cinza maior (da mesma cor que o plugue pequeno) à tomada AUX do gravador.
4. Conecte o plugue preto (ou de cor diferente dos demais) à tomada EAR do gravador.

Nota: *Esta conexão está detalhadamente explicada no Capítulo 7.*

Você não precisa conectar o gravador agora, a menos que queira gravar um programa ou carregar uma fita pré-gravada.

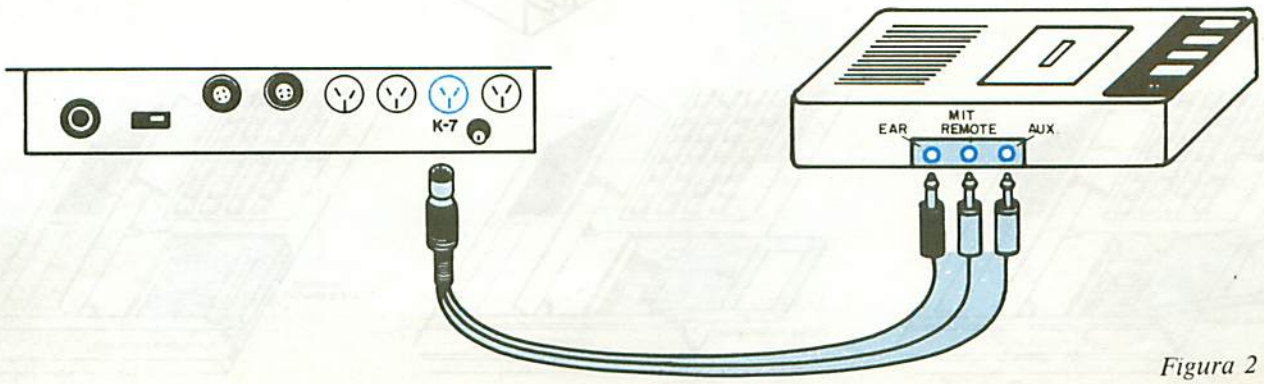


Figura 2

CONEXÃO DE UM CARTUCHO

O seu CP 400 possui um compartimento especial para alojar os cartuchos que você quiser usar. Esse compartimento (*gaveta*) fica do lado direito do teclado, protegido por uma tampa especial (figura 3). Para instalar o cartucho, levante a tampa e insira o cartucho. Só existe uma posição na qual ele se encaixa. Observe a seqüência de ilustrações da figura 4. Após conectado o cartucho, você pode fechar a tampa e ligar o equipamento. Mais adiante mostramos como fazer isso.

Se você tiver dificuldades em inserir o cartucho, lembre-se de que ele só entra em uma posição. Por isso, verifique se ele está na posição correta mostrada na ilustração.

O Computador deve estar sempre desligado quando você for colocar ou retirar o cartucho da gaveta.

Não deixe que nada seja colocado na gaveta do cartucho ou que se mexa no encaixe do cartucho ou, ainda, no conector. Isso danificaria permanentemente o equipamento.

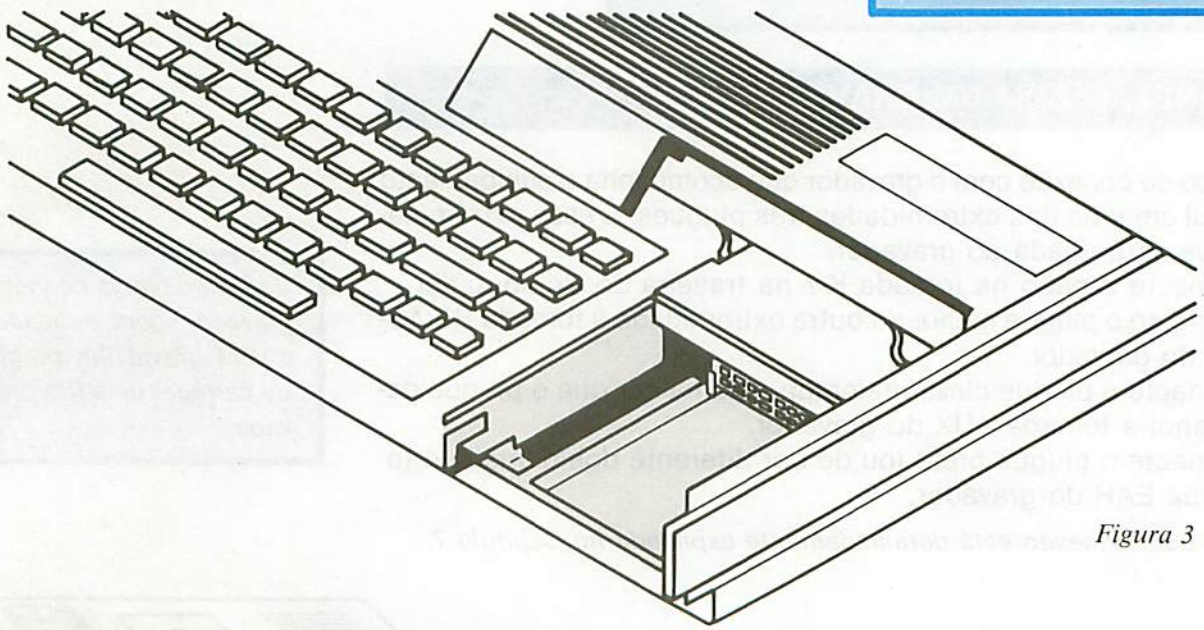


Figura 3

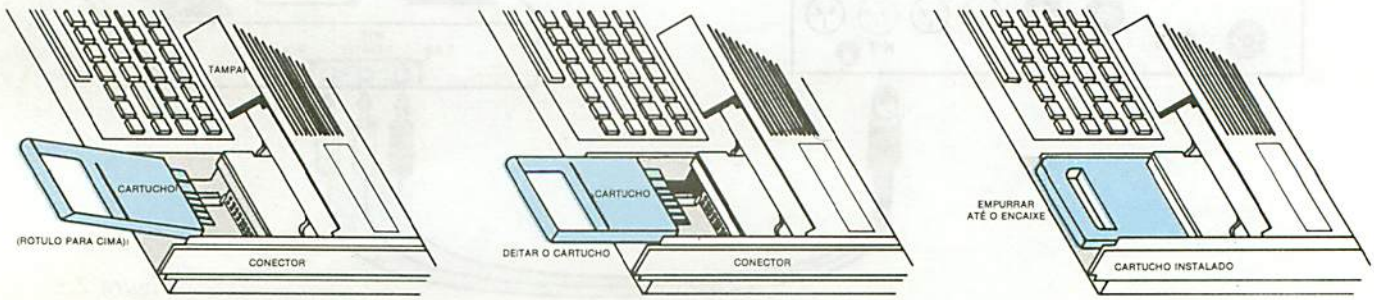


Figura 4

CONEXÃO DE OUTROS PERIFÉRICOS

Seu CP 400 permite a ligação de outros periféricos além do gravador, cartucho e TV. Periféricos são equipamentos auxiliares, geralmente responsáveis pela realização de uma só tarefa, como a exibição de mensagens, o armazenamento de dados etc.
Qualquer que seja o periférico que você vai usar (por exemplo, uma impressora), a primeira providência é certificar-se de que todos os aparelhos estão desligados (TV, computador e periférico).
Conecte os periféricos às tomadas apropriadas na traseira do Computador (veja na figura 5 as localizações corretas). Maiores detalhes são fornecidos com o próprio periférico.

PAINEL TRASEIRO DO CP-400

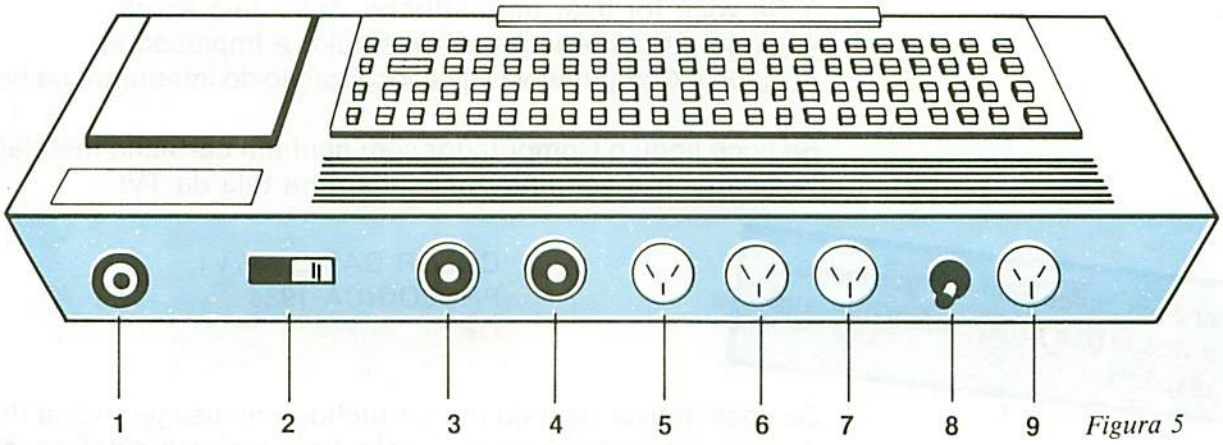


Figura 5

1. **Tomada da TV** — Conectar o plugue do cabo da TV aqui. A outra extremidade deve ser conectada na entrada VHF da antena de seu aparelho de TV.
2. **Seletor de canal** — Através dessa chave você pode escolher entre usar o canal 3 e o 4. Você deve testar as duas posições para verificar qual permite maior nitidez de imagem.
3. **Joystick direito** — O controlador joystick conectado aqui será o direito.
4. **Joystick esquerdo** — O controlador joystick conectado aqui será o esquerdo.
5. **Tomada RS 232** — Conectar a impressora serial ou outro equipamento compatível aqui.
6. **Monitor** — Conectar o cabo do monitor aqui.
7. **Gravador K-7** — Conectar o plugue do cabo do gravador aqui.
8. **Interruptor** — Essa chave liga e desliga o Computador. Mantenha-o sempre desligado durante as conexões.
9. **Fonte** — Conecte o cabo de força que acompanha o Computador aqui.

OPERAÇÃO

COMO LIGAR

O Computador e todos os periféricos devem estar desligados.

As instruções a seguir explicam como ligar e usar o seu CP 400 sem a necessidade de cartucho ou uma unidade de disco. Se você pretende utilizar um desses dois recursos, consulte a bula ou manual correspondente.

1. Ligue o televisor e coloque o volume num nível normal de audição.
2. Selecione o canal 3 ou 4 no televisor. Faça o mesmo no Computador. A chave seletora está na parte traseira de seu CP 400 (veja a figura 5).
3. Se você for usar um cartucho, conecte-o agora.
4. Ligue os periféricos (por exemplo, a impressora).
5. Ligue o Computador (veja a localização do interruptor na figura 5).

Se você ligou o Computador sem nenhum cartucho instalado, deve aparecer a seguinte mensagem na tela da TV:

v.r é um par de números que indicam a versão (v) e revisão (r) de que você dispõe.

```
COLOR BASIC  V v.r
PROLOGICA 1984
OK
```

Se você estiver usando um cartucho, a mensagem varia de acordo com o programa gravado nele. Para maiores detalhes, consulte a bula de operação do cartucho utilizado.

Se a mensagem não aparecer:

1. Confira se a televisão está funcionando.
2. Verifique os ajustes de brilho, contraste e sintonia fina em seu TV.
3. Se mesmo assim a mensagem não surgir, desligue todo o sistema, confira as conexões, e comece tudo de novo. Se persistir o problema, veja no Apêndice B, "Informações Técnicas", a tabela de defeitos e providências.

Não ligue ou desligue qualquer periférico enquanto o Computador estiver ligado. Isso pode acarretar uma operação anormal do mesmo. Mas, se isso ocorrer acidentalmente, desligue e ligue o Computador ou, então, pressione as duas teclas RESET ao mesmo tempo. Isso é feito para regularizar a operação do sistema.

TECLAS RESET

Para recomeçar um programa em cartucho, você não precisa desligar e ligar o Computador. Basta pressionar as duas teclas vermelhas marcadas RESET, que estão à direita do teclado.

Pressionando RESET você não limpa o conteúdo da memória. Se você estiver usando o BASIC, após pressionar RESET você ainda terá seu programa intacto na memória.

COMO DESLIGAR

A regra básica para desligar o seu Sistema é que nada deve ser desligado (ou ligado) enquanto o Computador estiver ligado. Isso quer dizer que deve-se **desligar os periféricos sempre depois** do Computador.

Se você desligar o Computador por qualquer razão, deixe-o assim por algum tempo (aproximadamente 15 segundos), para que seu circuito eletrônico se descarregue completamente.

Quando você desligar o Computador, todos os dados e programas que você introduziu ou carregou na memória RAM são apagados (ou "esquecidos"). Se você tiver dados ou programas que precisam ser "salvos", grave-os em fita antes de desligar o Computador. Para isso, veja o Capítulo 7.

USANDO O TECLADO

O teclado de seu Computador permite que você introduza texto, números, símbolos especiais e alguns caracteres de controle. A disposição das letras é muito parecida com a de uma máquina de escrever comum. Como nessas máquinas, o que estiver em cima nas teclas com dois símbolos deve ser digitado juntamente com as teclas SHIFT (que correspondem às teclas de maiúsculas das máquinas comuns). Portanto, para introduzir um "!", pressione SHIFT 1.

Tudo o que você digitar vai sair em maiúscula na tela. O seu Computador não representa letras minúsculas na tela, mas, se você precisa dessas letras para um relatório a ser impresso numa impressora serial, por exemplo, você pode utilizá-las. Para tanto, basta digitar SHIFT 0, e tudo o que você digitar será interpretado internamente e armazenado como minúscula e, para que você possa identificá-las na tela, as letras serão maiúsculas verdes em fundo preto, o contrário do normal. Você ainda poderá introduzir letras maiúsculas em seu relatório, usando SHIFT junto com a letra desejada. Para voltar ao modo "só de maiúsculas", digite novamente SHIFT 0.

O BASIC não reconhece nenhum comando digitado em minúscula. Por isso, quando for usar o BASIC, passe para o modo só de maiúsculas.

TECLAS ESPECIAIS

Como você já deve ter notado, existem algumas teclas no seu Computador que não existem nas máquinas de escrever. Essas teclas realizam funções especiais, como o SHIFT, que acabamos de explicar. Eis aqui uma explicação rápida dessas teclas.

TECLA	FUNÇÃO
←	Retrocesso e limpeza do último caractere (letra) digitado.
SHIFT 0	Muda o modo de introdução (só maiúsculas/maiúsculas e minúsculas).
SHIFT →	Introduz um colchete direito].
SHIFT ↓	Introduz um colchete esquerdo [.
SHIFT @	Pausa no programa. Pressione qualquer tecla para continuar.
SHIFT ↑	Introduz uma seta para trás ←.
SHIFT ←	Limpa a linha que está sendo introduzida e recomeça a introdução.
CLEAR	Cancela a linha atual, limpa a tela e coloca o cursor (o bloco brilhante) no canto superior esquerdo.
ENTER	Conclui uma introdução de linha. O Computador não interpreta nenhuma linha até que essa tecla seja usada.
BREAK	Interrompe a operação ou programa em andamento e prepara o Computador para novo comando pelo teclado.

JOYSTICKS

Usando os joysticks você pode controlar o Computador apenas movimentando a alavanca em uma dada direção. Os jogos com o Computador são as aplicações mais comuns destes controladores. Porém, eles podem ser utilizados para aplicações mais "sérias" que necessitem que você posicione uma certa informação na tela. O desenho de organogramas, fluxogramas ou circuitos elétricos na tela são excelentes exemplos dessas aplicações.

Os joysticks operam apenas com os programas escritos para eles.

Em termos de conexão, os joysticks podem ser trocados entre si, mas não com qualquer outro periférico do Computador (veja a figura 5).

Cada joystick é composto por uma alavanca de controle, que pode ser movimentada em qualquer direção, e um "botão" usado para indicar a posição desejada. Em outras palavras, você posiciona através da alavanca e pressiona o botão indicando que é aquela posição da tela que você quer. Isso é muito comum nos jogos, quando você posiciona a sua "arma espacial", por exemplo, e "dispara", utilizando o botão.

MANUTENÇÃO

Seu Computador necessita de poucos cuidados com sua manutenção. O ideal é mantê-lo sempre limpo e livre de poeira (principalmente o teclado).
Para limpar o console do seu CP 400, utilize um pano úmido, livre de fiapos.
Os periféricos, especialmente a impressora, geralmente precisam de cuidados especiais. Verifique no manual de cada um deles para maiores detalhes.
Vamos lhe ensinar agora como regular o seu televisor para obter a melhor imagem e o melhor som. Se preferir, você pode passar para o próximo capítulo.

TESTE DE AJUSTE DE COR

Para esse ajuste, o programa a seguir coloca as oito cores disponíveis (além do preto que, na realidade, é a ausência de cor) em oito barras verticais na tela. Ligue o Computador e digite o programa tentando seguir à risca a listagem abaixo:

```
NEW ENTER
 5 CLS 8 ENTER
10 FOR X = 0 TO 63 ENTER
20 FOR Y = 0 TO 31 ENTER
30 C = INT (X/8 + 1) ENTER
40 SET (X,Y,C) ENTER
50 NEXT Y, X ENTER
60 GOTO 60 ENTER
```

Se você cometer qualquer erro de digitação, pressione ENTER e repita a linha.

Seu TV mostrará na tela oito barras verticais nas seguintes cores: verde, amarelo, azul, vermelho, cinza, ciano (azul esverdeado), roxo e laranja. Com essa imagem você pode usar os controles de matiz, brilho, contraste e sintonia de seu televisor para regular a imagem conforme preferir.

Pressione BREAK quando quiser parar o programa.

Muitas vezes, para melhorar a imagem basta inverter os fios da antena.

UM COMENTÁRIO SOBRE CORES

Use o teste de cor para conseguir a melhor nitidez e separação entre as cores. Os tons gerados pelo seu aparelho podem não ser exatamente os descritos aqui, mas é importante que haja uma boa separação entre os vários tons.

TESTE DE SOM

O programa a seguir reproduz, através de seu aparelho de TV, toda a faixa de sons que seu Computador possui. Coloque o volume do seu televisor no nível normal de audição e introduza esse programa:

```
NEW ENTER
10 FOR X = 1 TO 255 ENTER
20 SOUND X,1 ENTER
30 NEXT X ENTER
```

Novamente, se você cometer qualquer erro de digitação, simplesmente pressione ENTER e repita a linha toda.

Verifique novamente o volume e digite:

```
RUN ENTER
```

SEÇÃO

II

NOÇÕES DE PROGRAMAÇÃO

A linguagem BASIC é uma das mais simples e difundidas linguagens de programação para microcomputadores pessoais no mundo.
Nesta Seção, você conhecerá as principais características dessa poderosa ferramenta de programação. Aprenderá como construir os seus próprios programas, como melhorá-los e como guardá-los para uso posterior. Esse conhecimento será indispensável para que você passe adiante e descubra as incríveis instruções apresentadas na Seção III.

CAPÍTULO 2

conheça seu computador

O Computador já foi ligado? Está pronto para receber o primeiro teste?
Neste capítulo apresentaremos você ao seu Computador. Mostraremos alguns de seus talentos e truques e também o que fazer para entendê-lo. Depois de terminar este capítulo, você estará pronto para programar. Não se preocupe com nada além da última linha que aparecer na tela. Ela deverá dizer:

```
OK
```

Todas as letras que você digitar deverão ser PRETAS com um fundo VERDE. Se estas estiverem invertidas (VERDE com um fundo PRETO), pressione SHIFT *e* 0, *para voltar ao modo normal.*

OK é o "pronto" do Computador. Ele está dizendo: "OK, eu estou pronto". Ele espera pacientemente por um comando seu. Você é o chefe. Você pode dizer a ele qualquer coisa que quiser.

Dê a ele seu primeiro comando. Digite exatamente o seguinte:

```
PRINT "OLA', EU SOU SEU COMPUTADOR COLORIDO"
```

Sempre que terminar de introduzir uma linha, pressione a tecla ENTER.

Quando você alcançar o fim da linha na tela, continue digitando. O resto da mensagem aparecerá na próxima linha, automaticamente.

Siga o cursor. Você pode digitar qualquer coisa onde ele estiver.

Agora verifique sua linha. Você colocou as aspas onde nós pedimos? Se você cometeu algum engano não há problema. Simplesmente pressione a tecla ← e o último caractere que você digitou desaparecerá. Pressione novamente e o penúltimo também desaparecerá. E assim por diante.
Pronto? Em sua tela deve aparecer:

```
OK
PRINT "OLA', EU SOU SEU COMPUTADOR COLORIDO"
OLA', EU SOU SEU COMPUTADOR COLORIDO
OK
```

Seu Computador obedece a seu comando e imprime a mensagem. Dê a ele outra mensagem para imprimir. Digite:

```
PRINT "1"
```

Pressione ENTER ... O Computador novamente obedece e imprime:

```
1
```

Tente outra:

```
PRINT "1 + 1" ENTER
```

O Computador obedece.

```
1 + 1
```

Você provavelmente esperava alguma coisa diferente, talvez a soma. Assim, experimente digitar sem as aspas. Digite:

```
PRINT 1 + 1 ENTER
```

Então o Computador imprime a resposta:

```
2
```

Estas aspas devem significar alguma coisa. Experimente algo mais com elas.
Digite cada uma dessas linhas:

```
PRINT 2 + 2 ENTER
PRINT "2 + 2" ENTER
PRINT "2 + 2 E' IGUAL A" 2 + 2
PRINT 8/2 "E' 8/2" ENTER
PRINT "10/4" ENTER
PRINT 10/4 ENTER
```

Você chegou a alguma conclusão?

STRINGS X NÚMEROS

String (conjunto ou seqüência de letras, números e/ou caracteres especiais).
O Computador vê tudo que você digitou depois de PRINT como String *ou* Números. *Se estiver entre aspas é uma* String *e o Computador a vê* exatamente *como é. Se não estiver entre aspas é Número. Assim o Computador analisa como um problema matemático.*

""O Computador considera as aspas como um jornalista. Se a mensagem está entre aspas o Computador deve imprimir exatamente como aparece. Se não estiver entre aspas, o Computador pode interpretá-la como uma soma, subtração, multiplicação ou divisão, isto é, uma operação matemática.

O REI DAS CONTAS

Qualquer problema aritmético é fácil para seu Computador. Deixe-o fazer uma divisão longa. Digite:

```
PRINT "5432 DIVIDIDO POR 12.3 E' "5432/12.3 ENTER
```

Deixe-o fazer também um problema de multiplicação:

```
PRINT 1589*23 ENTER
```

Note que o sinal de multiplicação do Computador é um asterisco, e não o sinal X que você sempre usou em matemática. Isto porque o Computador é uma máquina muito precisa e faria confusão se o mesmo sinal fosse usado tanto para multiplicação quanto para representar uma letra.
Experimente mais alguns exemplos:

```
PRINT "15*2=" 15*2 ENTER
PRINT 18*18 "E' O QUADRADO DE 18" ENTER
PRINT 33.3/22.82 ENTER
```

O ponto é usado pelo Computador para separar os inteiros das casas decimais de um número. E por isso é chamado de ponto decimal.

Freqüentemente pode aparecer mensagens estranhas em sua tela. Digite esta linha escrevendo de modo errado a palavra PPRINT:

```
PPRINT "TUDO BEM?" ENTER
```

O Computador imprime:

```
? SN ERRO
```

SN ERRO indica um erro de "sintaxe". Este é o modo do Computador dizer: "O comando PPRINT não existe no meu vocabulário... Eu não tenho idéia do que você quer que eu faça". Se aparecer uma mensagem de erro SN é porque provavelmente uma palavra não foi escrita corretamente.
O Computador também dará sua mensagem de erro quando não entender o que você quer, ou então quando você pedir que ele faça alguma coisa ilógica ou impossível.

Para conferir, experimente isto:

```
PRINT 8/0 ENTER
```

O Computador imprime:

```
?/0 ERRO
```

Que significa: "Não me peça para dividir por 0 — é impossível!!!" Se você receber uma mensagem de erro e não entender, consulte o Apêndice C. Nós listamos todas as mensagens de erro e o que provavelmente está errado.

ELE É UM SHOW

Até agora tudo que você viu seu Computador fazer foi imprimir em tela verde. Mas seu Computador também pode mostrar outras cores. Digite:

```
CLS (3) ENTER
```

Sua tela agora tem uma tonalidade azul com uma faixa verde no alto. Ao seu comando o Computador limpou a tela e imprimiu a cor nº 3 — azul.

Se você não obtiver a cor certa, verifique o teste de cor no Capítulo 1 — Seção I.

Por que a faixa verde? O Computador não pode digitar com um fundo azul. Tente imprimir alguns caracteres. Note que o Computador fornece estes caracteres em fundo verde.

Outras cores diferentes do verde serão para ilustrações gráficas. Mais adiante falaremos sobre esta capacidade de "cores". Pressione ENTER assim que você obtiver um OK em sua tela. Digite:

```
CLS (7) ENTER
```

Agora você deverá ter a cor "roxo" em sua tela com uma faixa verde no alto. Experimente mais algumas cores. Use qualquer número de 0 a 8. Seu Computador tem 9 cores. Cada cor tem um código numérico.

Se você obtiver uma mensagem dizendo PROLÓGICA ou ?FC ERRO é porque você está usando outro número que não de 0 a 8.

Digite CLS sem um código de número.

```
CLS ENTER
```

Se não usar um código de número, o Computador assume que você quer a limpeza da tela verde.

PARA AQUELES QUE GOSTAM DE MÚSICA

Digite:

```
SOUND 1,100 ENTER
```

Se você não escutar nada, aumente o volume e tente novamente. O que você está ouvindo são 6 segundos do tom mais baixo que o Computador pode reproduzir. Como obter o tom mais alto?

Digite:

```
SOUND 255,100 ENTER
```

Tente outros números.

Você quer saber o que esses dois números fazem? O primeiro dá o tom e o segundo indica a duração desse tom.
No primeiro número você pode usar qualquer um de 1 a 255. Tente!

```
SOUND 128,1 ENTER
```

E o Computador reproduzirá o tom durante aproximadamente 6/100 de um segundo. Teste 10:

```
SOUND 128,10 ENTER
```

O Computador emite o som durante 6/10 de um segundo. Experimente mais variações de ambos os números, mas sempre entre 1 e 255.

MAIS UM DETALHE: PODE-SE MUDAR AS LETRAS

Pressione SHIFT e 0 ao mesmo tempo. Depois, solte-as e escreva alguma coisa. As letras que você digitar serão agora verdes com fundo preto. Se não forem, tente novamente, pressionando levemente SHIFT antes de pressionar 0. Mantenha ambas pressionadas antes de liberá-las. Aí, tente outra vez escrever algo. Agora com as cores "trocadas", pressione ENTER e então digite esta linha de comando simples.

PRINT "OLA' " ENTER

O Computador fornece um ?SN ERRO. Ele não entendeu o comando. Pressione SHIFT e 0 novamente e digite algumas letras. Elas deverão voltar ao normal: preta com fundo verde. Pressione ENTER e digite novamente a mesma linha.
Agora vai funcionar.
O Computador não pode entender qualquer comando que você digitar com as cores "trocadas". Se você já estiver digitando assim, pressione SHIFT e 0 para obter as normais de volta.

SEU COMPUTADOR JAMAIS ESQUECE UMA LETRA

Uma das coisas que faz seu Computador poderoso é sua habilidade em se lembrar de tudo que você pede a ele. Para fazer o Computador lembrar do número 18 por exemplo, diga a ele que:

X = 18 ENTER

Agora digite qualquer coisa que você quiser para desviar a atenção dele, depois pressione ENTER. Para ver se o Computador lembra o que X significa, digite:

PRINT X ENTER

Ele se confundiu? Ou se esqueceu?

Faça então o seguinte:

X = 7.8 ENTER

Se você já conhece BASIC, atenção. O CP 400 não requer o comando LET.

MEMÓRIA DE SEU COMPUTADOR

X ⟶ ~~18~~ 7.8

Agora se você introduzir PRINT X, ele imprimirá o número 7.8. Veja no quadro ao lado o que aconteceu na memória de seu Computador.

Você não precisa usar a letra X. Você pode usar qualquer letra de A a Z. (Pode até usar duas letras.) Experimente digitar:

```
Y = 100 ENTER
B  2.5 ENTER
XY = 105 ENTER
```

Agora digite:

```
PRINT X, Y, B, XY
```

Para indicar uma string de letras ou números, não esqueça o caractere cifrão após a letra.

Por exemplo:

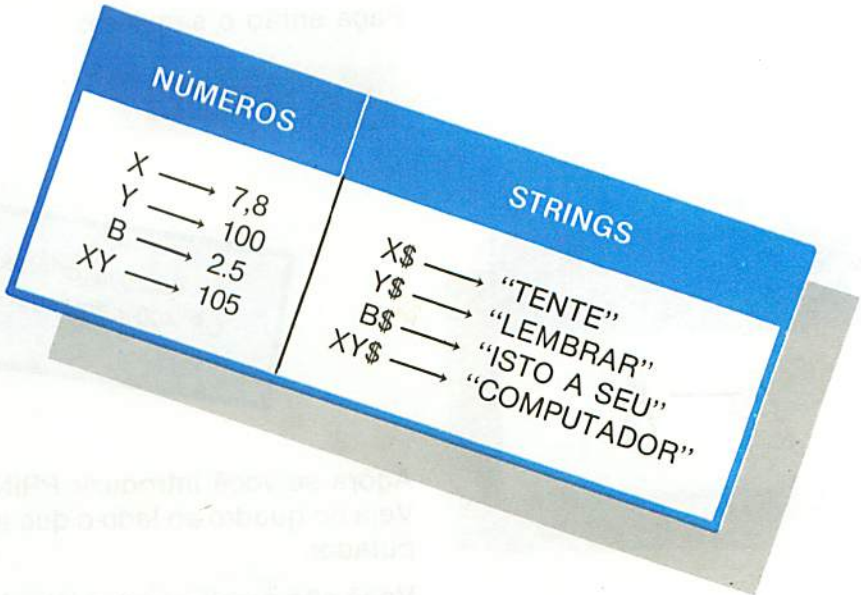
```
X$ = "TENTE"
Y$ = "LEMBRAR"
B$ = "ISTO A SEU"
XY$ = "COMPUTADOR"
```

Para o Computador o cifrão identifica uma string.

Veja se seu Computador está confuso. Digite:

```
PRINT X$, Y$, B$, XY$ ENTER
```

Existe um nome para essas letras que correspondem a valores ou strings. Elas são chamadas "variáveis".



Chame estas variáveis para ver se o Computador ainda se lembra da informação. Digite:

```
PRINT XY ENTER
```

para ver se XY ainda corresponde a 105.

Faça seu Computador lembrar uma letra que nós ainda não usamos. O que acontece?

Você pode imaginar estas variáveis como pequenas caixas onde você armazena sua informação. Um conjunto de caixas é para Strings; o outro para Números. Você usa essas variáveis para rotular cada caixa.

O COMPUTADOR É MUITO EXIGENTE COM SUAS NORMAS

Como nós dissemos antes, o Computador tem suas normas e vai ser exigente com você para que você as siga.

Você acha que o Computador aceitará estas linhas:

```
A = "12" ENTER
_B = "GUARDE ESTA STRING" ENTER
```

TM quer dizer Tipo Desigual. Significa que você não seguiu as normas.

Com estas linhas, o Computador responde com ?TM ERRO. Ele está dizendo que você deve seguir as normas.

NORMAS SOBRE STRINGS

1 — Qualquer dado entre aspas é uma String.
2 — Strings são somente guardadas em variáveis com o caractere **$**.

Veja no quadro as normas que você ignorou.

Para obedecer as normas do Computador, nós temos que colocar o sinal cifrão depois de A e B. Digite:

```
A$ = "12" ENTER
_B$ = "GUARDE ESTA STRING" ENTER
```

E o Computador as aceita.

Você acha que o Computador aceita a linha a seguir?

```
A$ = 12 ENTER
```

Estas são as normas que este comando ignorou:

Digite deste modo, assim o Computador aceita.

```
A = 12 ENTER
B = 22 ENTER
```

NORMAS SOBRE NÚMEROS

1 — Números que não estiverem entre aspas são Dados Numéricos.
2 — Dados Numéricos são guardados apenas em variáveis sem o caractere $.

Assim tudo foi guardado na memória de seu Computador.

MEMÓRIA DE SEU COMPUTADOR

Números	Strings
A ⟶ 12	A$ ⟶ "12"
B ⟶ 22	B$ ⟶ "GUARDE ESTA STRING"

Agora você pode fazer algo mais interessante com estas letras.
Digite:

```
PRINT A*2 ENTER
```

O Computador ainda lembra que A = 12

O Computador imprime o produto de A vezes 2.
Tente esta linha:

```
PRINT B/A
```

O Computador imprime o resultado de B dividido por A.
Digite:

```
PRINT A$*2 ENTER
```

Isto também faz o Computador imprimir ?TM ERRO.
Não se pode multiplicar strings.

NORMAS SOBRE VARIÁVEIS

Você pode usar dois caracteres quaisquer de A - Z para uma variável. O primeiro caractere deve ser uma letra de A - Z, o segundo, entretanto, pode ser um número ou uma letra. Se você quiser designar uma string, coloque um caractere $ depois dela. De outro modo ela será considerada um Número.

Agora que você sabe como o Computador pensa, será fácil escrever alguns programas.

CAPÍTULO 3

às suas ordens

Digite:

```
NEW ENTER
```

para limpar qualquer coisa que estiver na memória do Computador. Agora digite esta linha (Não esqueça o número 1Ø — é muito importante.):

```
10 PRINT "OLA', EU SOU SEU COMPUTADOR" ENTER
```

Você pressionou ENTER? E não aconteceu nada? Pelo menos, nada que você pudesse ver.
O que você fez foi digitar seu primeiro programa.
Digite:

```
RUN ENTER
```

O Computador prontamente executa o seu programa. Digite novamente RUN ENTER, e ele repetirá o programa. Seu programa será executado quantas vezes você quiser.
Já que saiu tudo bem, acrescente esta ou outra linha ao programa. Digite:

```
2Ø PRINT "QUAL E' O SEU NOME?"
```

Agora digite:

```
LIST ENTER
```

Seu Computador agora vai "Listar" todo o seu programa. Em sua tela deve aparecer exatamente isto:

```
10 PRINT "OLA', EU SOU SEU COMPUTADOR"
20 PRINT "QUAL E' O SEU NOME?"
```

O que acontecerá se você digitar RUN novamente? Tente.

```
RUN ENTER
```

E o Computador vai imprimir:

```
OLA', EU SOU SEU COMPUTADOR
QUAL E' O SEU NOME?
```

Responda a questão do Computador e pressione ENTER ... O

quê? Apareceu um SN ERRO? O Computador não entendeu o que você pretendia quando você digitou seu nome. De fato, o Computador não pode entender qualquer coisa a menos que você fale com ele do jeito que ele entende.
Deste modo use uma palavra que o Computador entenda — INPUT. Digite esta linha:

```
30 INPUT A$ ENTER
```

Se você perceber um erro depois de pressionar ENTER, apenas digite toda linha novamente. O Computador só considera a última versão introduzida.

Isto faz o Computador parar e esperar que você digite alguma coisa, que será guardada como A$. Adicione mais uma linha ao programa:

```
40 PRINT "OLA'," A$ ENTER
```

Agora liste o programa. Digite:

```
LIST ENTER
```

Seu programa deve estar assim:

```
10 PRINT "OLA', EU SOU SEU COMPUTADOR"
20 PRINT "QUAL E' O SEU NOME?"
30 INPUT A$
40 PRINT "OLA'," A$
```

Você pode adivinhar o que acontecerá quando você pressionar RUN? Tente.

```
RUN ENTER
```

Tudo certo? Isto provavelmente é o que acontecerá quando você executar o programa:

```
OLA', EU SOU SEU COMPUTADOR
QUAL E' O SEU NOME?
? MARIA
OLA', MARIA
```

Rode o programa novamente usando outros nomes:

```
OLA', EU SOU SEU COMPUTADOR
QUAL E' O SEU NOME?
? 011-240-8305
OLA', 011-240-8305
OK
OLA', EU SOU SEU COMPUTADOR
QUAL E' O SEU NOME?
? AGORA ENTENDI
OLA', AGORA ENTENDI
OK
```

Veja no quadro o que a linha 30 faz à memória de seu Computador cada vez que executa o programa.

Há um modo mais fácil de reiniciar sem ter que digitar o comando RUN.
Digite esta linha:

```
50 GOTO 10
```

O programa deve ficar assim:

```
10 PRINT "OLA', EU SOU SEU COMPUTADOR"
20 PRINT "QUAL E' SEU NOME?"
30 INPUT A$
40 PRINT "OLA'," A$
50 GOTO 10
```

Agora o seu programa vai rodar para sempre, pois toda vez que for executada a linha 50, o Computador volta a executar a linha 10. É isso o que a gente costuma chamar de ciclo (alguns preferem o termo *loop*, que quer dizer a mesma coisa). A única maneira de parar com esse ciclo interminável é usando a tecla BREAK.

UM REALCE NO SEU NOME

Para tirar uma das linhas de um programa, simplesmente digite o número da linha seguido de ENTER. Por exemplo:
50 ENTER
Isso vai apagar a linha 50 do programa.

Mude a linha 50 de forma que possamos dar ao seu nome o destaque que ele merece. Como mudar uma linha de programa? Ora, simplesmente digite-a novamente, com a alteração necessária. Digite:

```
50 GOTO 40
```

É assim que o programa deve ficar agora:

```
10 PRINT "OLA', EU SOU SEU COMPUTADOR"
20 PRINT "QUAL E' O SEU NOME?"
30 INPUT A$
40 PRINT "OLA'," A$
50 GOTO 40
```

Ponha-o para rodar (RUN), e veja o que acontece. Use BREAK para fazê-lo parar.
Podemos mudar bastante o programa apenas colocando uma vírgula ou um ponto e vírgula na linha 40. Tente com a vírgula primeiro:
Vamos deixar o OLA' de lado, por enquanto.

```
40 PRINT A$,
```

Rode o programa... A vírgula faz com que tudo seja impresso em duas colunas.
Pressione BREAK e tente agora com o ponto e vírgula. Digite:

```
40 PRINT A$;
```

e RUN ENTER ... Na certa você não vai conseguir ver o que está acontecendo até que use BREAK ... Viu como o ponto e vírgula agrupa as palavras?

NORMAS SOBRE PONTUAÇÃO DE IMPRESSÃO

Eis os efeitos na tela da pontuação no final da linha com a instrução PRINT:

1. *uma vírgula faz o Computador imprimir em colunas;*
2. *um ponto e vírgula faz o Computador imprimir as strings uma ao lado da outra;*
3. *sem pontuação o Computador imprime em linhas.*

APERFEIÇOANDO O PROGRAMA

Os programadores profissionais talvez achem que pressionar BREAK não seja a melhor maneira de interromper a execução do programa. Por que então não fazer com que o Computador pergunte se queremos que ele pare? Troque as linhas 40 e 50 do programa acima para:

```
40 PRINT "OLA'," A$
50 PRINT "VOCÊ QUER CONTINUAR?"
```

e adicione:

```
60 INPUT R$
70 IF R$ = "SIM" THEN 20
```

Rode o programa... Digite SIM e o programa perguntará novamente seu nome.
Digite qualquer coisa diferente e ele pára.
O programa está assim:

```
10 PRINT "OLA', EU SOU SEU COMPUTADOR"
20 PRINT "QUAL E' O SEU NOME"
30 INPUT A$
40 PRINT "OLA'," A$
50 PRINT "VOCÊ QUER CONTINUAR?"
60 INPUT R$
70 IF R$ = "SIM" THEN 20
```

Não se preocupe com IF/THEN. Falaremos sobre ele mais adiante.

Veja o que estas linhas fazem.
A linha 50 simplesmente imprime a pergunta.
A linha 60 fala ao Computador para parar e esperar por sua resposta — R$.

A linha 70 manda o Computador para a linha 20 SE (em inglês, IF) sua resposta R$ for SIM. Se não, o programa simplesmente acaba, desde que não haja mais linhas.

FAZENDO O COMPUTADOR CONTAR

Para conseguir alguns efeitos, vamos antes ensiná-lo a contar.
Digite isto:

```
10 FOR X = 5 TO 15
20 PRINT "X = " X
30 NEXT X
40 PRINT "EU TERMINEI DE CONTAR"
```

e rode o programa.
Rode o programa várias vezes, e a cada vez troque a linha 10 por uma destas linhas:

```
10 FOR X=1 TO 50
10 FOR X=5 TO 5
10 FOR X=15 TO 20
```

Você está vendo o que acontece com FOR e NEXT? Eles fazem o Computador contar. Estude o último programa sugerido:

```
10 FOR X=15 TO 20
20 PRINT "X=" X
30 NEXT X
40 PRINT "EU TERMINEI DE CONTAR"
```

A linha 10 diz ao Computador que o primeiro número deverá ser 15 e o último 20. Ele usa X para guardar estes valores, um por vez.
A linha 30 diz ao Computador para voltar à linha 10 e pegar o próximo valor de X — até chegar no último (20).
Atenção para a linha 20. Já que ela está entre as linhas FOR e NEXT o Computador vai executá-la a cada valor de X.

```
X=15
X=16
X=17
X=18
X=19
X=20
```

Coloque esta outra linha entre FOR e NEXT.

```
15 PRINT "...CONTANDO..."
```

e execute-o. Seu Computador executa qualquer linha que você inserir entre FOR e NEXT.
Escreva um programa para o Computador imprimir o seu nome 12 vezes.

FAÇA VOCÊ MESMO

.....

.....

Sugestão: O programa deve contar até 12.

Escreva um programa para imprimir a tabuada de multiplicação do 7 (7*1 até 7*10).

FAÇA VOCÊ MESMO

.....

.....

Sugestão: Com uma vírgula na linha PRINT, você obtém todos os problemas e resultados em sua tela ao mesmo tempo.

Terminou? Aqui estão os nossos programas:

Programa 3-1	Programa 3-2
10 FOR X = 1 TO 12 20 PRINT "APARECIDA 30 NEXT X	10 FOR X = 1 TO 10 20 PRINT "7*"X" = "7*X 30 NEXT X

CONTAR AOS SALTOS

Agora nós o faremos contar de um modo diferente. Limpe seu programa digitando NEW e então digite nosso programa, usando uma nova linha 10.

```
10 FOR X = 10 TO 19 STEP 2
20 PRINT "X = " X
30 NEXT X
40 PRINT "EU ACABEI DE CONTAR"
```

Rode o programa... Você viu o que o STEP 2 fez? Ele fez o Computador contar de 2 em 2. A linha 10 fala ao Computador que:

- O primeiro X é 10
- O último X é 19

... e STEP 2 ...

- Todos os X entre 10 e 19 de 2 em 2, isto é, 10, 12, 14, 16 e 18.

(STEP 2 diz ao Computador para somar 2 para obter o próximo valor de X.)

Ele ignora o último X (19) porque 18 + 2 = 20.
Tente mais algumas linhas FOR... STEP para você ver claramente como funciona; utilizando as seguintes linhas como linha 10.

```
10 FOR X = 5 TO 50 STEP 5
10 FOR X = 10 TO 1 STEP -1
```

CONTANDO OS SONS

Agora que você ensinou o Computador a contar, você pode repetir a contagem com "acompanhamento musical".
Apague seu programa anterior e digite isto:

```
10 FOR X=1 TO 255
20 PRINT "TOM" X
30 SOUND X, 1
40 NEXT X
```

Este programa faz o Computador contar de 1 até 255 (de 1 em 1). Para cada valor, ele executa a ordem das linhas 20 e 30.

- Linha 20 — imprime o valor atual de X.
- Linha 30 — dá o tom correspondente a X.

Por exemplo:

A primeira vez que o Computador passa pelo FOR, na linha 10, faz X igual a 1.

- Vai então para a linha 20 e imprime 1, como valor de X.
- A linha 30 dá o tom # 1.
- Volta à linha 10 e faz X igual a 2.
- E assim por diante.

O que acontece se você fizer esta troca na linha 10:

```
10 FOR X=255 TO 1 STEP -1
```

Tente isto: Para suspender a execução do programa, pressione SHIFT *e* @ *ao mesmo tempo. Depois pressione qualquer tecla para continuar.*

FAÇA VOCÊ MESMO

.....

.....

.....

.....

Você tentou? Agora usando STEP, troque a linha 10 para que o Computador execute os tons nas seguintes seqüências:

(1) De baixo para cima, a intervalos de 8 notas.
(2) De cima para baixo, a intervalos de 8 notas.
(3) Do meio (128) para cima, a intervalos de 4 notas.

Veja a resposta.

```
10 FOR X=1 TO 255 STEP 8
10 FOR X=255 TO 1 STEP -8
10 FOR X=128 TO 255 STEP 4
```

FAÇA VOCÊ MESMO

.....

.....

.....

.....

Agora veja se você pode escrever um programa que faça o Computador executar:

(1) De cima para baixo, com intervalos de 10 notas, e depois
(2) De baixo para cima, com intervalos de 5 notas.

A resposta está no Apêndice A.

APRENDENDO O RITMO DA MÚSICA

Agora você já pode usar o Computador para duas coisas importantes: dizer as horas e cantar (bom, *cantar* do jeito dele, é claro!). Vamos começar com um exercício simples:

```
10 FOR Z=1 TO 460*2
20 NEXT Z
30 PRINT "EU CONTEI ATE' 920"
```

Rode o programa. O tempo de espera será de 2 segundos. Seu Computador gasta 2 segundos para contar até 920.
Linhas 10 e 20 fornecem uma "pausa" com boa precisão em seu programa. Fazendo o Computador contar até 920, ele mantém o Computador ocupado por 2 segundos.
Como você pode ver, isto equivale a um "cronômetro". Apague o programa (NEW) e introduza este outro:

```
10 PRINT "QUANTOS SEGUNDOS"
20 INPUT S
30 FOR Z=1 TO 460*S
40 NEXT Z
50 PRINT "PASSARAM-SE "S" SEGUNDOS"
```

Este programa pede a pausa que você quer medir no seu "cronômetro".
Vamos adicionar algumas linhas ao nosso programa para que no fim do tempo soe um sinal.
Aqui está o programa:

```
10 PRINT "QUANTOS SEGUNDOS"
20 INPUT S
30 FOR Z=1 TO 460*S
40 NEXT Z
50 PRINT  "SEGUNDOS"
60 FOR T=90 TO 170 STEP 20
70 SOUND T,1
80 NEXT T
90 FOR T=150 TO 140 STEP -1
100 SOUND T,1
110 NEXT T
120 GOTO 50
```

Notar a instrução GOTO que colocamos no fim do programa. Ela mantém a mensagem e o alarme até que o programador use BREAK *ou* SHIFT @ *para pará-lo.*

O RELÓGIO FEITO POR PROGRAMA

Antes de falarmos sobre relógios vamos manter o Computador marcando o tempo. Logo você vai entender melhor.
Digite este novo programa:

```
10 FOR X=1 TO 3
20 FOR Y=1 TO 2
30 PRINT "X=" X, "Y=" Y
40 NEXT Y
50 NEXT X
```

RUN... Aparecerá em sua tela

```
X = 1    Y = 1
X = 1    Y = 2
X = 2    Y = 1
X = 2    Y = 2
X = 3    Y = 1
X = 3    Y = 2
```

Chama-se um contador dentro de outro contador ou um ciclo dentro de outro ciclo (o que você preferir). Programadores chamam isto de "ciclo alojado".

Eis o que o programa faz:
1º — Conta X de 1 a 3. Para cada valor de X faz o seguinte: conta Y de 1 a 2, e para cada valor de Y, imprime o valor de X, seguido do de Y.

Quando se coloca um ciclo dentro de outro, deve-se fechar o ciclo interno antes de fechar o ciclo externo.

CERTO

```
10 FOR X=1 TO 3
20 FOR Y=1 TO 2
30 NEXT Y
40 NEXT X
```

ERRADO

```
10 FOR X=1 TO 3
20 FOR Y=1 TO 2
30 NEXT X
40 NEXT Y
```

O PROGRAMA DAS HORAS

Usando o que você aprendeu o Computador pode ir um pouco além.
Digite:

```
10 FOR S=0 TO 59
20 PRINT S
30 SOUND 200,2
40 FOR T=1 TO 390
50 NEXT T
60 NEXT S
70 PRINT "PASSOU UM MINUTO"
```

Rode o programa... O que ele faz:
1 — Conta os segundos de 0 a 59.

E para cada segundo ele:
a) Imprime o segundo
b) Gera um tom
c) Pára um tempo suficiente para passar 1 segundo

2 — Quando termina a contagem, indica com a mensagem que já passou um minuto.

Agora coloque esta linha para limpar a tela:

```
15 CLS
```

Rode o novo programa. Ele:
I — Conta os segundos de 0 a 59 (linhas 10 e 60).

E para cada segundo:
a) Limpa a tela (linha 15)
b) Imprime o segundo no canto esquerdo da tela (linha 20)
c) Gera um tom (linha 30)
d) Pára um tempo suficiente para passar um segundo (linhas 40 e 50)

II — Quando termina a contagem, indica com a mensagem que já passou um minuto (linha 70).

Com esse conhecimento básico vamos construir um relógio sofisticado:

```
10 FOR H=0 TO 23
20 FOR M=0 TO 59
30 FOR S=0 TO 59
40 CLS
50 PRINT H";"M";"S
60 SOUND 200,2
70 FOR T=1 TO 375
80 NEXT T
90 NEXT S
93 FOR T=210 TO 216 STEP 3
94 SOUND T,1
95 NEXT T
100 NEXT M
102 FOR T=216 TO 210 STEP-3
103 SOUND T,1
104 NEXT T
110 NEXT H
```

Vamos ver o que o Computador faz neste programa:
I — Conta as horas de 0 a 23 (linha 10).

Para cada hora:
A. Conta os minutos de 0 a 59 (linha 20)

Para cada minuto:
(1) Conta os segundos de 0 a 59 (linhas 30 e 90)
a) Limpa a tela
b) Imprime a hora, o minuto e o segundo (linha 50)
c) Gera um tom (linha 60)
d) Pára um tempo suficiente para passar um segundo (linhas 70 e 80)
(2) Quando termina de contar os 60 segundos, gera um som de 3 notas crescente e retorna à linha 20 para o próximo minuto (linha 100).
B. Quando termina de contar todos os 59 minutos, gera um som de 3 notas decrescente e retorna à linha 10 para a próxima hora (linha 110).

Se pusermos a linha 120 GOTO 10 o relógio funcionará sem parar.

II — Quando termina de contar todas as horas (0-23) o programa termina.

OUÇA A MÚSICA DO COMPUTADOR

Agora vamos ensinar o seu Computador a cantar. Olhe no Apêndice D. Nós temos uma tabela de tons musicais, que mostra o número do tom do Computador para cada nota no teclado musical. Por exemplo, o tom nº 89 do Computador corresponde ao "Dó".

Digite:

```
10 SOUND 89,8
20 SOUND 108,8
30 SOUND 125,8
```

Rode o programa. Estas são as primeiras três notas da nossa escala musical. Vamos escutar o resto?

```
40 SOUND 133,8
50 SOUND 147,8
60 SOUND 159,8
70 SOUND 170,8
80 SOUND 176,8
```

Rode o programa novamente. Pronto você tem aí todas as notas com um mesmo tempo de duração.

CAPÍTULO 4

penso. logo, existo.

Aqui está uma situação típica onde o Computador deve tomar decisões:
Se (IF) você digitar AZUL ... *então* (THEN) a tela deve ficar azul, ou *Se* (IF) você digitar AMARELO ... *então* (THEN) a tela deve ficar amarela.
Digite este programa (não se esqueça de usar NEW para limpar a memória):

```
10 PRINT "VOCE QUER A TELA AZUL OU AMARELA?"
20 INPUT C$
30 IF C$ = "AZUL" THEN 100
40 IF C$ = "AMARELO" THEN 200
100 CLS(3)
110 END
200 CLS(2)
```

C$ = AZUL

C$ = AMARELO

Rode o programa várias vezes, tentando todas as cores. Se você digitar AZUL, a linha 30 vai mandar o Computador para a linha 100 onde a instrução CLS(3) faz a tela ficar toda azul. Feito isso, temos que fazer o Computador parar o programa, para que ele não execute a próxima linha, que faz a tela ficar amarela (200 CLS(2)). A linha 110 se encarrega disso. Quando o Computador executa a linha 110, ele não executa mais nada.
Por outro lado... se você digita AMARELO, a linha 40 manda o Computador para a linha 200. Essa linha deixa a tela amarela, e não é preciso nenhuma instrução END porque não tem mais programa depois da linha 200.
Espera aí!... E se eu não digitar nem AMARELO nem AZUL? Como é que o Computador vai agir nesse caso?
Tente rodar o programa e digitar outra cor que não o azul ou o amarelo. O que aconteceu? A tela ficou azul, certo? Você sabe por quê?

Se a condição não for verdadeira (se o que está depois do IF não acontecer), o restante da linha é ignorado (o que está depois do THEN). Nesse caso o Computador passa para a próxima linha.

Você pode melhorar este programa e fazer com que o Computador peça uma nova instrução no caso de você não digitar AZUL ou AMARELO. Com estas duas linhas você resolve o problema.

```
50 PRINT "VOCE DEVE DIGITAR AZUL OU AMARELO"
60 GOTO 20
```

Agora vamos ver se você consegue fazer mais uma mudança no programa.
No lugar do programa acabar quando deixa a tela azul ou amarela, mude-o para que ele repita a pergunta sobre a cor que você quer.

Você precisa mudar a linha 110.

Já alterou o programa?

```
10 PRINT "VOCE QUER A TELA AZUL OU AMARELA?"
20 INPUT C$
30 IF C$ = "AZUL" THEN 100
40 IF C$ = "AMARELO" THEN 200
50 PRINT "VOCE DEVE DIGITAR AZUL OU AMARELO"
60 GOTO 20
100 CLS(3)
110 GOTO 10
200 CLS(2)
210 GOTO 10
```

NORMAS SOBRE IF/THEN e GOTO

IF/THEN *é condicional. Você só deve fazer os desvios se a condição (C$ = "AZUL" ou C$ = "AMARELO") realmente acontecer. Chamamos este caso de condição verdadeira.*
GOTO *é incondicional. Você deve fazer o desvio sempre que chegar a uma linha GOTO.*

Tente acompanhar o desenrolar do programa pela listagem. Não rode o programa, apenas acompanhe mentalmente o que acontece após cada instrução. Nas instruções IF/THEN só ocorrerá o desvio se a condição for satisfeita. Já nas instruções GOTO, sempre ocorrerá o desvio indicado.

OS NÚMEROS AO ACASO

Pense num número de 1 a 10! ...7... Pense em outro! ...5... Você pode ficar indefinidamente nessa brincadeira sem que haja qualquer relação ou fórmula matemática entre os números que você escolhe mentalmente. Os números são escolhidos ao acaso, sem nenhuma regra que determine como eles devem aparecer.
O Computador também pode entrar nessa brincadeira! Para isso existe uma função em BASIC, chamada RND, que permite ao Computador escolher um número qualquer dentro de um limite que você determina. Por exemplo:

```
10 PRINT RND (10)
```

Quando você roda esse programa de uma só linha, o Computador imprime um número qualquer entre 1 e 10. Se você manda rodar

novamente, outro número será mostrado, e assim por diante. Observando a seqüência em que os números são escolhidos, você notará que não há uma regra para os números aparecerem. Eles são aparentemente casuais, ou aleatórios, como preferimos chamá-los. Modifique a linha 10 do programa e acrescente a 20, como segue:

```
10 PRINT RND (10);
20 GOTO 10
```

Rode o programa e pressiona BREAK quando quiser pará-lo. Note que o Computador vai encher a tela com números entre 1 e 10. Se você quiser que o Computador gere números entre 1 e 100, por exemplo, basta mudar a linha 10 para:

```
10 PRINT RND (100);
```

Vamos ver agora como o seu Computador pode aprender alguns jogos. Você pode usar a sua imaginação para incrementá-los, ou inventar os seus próprios.

ADIVINHE O NÚMERO

Neste jogo você tem que adivinhar um número que seu Computador escolheu aleatoriamente entre 1 e 10. Você terá apenas 5 tentativas. Para desistir basta digitar qualquer número maior que 10. Vamos lá, digite-o:

```
5 CLS
10 X=RND(10)
20 FOR Y=1 TO 5
30 INPUT "ADIVINHE QUAL O NUMERO <1-10>";N
40 IF N>10 THEN 100
50 IF N=X THEN 200
60 PRINT "AINDA NAO ACERTOU"
65 SOUND 40*Y,2
70 NEXT Y
80 PRINT "E' UMA PENA, VOCE FRACASSOU"
90 GOTO 110
100 PRINT "DESISTIU HEIN?"
110 END
200 PRINT "ATE' QUE ENFIM VOCE ACERTOU!!!"
210 PRINT "TENTOU "Y" VEZES"
```

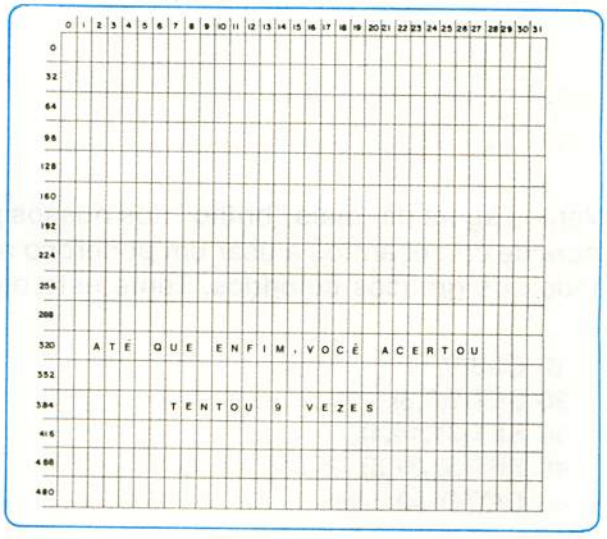
Na linha 10 o Computador vai escolher um número. Na linha 20 você tem um *loop* que vai contar suas 5 chances. Quando o número digitado N na linha 30 for igual ao número dado pelo Com-

putador na linha 50, ele vai para a linha 200 e imprime a mensagem de que você acertou.
A cada número que você digita e não acerta é gerado um tom que cresce a cada rodada.
Vamos melhorar a nossa rotina de acerto.
Adicione:

```
 83 SOUND 50,10
 84 SOUND 30,10
 85 SOUND 10,10
105 SOUND 5,5
106 SOUND 2,5
200 CLS
210 PRINT @ 321, "ATE' QUE ENFIM, VOCE ACERTOU!!!"
220 FOR X=200 TO 255 STEP 5
225 SOUND X,5
228 NEXT X
230 PRINT @ 392, "TENTOU "Y" VEZES"
```

Atenção para as linhas 210 e 230. Ambas usam PRINT @. Vamos ver o que elas fazem. A seguir colocamos uma tela quadriculada enumerando cada uma das 512 posições em que se pode imprimir na tela. Nós escrevemos as duas mensagens que queremos imprimir sobre o quadriculado exatamente onde queremos que elas apareçam. ATE' QUE ENFIM, VOCE ACERTOU começa na posição 321 (320 + 1). TENTOU "Y" VEZES está na posição 392 (384 + 8). Usando essas posições na linha PRINT @ nós dizemos ao Computador onde **exatamente** queremos as mensagens.

Colocamos este quadriculado no Apêndice E deste manual. Use-o para distribuir melhor as mensagens que os seus programas produzirão.



OS DADOS DA SORTE

Para o nosso próximo jogo você terá, antes de tudo, que ensinar o Computador a jogar dados. Para jogar um par de dados, o Computador deve gerar dois números de 1 a 6. Introduza, então, o programa:

```
10 CLS
20 X=RND(6)
30 Y=RND(6)
40 R=X+Y
50 PRINT @ 200, X
60 PRINT @ 214, Y
70 PRINT @ 391, "VOCE CONSEGUIU UM" R
80 PRINT @ 454, "VOCE QUER JOGAR OUTRA VEZ?"
90 INPUT R$
100 IF R$="SIM" THEN 10
```

Rode o programa e veja o que aconteceu.
A linha 10 limpa a tela para começar o jogo.
Na linha 20 o Computador escolhe o número que vai dar no primeiro dado.
Na linha 30 o Computador escolhe o número do segundo dado.
A linha 40 simplesmente soma os dois valores.
Da linha 50 até a 70 a jogada é mostrada na tela.
Na linha 90 você deve dizer se continua jogando ou não. Se você introduzir um SIM, o Computador joga os dados novamente, caso contrário, o programa termina.

PONDO MAIS COR EM SUA TELA

Vamos agora dar mais "brilho" aos nossos programas. Já está na hora de começarmos a usar um poderoso recurso de seu Computador: os **gráficos coloridos**. Tente este programa:

```
10 CLS(0)
20 SET(0,0,3)
30 SET(31,14,4)
40 SET(63,31,5)
50 GOTO 50
```

Não se preocupe com a linha 50. Isso será explicado mais à frente. Rode o programa e veja como vai aparecer um ponto azul (C = 3) no canto superior esquerdo da tela (0,0), linha 20; um ponto vermelho (C = 4) no meio da tela (31,14), na linha 30; e um ponto cinza (C = 5), no canto inferior direito (63,31), na linha 40.

A instrução SET diz ao Computador para colocar um ponto em uma certa posição que você determina. Essa posição é identificada por dois números:

- O primeiro identifica a posição horizontal do ponto. Pode ter qualquer valor entre 0 e 63.
- O segundo identifica a posição vertical do ponto. Pode ter qualquer valor entre 0 e 31.

No Apêndice E é mostrada uma tela quadriculada intitulada "Posições Gráficas na Tela". Essa tela está dividida em 64 (de 0 a 63) posições horizontais e 32 (de 0 a 31) verticais.
O terceiro número, como já vimos, identifica a cor. Todos os códigos e as cores a que se referem estão relacionados no Apêndice D.

O Computador usa posições diferentes para SET e PRINT @. Por isso, existem duas telas quadriculadas no apêndice.

Chegou a hora de ver para que serve a linha 50 GOTO 50. Tire essa linha do programa. Basta digitar 50 e pressionar ENTER. Depois disso rode o programa.
O que aconteceu? Os pontos não apareceram. Na verdade, eles foram colocados, mas quando o programa termina, o Computador coloca a mensagem OK na primeira linha, em cima dos pontos que ele acabou de colocar.
Para evitar isso (o fim do programa e a conseqüente mensagem OK) é que usamos uma linha 50 GOTO 50. Ela serve para "segurar" o andamento do programa. Quando encontra essa linha, o Computador entra em *loop* até que alguém pressione BREAK.

FAÇA UMA LINHA COM PONTOS

Para colocar mais de um ponto na tela, você precisa fazer uma pequena modificação no programa. Veja como ele vai ficar:

```
10 CLS(0)
20 SET(32,14,1)
30 SET(33,14,1)
40 GOTO 40
```

Ao executar esse programa, vão aparecer dois pontos verdes juntos na tela. Na realidade, a impressão é que tem um só ponto "largo". Vamos tentar mudar a cor do ponto da direita para amarelo. Altere a linha 30 para:

```
30 SET (33,14,2)
```

Agora você está esperando que, ao executar o programa, apareçam dois pontos, um verde e o outro amarelo, no centro da tela.

Rode o programa... Epa! Os dois pontos ficaram amarelos?! Por quê?

Olhe novamente, com bastante atenção, para a tela quadriculada do Apêndice E. Note que cada bloco de quatro posições está separado dos demais por uma linha mais escura. Por exemplo, o bloco no centro da tela contém as seguintes posições:

	HORIZONTAL	VERTICAL
Posição	32	14
Posição	33	14
Posição	32	15
Posição	33	15

Os quatro pontos (ou posições) de cada bloco devem ter a mesma cor ou, então, se a cor de fundo da tela for preta o bloco terá uma cor e o preto. Isso quer dizer que um bloco só pode ter duas cores por vez: o preto de fundo; e uma das oito cores (1 a 8) restantes.

Por isso é que o seu programa mostrou dois pontos amarelos.

Quando o Computador executou a linha 20, ele realmente "acendeu" um ponto verde na tela, pois todos os outros pontos do grupo estavam "apagados" (pretos). Aí, quando ele executou a linha 30, para colocar um ponto amarelo nesse mesmo grupo, ele teve que deixar todos os pontos "acesos" com a mesma cor e, nesse caso, ele fica sempre com a última cor escolhida, o amarelo.

Agora que já sabemos por que os pontos ficaram da mesma cor, vamos dar um jeito para fazer aparecerem pontos de cores diferentes, juntos. Vamos lá:

```
30 SET(31,14,2)
```

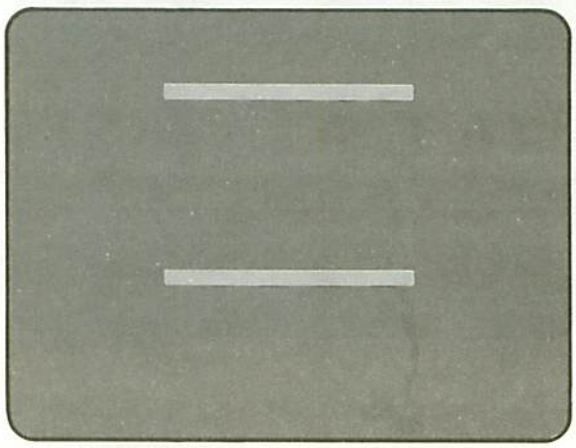
Agora sim! Como o ponto (31,14) está num grupo diferente do anterior (32,14), o Computador o deixa amarelo sem mexer na cor do outro.
Agora vamos mudar a linha 10. Digite 10 CLS(8). O que aconteceu? Pense!

O ROSTO DO COMPUTADOR

Vamos tentar desenhar alguma coisa na tela. Podemos começar com um rosto na cor cinza bem simples. Primeiro os contornos superior e inferior da cabeça:

```
 5 CLS(0)
10 FOR H=10 TO 53
20 SET(H,5,5)
30 SET(H,28,5)
40 NEXT H
50 GOTO 50
```

A tela deve ficar assim (as linhas devem ser cinzas e não brancas):



As linhas de 10 a 40 fecham um *loop* FOR/NEXT para variar H de 10 até 53. H representa a posição horizontal dos pontos traçados nas linhas 20 e 30.
Os contornos laterais serão feitos pelo seguinte trecho do programa:

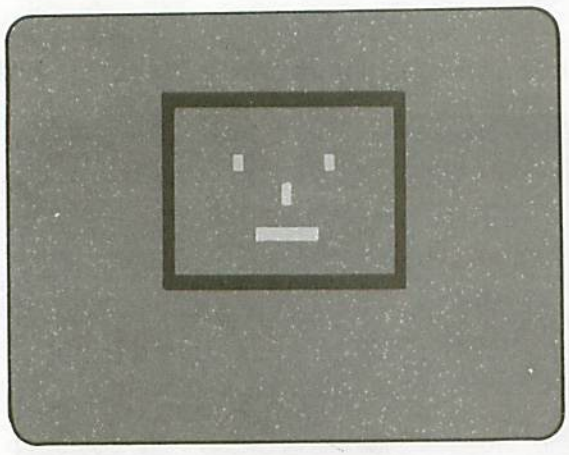
```
50 FOR V=5 TO 28
60 SET(10,V,5)
70 SET(53,V,5)
80 NEXT V
90 GOTO 90
```

Note que nós começamos a modificar desde a linha 50.

Rode o programa e veja que o contorno do rosto do seu Computador está completo. É claro que, já que ele é um "ser" estritamente matemático, nada melhor que uma cara "geométrica".
O trecho a seguir coloca o nariz vermelho, a boca alaranjada e os olhos verdes:

```
 90 SET(32,13,4)
100 SET(32,14,4)
110 SET(32,15,4)
120 FOR H = 25 TO 39
130 SET(H,20,8)
140 NEXT H
150 SET(22,10,1)
160 SET(42,10,1)
170 GOTO 170
```

Rode o programa e veja se vai aparecer alguma coisa do tipo:



FAZENDO SEUS OLHOS PISCAREM

Usando a instrução RESET nós podemos fazer o Computador piscar. Só é necessário colocar a posição do ponto, não a cor, porque a cor de fundo é sempre o preto.

Assim, introduza as seguintes linhas:

```
170 FOR T=1 TO 460
180 NEXT T
190 RESET(42,10)
200 RESET(22,10)
210 FOR X=1 TO 300
220 NEXT X
230 GOTO 150
```

Nós colocamos duas pausas nas linhas 170/180 e 210/220. A primeira dá um tempo para os olhos ficarem abertos, e a segunda, um tempo menor para eles se fecharem.
Com alguma criatividade, você pode fazer algumas variações nesse rosto, ou então partir para outras figuras, mais complexas.

O PONTO QUE SE MOVE

Usando SET e RESET, podemos fazer figuras em movimento. Introduza este programa para fazer um ponto andar pela tela:

```
 5 CLS(0)
10 FOR N=0 TO 63
20 SET(H,15,7)
30 RESET(H,15)
40 NEXT H
```

Não esqueça de apagar o programa anterior usando NEW.

Todo ponto colocado pela linha 20 é apagado pela linha 30. Vamos fazer o ponto voltar com as linhas:

```
50 FOR H=63 TO 0 STEP -1
60 SET(H,15,7)
70 RESET(H,15)
80 NEXT H
```

Para ir de um lado para o outro sem parar, adicione esta linha:

```
90 GOTO 10
```

Você vai perceber que o ponto anda muito rápido. Vamos acalmá-lo um pouco, com as linhas:

```
30 IF H>1 THEN RESET(H-2,15)
70 IF H>62 THEN RESET(H+2,15)
```

O sinal > significa o mesmo que em matemática, maior que. O sinal <, menor que. Estes sinais são usados para testar a relação entre os valores que eles separam.

PONDO TUDO SOB CONTROLE

Se você tem joysticks em seu Computador, existe uma infinidade de opções de uso para eles. Se ainda não os conectou, faça-o agora. Simplesmente encaixe-os na tomada correspondente atrás do Computador. Eles só se encaixam no conector correto, não há como errar.
Introduza agora este pequeno programa de teste:

```
10 CLS
20 PRINT @ JOYSTK(0),"*";
30 PRINT @ JOYSTK(1)+64,"*";
40 PRINT @ JOYSTK(2)+128,"*";
50 PRINT @ JOYSTK(3)+192,"*";
60 GOTO 10
```

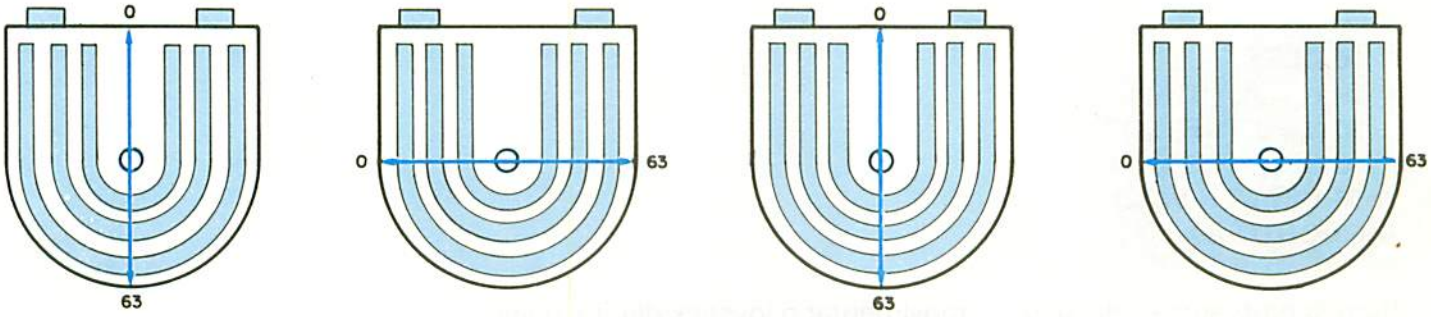
É muito importante colocar os pontos e vírgulas (;) no final das linhas 20, 30, 40 e 50.

Ao rodar o programa você vai perceber a movimentação dos quatro asteriscos em função da movimentação dos joysticks. Observe que movimentando o joystick direito de um lado para o outro (da esquerda para a direita) o primeiro asterisco também se movimenta, percorrendo as duas primeiras linhas da tela (as primeiras 64 posições do PRINT @), também da esquerda para a direita.

Pode haver um leve movimento dos segundo e quarto asteriscos.

Se você levar de volta a alavanca do joystick para a esquerda, o asterisco retorna para o começo da primeira linha.
Como você pode ver, o JOYSTK(0) verifica a posição horizontal (veja a figura abaixo) do joystick direito. Ele fornece um valor entre 0 e 63, por isso o asterisco precisa de duas linhas para se deslocar.
A posição vertical do joystick direito é verificada pelo JOYSTK(1). Experimente mover a alavanca de cima para baixo e veja como o segundo asterisco se move.
JOYSTK(2) e JOYSTK(3) verificam o joystick esquerdo. JOYSTK(2) para a posição horizontal e JOYSTK(3) para a vertical. Veja como isso funciona na ilustração abaixo:

JOYSTICK esquerdo **JOYSTICK direito**



JOYSTICK (3) JOYSTICK (2) JOYSTICK (1) JOYSTICK (0)

Mais uma coisa. Cancele a linha 20 e mude a 60 para:

```
60 GOTO 30
```

Rode o programa e experimente mover qualquer joystick. Nenhum dos asteriscos vai se mover. Isto é devido ao fato de que o Computador não verifica a posição de nenhum joystick sem antes verificar o JOYSTK(0). Não precisa nem usar o JOYSTK(0) para mover alguma coisa. Basta mencioná-lo em uma instrução qualquer como, por exemplo:

```
20 A = JOYSTK(0)
```

Aproveite e mude a linha 60 para:

```
60 GOTO 20
```

Agora, apesar de não estar usando o JOYSTK(0), ele o está verificando, e a todos os outros (1, 2 e 3). Portanto, lembre-se: sempre que o Computador tiver que ver a posição de JOYSTK(1), JOYSTK(2) e JOYSTK(3), ele terá primeiro que verificar JOYSTK(0).

É MUITO SIMPLES PINTAR

Introduza o seguinte programa:

```
10 CLS(0)
20 H = JOYSTK(0)
30 V = JOYSTK(1)
40 SET(H,V/2,3)
50 SET(63-H,V/2,2)
60 GOTO 20
```

Rode o programa e veja que, ao movimentar o joystick direito, duas linhas serão traçadas na tela. A linha azul acompanhará os movimentos que você fizer. Já a linha amarela se moverá no sentido inverso. À medida que você leva a alavanca para a esquerda, a linha se desloca para a direita. Tome cuidado para não fazer movimentos muito rápidos, pois a verificação dos joysticks não é muito ágil. Se você preferir uma só linha para tentar algum desenho, é só cancelar a linha 50.
Uma observação importante é o truque usado nas posições verticais das instruções SET (foi usado V/2 no lugar de V). Como você já viu, a tela é dividida em 64 posições horizontais e 32 (de 0 a 31) verticais quando você usa instruções SET e RESET. Como a posição vertical dos joysticks — JOYSTK(1) e JOYSTK(3) — resulta em um valor entre 0 e 63 (64 posições diferentes), precisamos dividir esse valor por dois para obter um intervalo que caiba dentro da tela.
Nós ainda não usamos o joystick esquerdo. Vamos usá-lo para mudar a cor da figura. Desta vez teremos apenas uma linha de cada vez. Introduza:

```
40 C = JOYSTK(2)
50 IF C< =31 THEN C = 3
60 IF C>31 THEN C = 2
70 SET(H,V/2,C)
80 GOTO 20
```

<= significa menor ou igual a.

Ao mover o joystick esquerdo para a direita o traço fica amarelo (C = 1). Um movimento para a esquerda desse mesmo joystick muda a cor para azul (C = 4).
E os botões? Para que servem os botões dos joysticks? Em um videogame, eles são usados para disparar as armas nas "guerras espaciais" ou para "chutar" nos jogos. Mas, como é que verificamos se estão ou não pressionados em um determinado momento do programa?
Para responder a essas perguntas, vamos antes testar esse pequeno trecho do seu programa:

```
100 P = PEEK(65280)
110 PRINT P
120 GOTO 100
```

Agora, introduza:

```
RUN 100 ENTER
```

Esta instrução diz ao Computador para começar a executar o programa a partir da linha 100. Seu Computador vai entrar em um *loop* e ficar imprimindo 255 ou 127.
A instrução PEEK faz com que o Computador verifique o conteúdo da posição indicada dentro dos parênteses. No caso, ele verificará a posição 65280 e seu conteúdo será 255 ou 127.
Pressione o botão do joystick direito. Quando fizer isso, o conteúdo dessa posição de memória muda para 126 ou 254.

Se você pressionar os botões sem que o programa esteja sendo executado, vão aparecer mensagens como @ ABCDEFG ou HIJKLMNO.

Pressione o botão esquerdo agora (do joystick esquerdo). O conteúdo deve mudar para 125 ou 253.
A partir dessa informação, você pode fazer o Computador tomar qualquer atitude durante uma partida ou mesmo durante uma aplicação séria. Por exemplo, podemos fazer o Computador limpar a tela quando você pressionar o botão direito. Digite:

```
110 IF P=126 THEN 10
120 IF P=254 THEN 10
```

Cancele a linha 80 e acrescente:

```
130 GOTO 20
```

Agora rode o programa. Faça algumas figuras, mudando as cores quando achar necessário. Quando quiser começar uma nova figura, aperte o botão.

EXERCÍCIO DO CAPÍTULO

Usando RND deixe o seu Computador compor uma música e a cada duração do tom, exibir uma cor ao acaso.
Depois de 15 tons gerados exibir na tela 3 faixas com suas cores preferidas.

CAPÍTULO 5

um professor fantástico

Seu Computador tem todos os atributos de um professor nato. Antes de tudo, ele é paciente, incansável e consciente. Dependendo do programador (estamos falando de você, naturalmente), ele pode ser imaginativo, criativo e muito entusiasmado.
Assim sendo, vamos aprender com ele! Nós podemos usar RND para que o Computador nos treine em um problema de matemática.
Digite:

```
10 CLS
20 X=RND (15)
30 Y=RND (15)
40 PRINT "QUANTO E"" X "*" Y
45 INPUT A
50 IF A=X*Y THEN 90
60 PRINT "A RESPOSTA E"" X*Y
70 PRINT "MELHOR SORTE DA PROXIMA VEZ"
80 GOTO 100
90 PRINT "CORRETO!!!"
100 PRINT "PRESSIONE <ENTER> QUANDO
        ESTIVER PRONTO PARA OUTRA"
105 INPUT A$
110 GOTO 10
```

Este programa ensinará a você a tabuada de multiplicação, de 1 a 15, verificando sua resposta. Partindo deste programa, você pode conseguir que o Computador efetue tabelas de qualquer operação. Tente.
Para fazer o programa um pouco mais interessante nós podemos fazer com que o Computador mantenha uma contagem total de todas as respostas corretas.
Digite:

```
15 T=T+1
95 C=C+1
98 PRINT "VOCE ACERTOU "C" EM "T" PROBLEMAS"
```

T marca quantas questões foram feitas pelo Computador. Antes de você rodar o programa T é igual a zero. Aí, toda vez que o Computador voltar para a linha 15, ele adiciona 1 a T.

Quando você ligar o Computador, ou quando digitar NEW ENTER, todas as variáveis numéricas serão iguais a zero.

C faz a mesma coisa. Só que ela marca as respostas corretas. Como ela está na linha 95, o Computador não incrementará C a menos que você acerte.
Há muitas maneiras de deixar este programa mais interessante. Acrescente mais algumas linhas ao programa para ter o seguinte:
1 — Chamar você pelo nome.
2 — Premiar sua resposta correta com um som, acendendo também uma luz.

3 — Imprimir o problema e mensagens em sua tela.
4 — Manter uma contagem final em porcentagem das respostas corretas.
5 — Terminar o programa se você obtiver 10 respostas, em uma série, corretas.

Use sua imaginação.

VAMOS MELHORAR O VOCABULÁRIO DELE

Para aumentar o vocabulário de seu Computador, digite e rode este programa:

```
10 DATA UVAS, LARANJAS, PERAS
20 FOR X=1 TO 3
30 READ F$
40 NEXT X
```

Então o que aconteceu? Nada? Nada que você possa ver. Para ver o que o Computador está fazendo adicione esta linha e rode:

```
35 PRINT "F$="F$
```

A linha 30 diz ao Computador para:
1 — Procurar uma linha DATA.
2 — Ler o item 1 da lista — UVAS.
3 — Guardar UVAS na variável F$.
4 — Imprimir F$ na tela.

Na segunda vez ele faz a mesma coisa, só que lê o item 2 — LARANJAS. E assim por diante.
O que fazer para que o Computador leia a mesma lista outra vez? Digite:

```
60 GOTO 10
```

e rode o programa. Ele imprime OD ERRO IN 30. OD significa insuficiência de dados. O Computador já mostrou o conteúdo da sua linha DATA.
Digite esta linha e rode o programa:

```
50 RESTORE
```

Você se lembra como fazer para o Computador parar durante a execução de um programa? Pressione SHIFT @. Pressione qualquer tecla para que ele continue.

Agora fica como se o Computador não tivesse lido nada. O Computador lerá toda a lista outra vez.
O importante das linhas DATA é que você pode colocá-las em qualquer lugar do programa.

AGORA JÁ MELHORAMOS O SEU VOCABULÁRIO

Aqui estão mais algumas palavras e definições que você deve querer testar:

PALAVRAS	DEFINIÇÕES
10 DATA TACITURNO,	FALA POUCO
20 DATA LOQUAZ,	FALADOR
30 DATA VOCIFERANTE,	QUE ESBRAVEJA
40 DATA CONCISO,	BREVE
50 DATA EFUSIVO,	COMUNICATIVO

Agora vamos fazer com que o Computador escolha uma palavra qualquer da lista. Bem... há 10 itens na lista. Vamos ver se desse modo funciona:

```
60 N = RND (10)
70 FOR X=1 TO N
80 READ A$
90 NEXT X
100 PRINT "A PALAVRA ESCOLHIDA E':" A$
```

Execute várias vezes para ver se funciona. Ele não faz o que de fato queremos. O que nós queremos que o Computador faça é escolher uma palavra dos itens 1, 3, 5, 7 ou 9.
Felizmente, há um jeito de explicar isso ao Computador.
Digite:

```
65 IF INT (N/2) = N/2 THEN N = N-1
```

Rode o programa algumas vezes mais. Ele deverá funcionar, agora. INT diz ao Computador para apenas olhar a parte inteira do número e ignorar a parte decimal. Assim, o Computador vê INT (3.9) como 3.

Veja como a linha 65 trabalha. Digamos que o número que o Computador escolheu é 10. O Computador faz este cálculo:

```
INT (10/2)=10/2
INT (5) = 5
     5 = 5
```

Como isto é verdade, 5 é igual a 5, o Computador completa a parte THEN da linha e faz N igual a 9, isto é, 10-1.
Por outro lado, se o Computador escolher 9, ele faz o seguinte:

```
INT (9/2)=9/2
INT (4.5)=4.5
     4=4.5
```

Como isto não é verdade, 4 não é igual a 4.5, o Computador não subtrai 1 de N. 9 permanece 9.
Agora que o Computador está apto a ler uma palavra aleatoriamente, ele deve também ler a definição da palavra. Você pode fazer esta condição adicionando estas linhas no fim do programa:

```
110 READ B$
120 PRINT "A DEFINICAO E':" B$
```

Rode o programa várias vezes. Para que o Computador imprima uma palavra aleatoriamente e sua definição em seguida, adicione esta linha no início do programa:

```
5 CLEAR 100
```

Adicione, também, estas linhas no fim do programa.

```
130 RESTORE
140 GOTO 60
```

É deste modo que o Computador escolhe uma nova palavra ao acaso e sua definição de uma lista de dados.

Se você preferir, adicione mais algumas palavras e definições usando a linha DATA.

Para variar este programa você pode tentar Estados e capitais, cidades e municípios, palavras estrangeiras e seus significados e muitas outras idéias.

Vamos completar este programa? Então faça de modo que o Computador:

1 — Imprima somente a definição.
2 — Pergunte a você a palavra.
3 — Compare sua resposta com a palavra aleatória correta.
4 — Diga para você se a resposta está correta. Se não, mostrar a certa.

Aqui está o nosso programa:

```
  5 CLEAR 100
 10 DATA TACITURNO, FALA POUCO
 20 DATA LOQUAZ, FALADOR
 30 DATA VOCIFERANTE, QUE ESBRAVEJA
 40 DATA CONCISO, BREVE
 50 DATA EFUSIVO, COMUNICATIVO
 60 N = RND (10)
 65 IF INT (N/2) = N/2 THEN N = N-1
 70 FOR X = 1 TO N
 80 READ A$
 90 NEXT X
110 READ B$
120 PRINT "O QUE SIGNIFICA A PALAVRA:" B$
130 RESTORE
140 INPUT R$
150 IF R$ = A$ THEN 190
160 PRINT "ERRADO"
170 PRINT "A PALAVRA CORRETA E':" A$
180 GOTO 60
190 PRINT "CORRETO"
200 GOTO 60
```

AJUDA MATEMÁTICA POR COMPUTADOR

Resolver complicadas fórmulas matemáticas com bastante velocidade e precisão é uma coisa que seu Computador sabe fazer muito bem. Mas antes de introduzir você nesta loucura que são as fórmulas matemáticas, veremos alguns caminhos mais curtos para resolvê-las, sem implicar em programas muito extensos e complicados. Um modo fácil de manipulá-las é usando as sub-rotinas. Experimente este programa:

```
 10 PRINT "EXECUTANDO O PROGRAMA PRINCIPAL"
 20 GOSUB 500
 30 PRINT "DE VOLTA AO PROGRAMA PRINCIPAL"
 40 END
500 PRINT "EXECUTANDO A SUB-ROTINA"
510 RETURN
```

A linha 20 faz o Computador ir para a sub-rotina que começa na linha 500. O RETURN diz ao Computador para voltar à linha que vem imediatamente após o GOSUB. Cancele a linha 40 e veja o que acontece quando rodar o programa. Pronto. Na sua tela aparecerá o seguinte:

```
EXECUTANDO O PROGRAMA PRINCIPAL
EXECUTANDO A SUB-ROTINA
DE VOLTA AO PROGRAMA PRINCIPAL
EXECUTANDO A SUB-ROTINA
?RG ERRO IN 510
```

RG significa RETURN sem GOSUB. Você consegue ver por que tirando o END na linha 40 ocorre este erro?
O Computador segue o programa da mesma forma que antes de você cancelar a linha END. Ele vai para a sub-rotina na linha 500 por meio de GOSUB e retorna para a linha imediatamente após a GOSUB, isto é, linha 30.
Então, como você tirou a linha END, ele vai para a próxima que no programa é a sub-rotina. Quando ele chega no RETURN não sabe para onde deve voltar, pois não foi enviado anteriormente a esta sub-rotina por um GOSUB.
Aqui está a sub-rotina que acha um número na potência que você quiser:

```
   10 INPUT "DIGITE UM NUMERO";N
   20 INPUT "DIGITE A POTENCIA A QUAL ELE SERA'
      ELEVADO";P
   30GOSUB 2000
   40 PRINT:PRINT N"ELEVADO A POTENCIA"P"E""E
   50 GOTO 10
 2000 REM FORMULA DE POTENCIACAO
 2010 E=1
 2020 FOR X=1 TO P
 2030 E=E*N
 2040 NEXT X
 2050 IF P=0 THEN E=1
 2060 RETURN
```

Note que introduzimos algumas coisas novas nesse programa. Olhe a linha 40. Se você preferir pode combinar duas ou mais linhas de programa em uma só, usando dois pontos para separar as instruções. A linha 40 contém as linhas:

```
PRINT
e
PRINT N "ELEVADO A POTENCIA"P"E""E
```

A linha 2000 tem um REM. REM não significa nada para o Computador. Ele ignora qualquer linha que começa com REM. Você pode colocar linhas REM em qualquer lugar do seu programa. Assim seu programa tem um título, lembrando a você o que ele faz.

DÊ UMA MÃO AO COMPUTADOR

Como as fórmulas matemáticas são muito complexas, seu Computador precisa de alguma ajuda para entendê-las. Por exemplo, você quer que o Computador resolva este problema:

Divida a subtração de 20 – 4 por 8

Você quer que o Computador resolva este problema deste modo:

$$\frac{20 - 4}{8} = \frac{16}{8} = 2$$

Mas seu Computador resolve de outro modo. Vamos ver. Digite esta linha de comando:

PRINT 20 – 4/8 ENTER

Acontece que o Computador segue algumas regras.

A palavra operação significa alguma coisa que você quer que o Computador faça. Aqui estamos falando de "operações" de adição, subtração, multiplicação e divisão.

REGRAS ARITMÉTICAS

O Computador resolve os problemas aritméticos na seguinte ordem:

1. *Primeiro qualquer operação de multiplicação e divisão.*
2. *Depois operação de adição e subtração.*
3. *No caso de haver mais que uma multiplicação/divisão ou adição/subtração, as operações são resolvidas da esquerda para a direita.*

No problema acima o Computador segue estas regras. Ele resolveu a operação de divisão primeiro (4/8 = 0,5) e aí então a subtração (20 – 0,5 = 19,5). Para conseguir que o Computador resolva o problema de modo diferente você deve usar parênteses. Digite esta linha:

PRINT (20 – 4)/8 ENTER

Quando o Computador vê uma operação entre parênteses, ele a resolve antes de qualquer outra.

REGRAS DE PARÊNTESES

1. *Quando um Computador vê um problema contendo parênteses, ele resolve a operação dentro dos parênteses e depois resolve as demais operações.*

2. *Se houver parênteses dentro de parênteses o Computador resolve primeiro o parênteses mais interno e vai para o mais externo.*

ELE FAZ TODO O SERVIÇO

Com o que você aprendeu neste capítulo, você pode colocar as fórmulas complicadas em sub-rotinas. O programa a seguir usa duas sub-rotinas.

```
 10 INPUT "SEU DEPOSITO MENSAL";D
 20 INPUT "TAXA ANUAL DO BANCO";I
 30 I=I/12*0.01
 40 INPUT "NUMERO DE DEPOSITOS";P
 50 GOSUB 1000
 60 PRINT "VOCE TERA' $ "FV" EM "P"MESES"
 70 END
1000 REM FORMULA DE JUROS COMPOSTOS
1010 N=1+I
1020 GOSUB 2000
1030 FV=D*((E-1)/I)
1040 RETURN
2000 REM FORMULA DE POTENCIACAO
2010 E=1
2020 FOR X=1 TO P
2030 E=E*N
2040 NEXT X
2050 IF P=0 THEN E=1
2060 RETURN
```

Note que o programa tem uma sub-rotina "chamando" outra sub-rotina. Isto é perfeitamente possível com um GOSUB, que envia o Computador para cada sub-rotina, e um RETURN em cada sub-rotina para retornar à linha imediatamente após o GOSUB e continuar o processamento.

UM TALENTO COM PALAVRAS

Uma das maiores habilidades de seu Computador é seu talento com as palavras. Ele pode misturar, combinar ou separar palavras para o que você quiser. Como conseqüência deste talento você pode ensiná-lo a ler, escrever e desenvolver um diálogo respeitável. Para iniciar veja o seguinte:

```
10 PRINT "DIGITE UMA SENTENCA"
20 INPUT S$
30 PRINT "SUA SENTENCA TEM"LEN (S$)"CARACTERES"
40 INPUT "QUER TENTAR OUTRA",A$
50 IF A$ = "SIM" THEN 10
```

Gostou? LEN diz ao Computador para medir o comprimento da string S$. Sendo obediente, seu Computador conta cada caractere da sentença, incluindo os espaços e a pontuação.
Eis aqui outra experiência. Apague seu programa e digite este para fazê-lo compor um texto.

```
10 A$ = "E' O INICIO"
20 B$ = " "
30 C$ = "DE"
40 D$ = A$ + B$ + C$
50 E$ = "UMA TELA"
60 F$ = D$ + B$
70 G$ = "PLANA"
80 H$ = F$ + E$ + B$ + G$
90 PRINT H$
```

Aqui o Computador combina as strings. O sinal de adição é que indica para fazer isto. Uma precaução na combinação de strings — some esta linha em seu programa e rode:

```
100 I$ = H$ + H$ + H$ + H$
```

Quando você roda este programa, o Computador imprime OS ERRO NA 80. OS significa que falta espaço para strings. O Computador reserva somente 200 caracteres para trabalhar com strings. Coloque esta linha no início de seu programa para reservar mais espaço para strings:

```
5 CLEAR 500
```

Rode o programa novamente. Tenha cuidado para que sua variável string não contenha mais de 255 caracteres, pois isso recairia em um erro. Cada variável string deve conter um máximo de 255 caracteres.
Agora que o Computador já combina strings, deixe-o dividi-las. Digite e rode este programa.

```
10 INPUT "DIGITE UMA PALAVRA";P$
20 PRINT "A PRIMEIRA LETRA E': "LEFT $ (P$,1)
30 PRINT "AS DUAS ULTIMAS LETRAS SAO: "RIGHT$ (P$,2)
40 GOTO 10
```

Na linha 10 você introduz uma string em P$. Vamos dizer que a string seja a palavra INFORMÁTICA. Seu Computador desenvolve então vários cálculos nas linhas de 20 e 30 para obter a primeira letra à esquerda e as duas últimas à direita da string.

LEFT$ (P$,1) I	NFORMATI	CA RIGHT$ (P$,2)

Some esta linha ao programa:

```
5 CLEAR 500
```

Lembra-se de como se limpa um programa?
Digite NEW **ENTER**.

Assim seu Computador estabelece à parte os espaços para trabalhar com as strings. Agora introduza uma sentença no lugar de uma palavra.

Apague seu programa e digite este:

```
10 CLEAR 500
20 INPUT "DIGITE UMA SENTENCA";S$
30 PRINT "DIGITE UM NUMERO DE 1 A "LEN (S$)
40 INPUT X
50 PRINT "O TRECHO DA SENTENCA COMECARA' NO
   CARACTERE "X
60 PRINT "DIGITE UM NUMERO DE 1 A "LEN (S$) - X+1
70 INPUT Y
80 PRINT "O TRECHO TERA' "Y"CARACTERES DE
   COMPRIMENTO"
90 PRINT "O TRECHO E':"MID$ (S$,X,Y)
100 GOTO 20
```

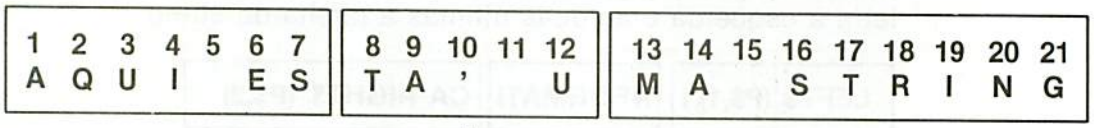
Rode este programa mais algumas vezes para ver se você entendeu a função de MID$.
Eis aqui como o programa funciona. Como sentença digite "AQUI ESTA' UMA STRING"

```
S$  → AQUI ESTA' UMA SEQUENCIA
```

Na linha 30, o Computador primeiro calcula o comprimento de S$ —21 caracteres. Ele então pede a você para escolher um número de 1 a 21. Suponhamos que você escolheu o número 8.
O Computador, então, na linha 60, pede para escolher um número de 1 a 13 (20 – 8) + 1. Escolha 5.

```
X  → 8
Y  → 5
```

Na linha 90, o Computador dá uma MID string de S$ que inicia no caractere número 8 e tem 5 caracteres de comprimento.



-5-
MID (S$,8,5)

Vamos testar mais um exemplo contendo MID$. Apague seu programa e digite:

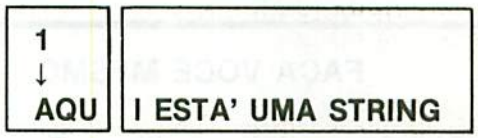
```
10 INPUT "DIGITE UMA SENTENCA";S$
20 INPUT "DIGITE UMA PALAVRA NA SENTENCA";P$
30 L=LEN (P$)
40 FOR X=1 TO LEN (S$)
50 IF MID$ (S$,X,L) = P$ THEN 90
60 NEXT X
70 PRINT "SUA PALAVRA NAO ESTA' NA SENTENCA"
80 END
90 PRINT "——— COMECA NO CARACTERE NUMERO" X
```

Rode este programa algumas vezes mais. Aqui está como ele funciona.
Digamos que você digitou a string anterior, e a palavra que você introduziu para P$ é "UMA". Na linha 30, o Computador calcula então o comprimento de P$ — 3 caracteres.

```
S$——— AQUI ESTA' UMA STRING
P$——— UMA
L——— 3
```

O Computador, através do ciclo FOR/NEXT (linha 40-90), conta cada caractere em S$, começando com o 1.º caractere e terminando com o caractere de número 21, que corresponde ao comprimento de S$ — LEN(S$).

Cada vez que ele conta um novo caractere o Computador olha para um novo trecho MID$. Cada trecho inicia no caractere X e tem L ou 3 caracteres de comprimento.
Por exemplo, quando X é igual a 1, o Computador olha para este trecho:



-3-

MID$ (S$,1,3)

Na quarta vez que passar pelo *loop*, quando X for igual a 4, o Computador olha para isto:

-3-

MID$ (S$,4,3)

E finalmente acha a MID string quando X for igual a 11.

Você está começando a achar que o Computador pode corrigir suas sentenças. Ele pode ser programado para ajudá-lo na escrita e datilografia.
Imagine que você tem uma frase que você quer aumentar:

```
10 A$ = "MUDAR UMA SENTENCA"
```

Inserindo estas palavras no início da sentença:

```
E' FACIL
```

O programa deve imprimir esta nova sentença:

```
E' FACIL MUDAR UMA SENTENCA
```

FAÇA VOCÊ MESMO

.....

.....

.....

.....

Este é nosso programa:

```
10 A$ = "MUDAR UMA SENTENCA"
20 B$ = "E' FACIL"
30 C$ = B$ + " " + A$
40 PRINT C$
```

Agora veja o que você pode adicionar ao programa para fazer com que o Computador:

1. Ache o início do trecho:

```
UMA SENTENCA
```

2. Apague esse trecho formando uma nova string:

```
E FACIL MUDAR
```

3. Adicione estas palavras no fim da string:

```
QUALQUER COISA QUE VOCE QUEIRA
```

FAÇA VOCÊ MESMO

.....

.....

.....

.....

Sugestão: *Para formar a string E' FACIL MUDAR, você precisa obter a parte esquerda da sentença E' FÁCIL MUDAR UMA SENTENCA.*

Resposta:

```
10 A$ = "MUDAR UMA SENTENCA"
20 B$ = "E' FACIL"
30 C$ = B$ + " " + A$
40 PRINT C$
50 Y = LEN ("UMA SENTENCA")
60 FOR X = 1 TO LEN (C$)
70 IF MID$ (C$,X,Y) = "UMA SENTENCA" THEN 90
80 NEXT X
85 END
90 D$ = LEFT$ (C$,X-1)
100 E$ = D$ + "QUALQUER COISA QUE VOCE QUEIRA"
110 PRINT E$
```

Este tipo de programa é a base de um programa de processamento de textos — um programa para reduzir gastos com datilógrafos...

ELE ESTÁ DE OLHO EM VOCÊ

Você verá agora que o Computador pode observá-lo, ficando atento e reagindo a tudo que você fizer. Por estar atento, queremos dizer prestar atenção ao teclado para ver se você está pressionando alguma tecla. É a palavra INKEY$ que faz isto possível.
Digite:

```
10 A$ = INKEY$
20 IF A$ < >"" GOTO 50
30 PRINT "VOCE NAO PRESSIONOU NADA"
40 GOTO 10
50 PRINT"A TECLA QUE VOCE PRESSIONOU E' ———" A$
```

Lembre-se que < > significa diferente de.

""uma string vazia —— sem nada.

INKEY$ manda o Computador olhar o teclado para ver se você pressionou alguma coisa. O Computador faz isto em alta velocidade. Enquanto você pressiona alguma tecla, o Computador faz pelo menos 20 verificações.
O programa armazena a letra correspondente à tecla pressionada em A$. Se A$ é igual a "", o Computador imprime "VOCE NAO PRESSIONOU NADA" e retorna à linha 10 para verificar o teclado novamente. Caso contrário, o Computador vai à linha 50 e imprime a tecla.

Vamos verificar, digite isto e rode o programa:

```
60 GOTO 10
```

Não importa quão rápido você é, o Computador é muito mais. Elimine a linha 30 do programa. Assim você verá qual tecla pressionou.

COMPUTADOR E MÚSICA

Tente usar INKEY$ para fazer um piano de seu teclado. Veja no Apêndice D a lista das notas de Dó natural até Dó maior:

Dó C-89	Mi E-125	Sol G-147	Si B-170
Ré D-108	Fá F-133	Lá A-159	Dó C-176

Você pressiona uma tecla e o computador emite uma nota. Apague o programa anterior e digite:

```
10 A$=INKEY$
20 IF A$=""THEN 10
30 IF A$="A" THEN T=89
40 IF A$="S" THEN T=108
50 IF A$="D" THEN T=125
60 IF A$="F" THEN T=133
70 IF A$="G" THEN T=147
80 IF A$="H" THEN T=159
90 IF A$="J" THEN T=170
100 IF A$="K" THEN T=176
110 IF T=0 THEN 10
120 SOUND T,5
130 T=0
140 GOTO 10
```

Rode o programa. Bem, o que você está esperando? Toque uma música. Agora já é possível. Digite qualquer tecla da terceira fila do teclado de A a K.

Como isto mudaria o programa?
120 SOUND T,1

Por que o programa não funcionará se você usar INPUT ao invés de INKEY$?

Se você usar INPUT, o Computador vai esperar até que você pressione ENTER antes de saber o que você digitou. Com INKEY$ ele vê tudo que você digita. Não precisa de aviso.
Há outro modo de escrever este programa usando as linhas READ e DATA.
Você sabe como?

Eis aqui o que fizemos:

```
 10 A$=INKEY$
 20 FOR X=1 TO 8
 30 READ B$, T
 40 IF A$=B$ THEN SOUND T,5
 50 NEXT X
 60 RESTORE
 70 GOTO 10
 80 DATA A, 89, S, 108
 90 DATA D, 125, F, 133
100 DATA G, 147, H, 159
110 DATA J, 170, K, 176
```

Usando linhas DATA e READ será fácil acrescentar mais notas ao repertório de seu Computador.

Agora vamos dar um programa que vai testar sua agilidade mental!

```
 10 X=RND(45)+4
 20 Y=RND(45)+4
 30 PRINT" QUANTO E""X "+" Y
 40 T=0
 45 B$=""
 50 A$=INKEY$
 60 T=T+1
 70 SOUND 128,1
 80 IF T=45 THEN 200
 90 B$=B$+A$
 95 IF LEN(B$)< >2 THEN 50
100 IF VAL(B$)=X+Y THEN 130
110 PRINT "ERRADO",X"+"Y"="X+Y
120 GOTO 10
130 PRINT"CORRETO"
140 GOTO 10
200 CLS(4)
210 SOUND 180,30
220 PRINT "SEU TEMPO ACABOU"
```

O programa se resume em dar dois números escolhidos ao acaso, para você calcular a soma. Se você der o resultado errado, o Computador imprime ERRADO, dá o resultado correto e passa para outro teste. Se você colocar o valor certo, ele imprime CORRETO

e passa novamente para novo teste. Esta é a função principal do nosso programa. Vamos ver agora seus detalhes.

Quando você demorar para dar a resposta, o Computador imprime a mensagem que seu tempo acabou e encerra as rodadas de teste. Para novos testes temos que digitar RUN ENTER.

Note que os números que o Computador vai escolher estão entre 1 e 49. Assim a soma não terá 1 ou 3 dígitos. Se quando for feito o teste, você apenas digitar um número, ele vai esperar (um certo tempo é claro!) que você digite o outro, para depois comparar o resultado. Tudo isto é feito nas linhas 45, 50, 90, 95 e 100. Enquanto espera sua resposta ele emite um som. Quando seu tempo termina, ao mesmo tempo que outro som é emitido, a tela muda de cor.

Tudo bem? Vamos agora testar seu raciocínio? A partir deste programa dê asas a sua imaginação e crie outros tipos de jogos.

E a linha 100? Você já entendeu qual é a função de VAL? Muito fácil, não? Ele apenas dá o valor numérico de uma string para podermos igualá-la a um número.

EXERCÍCIO DO CAPÍTULO

Escreva um programa em que:

1. O Computador pede a você para introduzir (INPUT):
   - **a.** Uma sentença
   - **b.** Uma frase a ser retirada da sentença
   - **c.** Colocar uma nova frase no lugar da que foi retirada
2. O Computador então imprime (PRINT) a nova sentença.

Isto pode levar algum tempo, mas você tem tudo que precisa para escrevê-lo. Nossa resposta está no Apêndice A.

CAPÍTULO

6

corrigindo erros

Desde o começo deste livro você vem gastando muito tempo para redigitar linhas de programa quando necessitava fazer alguma modificação.
O COLOR BASIC oferece um modo bem mais fácil de fazer essas mudanças em seus programas. Nós chamaremos esta característica de "edição" de programa. Com ela você poderá fazer toda sorte de modificação em seus programas.

MUDE ESSA LINHA!

Suponha que você cometeu um erro quando digitava seu programa e a linha ficou assim:

```
20 IH MID$ (S$,X,L$)=P$ THEN PINT "VERDE"
```

Ela deve ser corrigida. Então, para editar a linha 20, digite:

```
EDIT 20 ENTER
```

e a tela exibirá:

```
20 IH MID$(S$,X,L$)=P$ THEN PINT "VERDE"
20 ■
```

Você pode agora fazer as mudanças necessárias.

Para movimentar o cursor no modo de Edição, pressione ESPAÇO para mover para frente, ou ←, para trás.

TROCANDO AS LETRAS

Primeiro você precisa mudar o H de IH para F:

- Posicione o cursor "em cima" da letra incorreta (H);
- Pressione C para "trocar" a letra;
- Pressione a tecla com a letra correta (F).

A mudança está feita. A linha então será:

```
20 IF ■
```

Para ver a linha toda, pressione L e a linha será listada na sua forma atual, mas ainda poderá ser alterada.
Para trocar mais que um caractere no modo de Edição:
- Posicione o cursor sobre a primeira letra a ser mudada;
- Digite o *número* de caracteres que serão trocados;
- Pressione C para "trocar";
- Digite os novos caracteres.

Nota: *Uma vez que você digita o* número *seguido de* C *para trocar, você deve pressionar* n *teclas antes que possa sair do subcomando. Ou seja, você deve completar a "troca".*

SAIA DAÍ!

O primeiro erro já foi arrumado, mas há ainda um cifrão a mais dentro dos parênteses. O L é uma variável numérica e não string. O cifrão precisa ser apagado. Para isso, devemos proceder do mesmo modo que na troca:

- Posicione o cursor "sobre" o caractere a ser apagado;
- Pressione D;

e a operação está completada. Neste caso:

```
20 IF MID$(S$,X,L ■
```

Quando dizemos "caracteres" estamos dizendo "espaços em branco" também.

Para verificar se o caractere foi realmente apagado, pressione L para listar toda a linha e continue editando, ou pressione ENTER para terminar a operação de edição.

Há outro modo para apagar um determinado número de caracteres:

- Posicione o cursor no topo do primeiro caractere a ser apagado;
- Digite o *número* especificando quantos caracteres serão apagados;
- Pressione D.

NÃO ACABEI AINDA!

Agora a linha 20 deve estar assim:

```
20 IF MID$(S$,X,L)=P$ THEN PINT "VERDE"
```

Ainda está errado. Falta o R no PINT. Portanto, mãos à obra:

- Posicione o cursor "em cima" do caractere onde a inserção será feita;
- Pressione I;
- Digite os caracteres que serão inseridos.

Há uma grande diferença, entretanto, entre inserção e as características de edição já descritas. Mesmo depois de ter completado a inserção, você ainda permanecerá neste modo. Assim, caso você tente avançar o cursor (pressionando ESPAÇO), você vai estar inserindo espaços.
Para sair do modo de Inserção, você deve pressionar ENTER para terminar de uma vez a edição da linha ou então pressionar as teclas SHIFT e ↑ juntas para parar de inserir, mas continuar mexendo na linha (editando).

EMENDAS E REMENDOS

O COLOR BASIC tem uma função que combina as características de apagar e inserir. Esta função é chamada "Corte", porque ela deixa você alterar uma linha cortando o seu fim e inserindo novos caracteres. Para usar esta característica, você precisa:

- Posicionar o cursor "sobre" o *primeiro* caractere que será cortado;
- Pressionar H;
- Digitar os novos caracteres;
- Pressionar ENTER para desistir, ou SHIFT ↑ para continuar a edição.

Suponha que a linha 20 fosse assim:

```
20 IF MID$(S$,X,L)=V$ THEN 70
```

e *que você* quisesse mudar tudo depois do sinal de igual (=). Você precisa posicionar o cursor sobre o V de V$:

```
20 IF MID$(S$,X,L) = ■
```

Agora pressione H e digite a nova informação:

```
20 IF MID$(S$,X,L)="FIM" THEN PRINT "FIM"
```

Você está parado no modo de Inserção. Pressione ENTER para terminar a edição, ou SHIFT ↑ para parar de inserir e continuar editando.

AUMENTE ESTA LINHA!

Pode ser que você queira inserir mais uma instrução na linha 20. Um modo para fazer isto é redigitar a linha toda, mas isso é um trabalho desnecessário. Um outro modo é acionar o modo de Edição, levar o cursor até o final da linha, e aí então usar a opção de inserção — este é menos grosseiro, mas ainda trabalhoso.
O seu Computador tem uma solução simples para este problema. Você pode adicionar o que quiser na linha, com a opção de ampliar no modo de Edição:

- Pressione X (para ampliar);
- Digite os caracteres adicionais;
- Pressione ENTER para terminar a edição ou SHIFT ↑ para continuar.

Por exemplo, para ampliar a linha 20, acione o modo de Edição e sua tela exibirá:

```
20 ■
```

Pressione X e o cursor imediatamente passará para o final da linha, colocando você no modo de Inserção.

```
20 IF MID$(V$,X,L)= "FIM" THEN PRINT "FIM" ■
```

Você está no modo de Inserção e pode começar a acrescentar texto à:

```
20 IF MID$(V$,X,L)= "FIM" THEN PRINT "FIM": GOTO 210
```

ONDE ESTÁ AQUELA LETRA?

Outro modo de mover o cursor para frente rapidamente é fazer o Computador procurar na linha do programa por um determinado caractere.
Se, por exemplo, você está no modo de Edição e decide tirar o L da linha 20, como moveria o cursor sem pressionar ESPAÇO?
A resposta é a seguinte:

- Pressione S;
- Pressione a tecla do caractere que você quer modificar (nesse caso, L).

O cursor buscará por toda linha até encontrar o caractere especificado. Você pode então trocar a letra ou usar qualquer outra opção de edição. Isto funciona bem em uma linha igual à linha 20, que tem uma única ocorrência de uma determinada letra (apenas um L), mas o que acontece numa linha de programa que tem mais de uma ocorrência? Neste caso, o que acontece se você coloca inadvertidamente um $ extra na variável X?

```
20 IF MID$(S$,X$,3)=P$ THEN PRINT "VERDE"
```

Se você simplesmente disser ao Computador para buscar um $, ele vai parar no primeiro. Você pode dizer ao Computador para buscar o terceiro $, assim: entrar o modo de Edição da linha 20, pressionar 3 (significa "terceira" ocorrência) seguido pelo S (para "busca"). E, então, pressionar $ (o caractere que você quer). Sua tela terá:

```
20 IF MID$(S$,X ■
```

Você pode então desenvolver qualquer operação de edição. Neste caso, você quer tirar o $. Lembra-se de como se faz isso? Bem, então faça!

CORTANDO A CONVERSA

Em certos casos precisamos cortar um trecho inicial de uma linha. O K permite eliminar (apagar) tudo até a enésima ocorrência de um dado caractere. Estando no modo de Edição, faça o seguinte:

- Digite o número da ocorrência do primeiro caractere a ser mantido na linha (ele não será eliminado);
- Pressione K;
- Pressione a tecla correspondente a esse caractere.

Vamos supor que você quer eliminar a primeira parte da linha 20 (tudo que vem antes do comando PRINT). Primeiro entre no modo de edição da linha 20, pressione 2 (para a segunda ocorrência) e então K. Depois, pressione P (a primeira letra de texto que restará). Se você então listar a linha, verá o seguinte:

```
20 PRINT "VERDE"
```

MOVENDO O CURSOR

Nós mostramos a você como mover o cursor para frente ou para trás em um movimento relativamente lento. Mostramos também como movê-lo para frente rapidamente durante uma determinada operação de edição (tal como a busca). Algumas vezes, entretanto, é preciso mover o cursor rapidamente mas sem usar nenhuma opção de edição. Eis como se faz:

- Digite o número de espaços que você quer que o cursor avance;
- Pressione ESPAÇO

Por exemplo, se você pressionar 5 ESPAÇO, o cursor avançará 5 caracteres. Você pode então acionar qualquer função de edição.

Se você pressiona L para listar a linha durante o uso da inserção, você vai inserir a letra "L" na linha de programa em vez de listar a linha.

Para o mesmo movimento, mas em sentido contrário, troque o ESPAÇO por ←. Por exemplo, se o cursor está posicionado no final da linha 20 e você quer retroceder 7 espaços:

```
20 IF MID$(S$,X,L)=P$ THEN PRINT "VERDE"
```

OPÇÕES DE EDIÇÃO

Sintaxe do comando	Operação desenvolvida
EDIT (n.º da linha)	Edita a linha especificada
L	Lista a linha que está sendo editada
C novo caractere	Troca um caractere
n C novo caractere	Troca os próximos *n* caracteres por novos caracteres
I	Insere caracteres
D	Anula um caractere
n D	Anula *n* caracteres
H	Corta o resto da linha a partir do cursor e permite a inserção de texto
X	Amplia a linha (cursor se move para o fim da linha e o modo de Inserção é ativado)
S caractere	Busca pela 1.ª ocorrência do caractere especificado
n S caractere	Busca pela enésima ocorrência do caractere especificado
K	Elimina (anula) o resto da linha
n K caractere	Elimina (anula) até a enésima ocorrência do caractere especificado
n ESPAÇO	Avança o cursor *n* espaços; se *n* é omitido, 1 é usado
n ←	Retorna o cursor *n* espaços; se *n* é omitido, 1 é usado

Pressione 7 (o número de espaços a mover) seguido por → e sua tela exibirá:

```
20 IF MID$(S$,X,L)=P$ THEN PRINT
```

EDITAR NO ATACADO

Uma função muito útil quando estamos corrigindo um programa extenso é aquela que permite mexer em várias linhas ao mesmo tempo. Por exemplo, se você quer apagar as linhas de programa entre 10 e 50, basta digitar:

```
DEL 10-50 ENTER
```

Se você quer apagar apenas a linha 15, use:

```
DEL 15
```

e assim por diante.

SINTAXE DO COMANDO	OPERAÇÃO DESENVOLVIDA
DEL -	Apaga todo o programa da memória
DEL nº da linha	Apaga a linha especificada
DEL nº da linha -	Apaga a linha especificada até o fim do programa
DEL nº da linha - nº da linha	Apaga do 1º nº da linha ao 2º nº da linha inclusive
DEL - nº da linha	Apaga todas as linhas do início do programa à linha especificada, inclusive

RENUMERAR AS LINHAS

Uma outra função que opera "no atacado" é a que permite pegar um trecho do programa na memória e mudar a numeração das linhas. Essa função tem várias aplicações tanto para abrir espaço na memória, para inserir novas linhas, como também para organizar um programa já depurado.

RENUM nova linha, linha inicial, incremento

nova linha especifica o novo número da primeira linha a ser renumerada. É opcional; se omitido, é usado 10.

linha inicial especifica o nº da linha no programa original onde você quer iniciar a renumeração. Também é opcional; se omitido o programa inteiro será renumerado.

incremento especifica o intervalo a ser usado na numeração das linhas. Como as demais, também é opcional; se omitido, é usado 10.

RENUM não altera a ordem das linhas de programa. Apenas atribui novos números à ordem existente.

Tente renumerar as linhas do programa

```
 5 CLS
10 FOR I = 1 TO 200 STEP 5
20 SOUND I,2
25 PRINT "TOM = "I
30 NEXT I
40 GOTO 5
```

Se você quiser que a primeira linha do programa seja a de número 100 com um intervalo de 5, digite:

```
RENUM 100,5,5
```

O processo de renumeração começará pela linha 5. A primeira nova linha do programa terá o número 100. Daí em diante, os números serão incrementados de 5. Liste o programa renumerado:

```
100 CLS
105 FOR I = 1 TO 200 STEP 5
110 SOUND I,2
115 PRINT "TOM = "I
120 NEXT I
125 GOTO 100
```

Note que o Computador também renumera o número da linha depois da instrução GOTO, na linha 125.
Lembre-se de que todos os parâmetros de RENUM são opcionais. Conseqüentemente, há diversas variações de RENUM. Por exemplo, se você digitar:

```
RENUM ENTER
```

o Computador renumerará o programa inteiro sendo 10 o número da primeira linha e 10 o incremento. E se você quiser renumerar todas as linhas depois da linha 5? Tente, digitando:

```
RENUM 100, 10, 100 ENTER
```

Agora, liste o programa. Você verá que o Computador não mexe na linha 5, mas renumera todas as linhas subseqüentes para 100, 110 etc. Se você digitar:

```
RENUM 10000, 100 ENTER
```

o Computador renumerará a linha 100 e todas as linhas mais altas. A primeira linha renumerada tornar-se-á linha 10000, e um incremento de 10 será usado como intervalo. O programa inteiro será renumerado, iniciando com um número de nova linha 100, e incrementado de 100. O último modo para usar RENUM é digitar:

```
RENUM, ,5
```

O programa inteiro será renumerado, iniciando com um número de nova linha 10 e usando um incremento de 5.

CUIDADOS AO RENUMERAR

RENUM não pode ser usado para trocar a ordem das linhas de programa. Por exemplo, se seu programa original tem linhas numeradas 10, 20 e 30, então o comando:

```
RENUM 15, 30
```

é ilegal, pois teria que colocar a linha 30 na frente da linha 20. Neste caso, um ?FC ERRO resultará, e o programa original não será alterado.

RENUM não criará números de *nova linha* maior que 63999. Neste caso, o Computador avisa do erro e não mexe no programa. Se um número de linha não existente é mencionado no seu programa original, RENUM imprime uma mensagem de advertência, do tipo:

```
ULxxxxNAyyyy
```

onde: xxxx é o número da linha original
yyyy é o novo número da linha contendo xxxx

RENUM vai renumerar o programa apesar da mensagem de advertência, embora a linha indefinida não seja renumerada.

CAPÍTULO 7

grave em fita

Você logo estará escrevendo longos programas. E certamente vai querer usá-los outras vezes. Como fazer isso se quando desligamos o nosso Computador o nosso programa é perdido? Devemos, então, antes disso guardá-lo. Mas onde? Os sistemas mais sofisticados (e caros!) possuem unidades especiais com discos magnéticos, que podem guardar os programas bem como as informações que eles utilizam. O seu Computador também pode usar esses discos magnéticos, mas aqui nós vamos falar de um meio muito mais barato e simples... uma fita cassete comum!

O uso da unidade de disco está descrito no livro que a acompanha.

Praticamente qualquer gravador cassete comum pode ser conectado ao seu Computador. O que pode variar são os nomes das tomadas onde serão feitas as ligações e o ajuste do volume de reprodução, que veremos mais à frente. As tomadas (*jacks*) que usaremos no gravador são três e têm as seguintes funções:

1. Controle remoto do motor do gravador. Normalmente esta tomada é indicada pela palavra REMOTE ou simplesmente REM. O Computador vai usar esta tomada para controlar a operação do gravador. Ela tem um diâmetro menor que as outras.

2. Receber as informações que o Computador envia para gravá-las em fita. Esta tomada em geral recebe o nome de AUXILIAR ou AUX. Quando usado para gravar um programa na fita, o Computador deve ser ligado como um sintonizador ou toca-discos ao gravador(MIC).

3. Enviar a informação gravada na fita para o Computador. Quando você "tocar" a fita que tem o seu programa, será por esta tomada que o Computador receberá a informação. Ela normalmente leva o nome EARPHONE (EAR). Se os nomes dos *jacks* não coincidirem com os de seu gravador, procure no manual deste identificá-los pelas funções que realizam.

Para usar o gravador, faça o seguinte:

1. Separe o gravador que você vai usar, a fita, de preferência nova, e o cabo de conexão que vem com o seu Computador.

2. Conecte o cabo na tomada K-7, atrás de seu Computador, e no gravador da seguinte forma:
   - O plugue cinza menor na tomada REMOTE.
   - O plugue cinza maior na tomada AUX.
   - O plugue preto na tomada EARPHONE.

3. Ligue o gravador à rede elétrica.

Uma vez ligado o gravador ao Computador e à rede, você já pode gravar o seu primeiro programa. Para isso, siga estas instruções:

1. Digite o programa que você quer gravar e rode-o para ter certeza de que ele está operando bem. Faça todas as modificações e correções antes da gravação.

2. Coloque a fita no gravador. Posicione-a no início e pressione os botões PLAY e RECORD do gravador ao mesmo tempo.

3. Dê um nome ao programa que você vai gravar. Você pode usar qualquer nome de até 8 letras. Nós, por exemplo, usamos "NOME".

4. Grave o seu programa na fita, com o seguinte comando: CSAVE "NOME" ENTER.

Você pode usar qualquer palavra de até 8 letras no lugar de NOME.

O motor do gravador vai começar a girar, e você estará gravando o programa que digitou no Computador em uma fita, como se fosse música! Observe a tela. Quando o Computador avisar:

```
OK
```

e o motor parar, seu programa já terá sido gravado. Ele foi apenas copiado e permanece ainda na memória do Computador.

CARREGANDO

Para inverter o processo e carregar (copiando) o programa que está na fita para a memória do Computador proceda da seguinte forma:

1. Em primeiro lugar, rebobine a fita (coloque-a de novo no início), e ligue os cabos do gravador como já foi mostrado.

2. Aperte o botão PLAY do gravador. Ajuste o volume do gravador conforme recomendado no seu Manual de Operação.

3. Digite NEW e pressione ENTER para tirar qualquer programa da memória.

4. Use o comando CLOAD com o nome do seu programa. No nosso exemplo:

```
CLOAD "NOME" ENTER
```

O motor do gravador vai começar a funcionar. Observe a tela. A letra

```
P
```

vai aparecer no canto superior esquerdo. Isso significa que o Computador está procurando o seu programa. Quando o Computador o encontrar, ele vai mostrar a letra E seguida pelo NOME do programa, assim:

```
E NOME
```

Essa mensagem vai aparecer no alto da tela. Depois disso, quando o Computador imprimir a mensagem

```
OK
```

e o motor parar, o carregamento estará terminado. Agora você já pode rodar seu programa (RUN e ENTER).

Se existirem vários programas gravados na mesma fita, o Computador vai mostrar o nome de cada programa encontrado até chegar naquele que tem o nome que você pediu.

GRAVANDO MAIS DE UM PROGRAMA

Se você tentar carregar um programa que não está na fita, o Computador não vai parar de procurar. Nesse caso você deve usar as teclas RESET, pressionando as duas ao mesmo tempo.

Para gravar vários programas na mesma fita, você deve tomar cuidado para não gravar um programa em cima de outro. Aqui está um bom meio de posicionar a fita no final do seu último programa:

1. Rebobine totalmente a fita.
2. Pressione o botão PLAY do gravador.
3. Digite SKIPF e o nome do último programa que você gravou na fita. Por exemplo, se o último programa for o nosso velho conhecido "NOME", use:

```
SKIPF "NOME" ENTER
```

O Computador exibe no canto esquerdo superior o nome de todos os programas que ele encontra na fita. Quando ele chega ao fim do programa desejado, o motor do gravador pára e:

```
OK
```

aparece na tela.

4. A essa altura a sua fita já estará posicionada. Pressione as teclas PLAY e RECORD do gravador, dê um nome ao novo programa e grave-o (CSAVE). Se você não se lembrar do nome do último programa, use qualquer um que você ainda não tenha usado. O Computador vai mostrar o nome de cada programa que ele encontrar. Quando chegar no fim da fita, vai aparecer uma mensagem I/0 ERRO. Não se preocupe com ela. Assim você pode ver o nome de todos os programas e com certeza reconhecer aquele que você queria. Com esse nome você dá o comando SKIPF outra vez para posicionar a fita convenientemente. (Não se esqueça de rebobinar a fita.)

DICAS DA BOA GRAVAÇÃO

Aqui estão algumas sugestões de como se fazer boas gravações:

- Quando você não estiver usando o gravador para gravar ou carregar um programa, não deixe os botões RECORD e PLAY pressionados. Libere-os com STOP.
- Não tente regravar uma fita já usada pelo Computador. Apesar do processo de gravação apagar a informação anterior, podem ficar sinais suficientes para confundir a nova gravação. Se você pretende usar a mesma fita uma segunda ou terceira vez, utilize um desmagnetizador de boa qualidade para limpá-la completamente.

Agora digite quantos programas quiser, pois você já sabe como guardá-los para uso futuro.

CAPÍTULO

8

alguns retoques

Antes de prosseguir, há mais algumas palavras em BASIC que queremos mostrar. Você deve conhecê-las. Elas, na certa, tornarão a programação mais fácil.
A primeira palavra é STOP. Digite e rode este programa:

```
10 A=1
20 A=A+1
30 STOP
40 A=A*2
50 STOP
60 GOTO 20
```

O Computador imprime:

```
BREAK NA 30
OK
```

O Computador vai parar o programa quando chegar na linha 30. Neste ponto você pode digitar uma linha de comando para ver o que o seu programa fez até agora. Por exemplo, digite:

```
PRINT A ENTER
```

O Computador imprime 2, o valor de A, quando ele parou de executar o programa. Agora digite:

```
CONT ENTER
```

O Computador continua a executar o programa de onde parou. Em outras palavras, ele continua executando o programa da linha 40. Aí ele imprime:

```
BREAK NA 50
```

O segundo lugar tem um STOP. Agora você pode digitar

```
PRINT A ENTER
```

novamente. Ele imprime 4, que é o valor de A na linha 50. Digite CONT novamente e o Computador pára (na volta) na linha 30. Se você colocar PRINT A, ele imprimirá 5, o valor de A na linha 30. STOP e CONT são usadas quando seu programa não está trabalhando como você espera. Ao colocar a linha STOP no seu programa, você pode analisar o que está acontecendo de errado.

PARA PROGRAMAS AMBICIOSOS

Digite NEW para limpar a memória e então digite:

```
PRINT MEM ENTER
```

O Computador imprime quanto espaço para armazenar programas e dados está a sua disposição na memória do Computador. Quando você está digitando um programa muito extenso, vai querer que o Computador execute um PRINT MEM de tempo em tempo, para ver se você está trabalhando dentro da memória.

Para economizar memória, você pode omitir espaços em seu programa antes e depois da pontuação, operadores e palavras em BASIC.

AJUDA NA DIGITAÇÃO

Digite este programa:

```
 10 INPUT "DIGITE 1, 2, ou 3"; N
 20 ON N GOSUB 100, 200, 300
 30 GOTO 10
100 PRINT "VOCE DIGITOU 1"
110 RETURN
200 PRINT "VOCE DIGITOU 2"
210 RETURN
300 PRINT "VOCE DIGITOU 3"
310 RETURN
```

Rode o programa.
A linha 20 poderia ser substituída por estas 3 linhas:

```
18 IF N=1 THEN GOSUB 100
20 IF N=2 THEN GOSUB 200
22 IF N=3 THEN GOSUB 300
```

Simplesmente temos uma quantidade menor de linhas para digitar quando usamos ON...GOSUB.
ON...GOSUB diz ao Computador para olhar para o número que segue ON — neste caso o número é N. Se ele é 1, o Computador vai para a sub-rotina iniciada na linha do número seguinte a GOSUB. Se N é 2, o Computador vai para a sub-rotina iniciada na linha do segundo número; se N é 3, o Computador vai para a sub-rotina iniciada na linha do terceiro número.
E se N for igual a 4? Como não há 4 números de linha, o Computador vai simplesmente para a próxima linha do programa.

Eis aqui um programa que usa ON...GOSUB:

```
  5 FOR P=1 TO 600: NEXT P
 10 CLS : X=RND (100); Y=RND(100)
 20 PRINT "(1) ADICAO"
 30 PRINT "(2) SUBTRACAO"
 40 PRINT "(3) MULTIPLICACAO"
 50 PRINT "(4) DIVISAO"
 60 INPUT "QUE EXERCICIO (1-4)"; R
 70 CLS
 80 ON R GOSUB 1000, 2000, 3000, 4000
 90 GOTO 5
1000 PRINT "QUANTO E"" X " + " Y
1010 INPUT A
1020 IF A=X+Y THEN PRINT "CORRETO" ELSE
     PRINT "ERRADO"
1030 RETURN
2000 PRINT "QUANTO E"" X " - " Y
2010 INPUT A
2020 IF A=X-Y THEN PRINT "CORRETO" ELSE
     PRINT "ERRADO"
2030 RETURN
3000 PRINT "QUANTO E"" X "*" Y
3010 INPUT A
3020 IF A=X*Y THEN PRINT "CORRETO" ELSE
     PRINT "ERRADO"
3030 RETURN
4000 PRINT "QUANTO E"" X "/" Y
4010 INPUT A
4020 IF A=X/Y THEN PRINT "CORRETO" ELSE
     PRINT "ERRADO"
4030 RETURN
```

Notar a palavra ELSE nas linhas 1020, 2020, 3020, 4020. Você pode usar ELSE se quiser que o Computador faça alguma coisa de especial quando a condição não é verdadeira. Na linha 1020, se sua resposta A é igual a X + Y, o Computador imprime CORRETO, caso contrário (ELSE) imprime ERRADO.

Aqui está parte de um programa usando ON...GOTO:

```
   10 CLS
   20 PRINT@ 134, "(1) CANASTRA"
   30 PRINT@ 166, "(2) SETE E MEIO"
   40 PRINT@ 198, "(3) POQUER"
   50 PRINT@ 354, "O QUE VOCE QUER JOGAR?"
   60 INPUT A
   65 CLS
   70 ON A GOTO 1000, 2000, 3000
1000 PRINT@ 230, "JOGO DE CANASTRA"
1010 END
2000 PRINT@ 236, "JOGO SETE E MEIO"
2010 END
3000 PRINT@ 235, "JOGO DE POQUER"
3010 END
```

IMPONDO CONDIÇÕES

Qualquer um que conhece um pouco de inglês sabe a diferença entre AND e OR — até seu Computador. Por exemplo, vamos imaginar que a PROLÓGICA tenha uma vaga para um programador. Para se candidatar você deve ter:

```
UM DIPLOMA DE PROGRAMADOR
AND (e)
EXPERIÊNCIA EM PROGRAMAÇÃO
```

Aqui está o programa. Limpe a memória e digite:

```
10 PRINT "VOCE TEM..."
20 INPUT "UM DIPLOMA DE PROGRAMADOR"; D$
30 INPUT "EXPERIENCIA EM PROGRAMACAO"; E$
40 IF D$ = "SIM" AND E$ = "SIM" THEN PRINT "O EMPREGO
   E' SEU" ELSE PRINT "DESCULPE, NOS NAO PODEMOS
   CONTRATA-LO"
50 GOTO 10
```

Rode o programa. Com a sua experiência no seu Computador você deve responder às questões deste modo:

```
VOCE TEM...
UM DIPLOMA DE PROGRAMADOR? NAO
EXPERIENCIA EM PROGRAMACAO? SIM
DESCULPE, NOS.NAO PODEMOS CONTRATA-LO
```

Suponha agora que a PROLÓGICA decidiu ser menos exigente. Aqui estão as novas qualificações para o trabalho:

```
UM DIPLOMA DE PROGRAMADOR
OR (ou)
EXPERIENCIA EM PROGRAMACAO
```

Foi trocado AND por OR. Para fazer esta troca em seu programa digite:

```
40 IF D$ = "SIM" OR E$ = "SIM" THEN PRINT
   "O EMPREGO E' SEU"
   ELSE PRINT "DESCULPE, NOS NAO PODEMOS
   CONTRATA-LO"
```

Para ver a diferença que esta palavra faz, rode o programa:

```
VOCE TEM...
UM DIPLOMA DE PROGRAMADOR? NAO
EXPERIENCIA EM PROGRAMACAO? SIM
O EMPREGO E' SEU
```

Agora que você verificou que seu Computador entende a diferença entre AND e OR, você pode usá-las em seus programas. Nós usaremos estas palavras nos próximos capítulos.

MAIS AJUDA EM MATEMÁTICA

Há mais algumas palavras que você pode querer usar para ajudar em programas matemáticos.

SGN

SGN indica se um número é positivo, negativo ou nulo. Digite:

```
10 INPUT "DIGITE UM NUMERO"; X
20 IF SGN(X) = 1 THEN PRINT "POSITIVO"
30 IF SGN(X) = 0 THEN PRINT "ZERO"
40 IF SGN(X) = -1 THEN PRINT "NEGATIVO"
50 GOTO 10
```

Rode o programa. Verifique alguns números, tais como:

```
0 -12.4.5 -.32
```

ABS

ABS indica o valor absoluto do número (a magnitude do número sem considerar o seu sinal). Digite:

```
10 INPUT "DIGITE UM NUMERO"; N
20 PRINT "VALOR ABSOLUTO E'" ABS(N)
30 GOTO 10
```

Rode o programa e verifique os mesmos números acima.

STR$

STR$ converte um número em uma string. Exemplo:

```
10 INPUT "DIGITE UM NUMERO"; N
20 A$ = STR$(N)
30 PRINT A$ +  E' AGORA UMA STRING"
```

Mais alguma coisa antes de terminar este capítulo.

```
10 X = 1
20 PRINT X;
30 X = X*10
40 GOTO 20
```

Notar o erro OV (estouro) no final. O Computador não pode manipular números maiores que 1E+38 ou menores que −1E−38.

Algumas vezes o número será tão grande ou tão pequeno que o Computador só consegue imprimi-lo em "notação exponencial". O número 1 bilhão (1 000 000 000), por exemplo, pode ser escrito "1E+09". Isto significa "o número 1 seguido por nove zeros".

Ou, tecnicamente, $1*10^9$, que é 1 vezes 10 à nona potência:
*1*10*10*10*10*10*10*10*10*10*

Se temos uma resposta "5E−06", significa que devemos deslocar o ponto decimal, que vem depois do 5, 6 casas para a esquerda inserindo zeros onde for necessário. Tecnicamente, isto significa $5*10^{-6}$ ou 5 milionésimos (0.000 005).

Isto é 5/10/10/10/10/10

ASC e CHR$

Digite:

```
PRINT ASC("A") ENTER
```

e o Computador imprime:

```
65
```

65 é o código ASCII para o caractere A.
Digite:

```
PRINT CHR$(65) ENTER
```

e o Computador imprime:

```
A
```

o caractere que o número 65 representa é o A.
Padrão ASCII significa American Standard Code for Information Interchange (Código Padrão Americano para Intercâmbio de Informações). Pelo uso destes códigos seu Computador é capaz de comunicar-se por telefone com outros Computadores.
O Apêndice F dá a tabela de códigos ASCII dos caracteres. Se, por exemplo, quisermos utilizar a tecla ← para mover um ponto para trás, é necessário utilizar o seu valor ASCII.
Digite:

```
10 CLS6
20 H=62
30 RESET(H,15)
40 A$=INKEY$
50 IF A$=CHR$(8) THEN 70
60 GOTO 30
70 SET(H+1,15,6)
80 H=H−1
90 GOTO 30
```

Para avançarmos o ponto usaremos CHR$(9) que representa a tecla →, junto com umas pequenas modificações.

O COMPUTADOR TAGARELA

Quem disse que o Computador não pode falar? Sua voz, entretanto, soará estranha.
Nós conseguimos que o Computador fale usando a gravação de sua própria voz. Assim você aumentará o interesse e as brincadeiras em seus programas, particularmente os jogos e programas educativos. Mesmo que você não tenha um gravador, poderá ainda usar algumas idéias gráficas que nós damos neste capítulo.
Desconecte os 3 cabos de seu gravador para o Computador. Coloque uma fita, ponha no começo, pressione PLAY e RECORD, e grave algo. Diga qualquer coisa que você quiser.
Agora digite este programa:

```
 5 CLS
10 INPUT "PRESSIONE ⟨ENTER⟩ PARA OUVIR A GRAVACAO";
   A$
20 MOTOR ON
30 AUDIO ON
```

Se você não quiser gravar nada, pode tentar este programa usando uma fita de música ou uma com seus programas.

Pronto? Antes de executar o programa você precisa preparar sua fita para tocar:

- Retroceder a fita até o começa da gravação.
- Conectar seu gravador ao Computador.

O Capítulo 7 mostra como conectá-lo.

- Pressione PLAY no seu gravador.
- Aumente o volume de sua TV.

MOTOR ON liga seu gravador. AUDIO ON conecta o som de seu gravador ao da TV.

Adicione estas linhas:

```
35 CLS
40 A$=INKEY$
50 PRINT@225, "PRESSIONE X PARA DESLIGAR O GRAVADOR"
60 IF A$ <> "X" THEN 40
70 AUDIO OFF
80 MOTOR OFF
```

Prepare sua fita e rode o programa. A linha 40 diz ao Computador para rotular qualquer tecla que você pressionar como A$. Se você não está pressionando um "X", a linha 60 envia o programa de volta à linha 40. Se você pressionar X, a conexão de ÁUDIO e o MOTOR do gravador são desligados.

EXERCÍCIO DO CAPÍTULO

Fazer um programa que desenhe linhas na tela, usando como comandos as teclas ← quando quiser ir para a esquerda; → quando quiser ir para a direita; ↑ quando quiser subir; e ↓ quando quiser descer.

Inicie seu programa fixando um ponto.

CAPÍTULO 9

jogos de ação

Você está pronto para praticar tênis ou então tiro ao alvo? Você pode ensinar seu Computador a jogar qualquer jogo aprendendo mais uma palavra em BASIC. E esta palavra é POINT.
Limpe a memória e digite este programa:

```
 5 CLS(0)
10 FOR X=1 TO 5
20 SET (RND(64)-1, RND(30)+1,8)
30 NEXT X
40 FOR V=2 TO 31
50 FOR H=0 TO 63
60 IF POINT(H,V)< >0 THEN GOSUB 100
70 NEXT H, V
80 END
100 PRINT@0, "O PONTO" H "," V " ESTA' ACESO"
110 RETURN
```

Na linha 60 o Computador observa cada ponto POINT da localização vertical 2 até 31 e horizontal 0 até 63 para ver quais estão iluminados. Os que estiverem — isto é, o POINT não é igual a 0 — farão com que a linha 100 imprima suas localizações horizontal e vertical.

ASTERÓIDES E PLANETAS

Neste jogo, nós vamos usar o joystick direito, portanto ele deve estar conectado. Nós podemos criar asteróides colocando os pontos aleatoriamente na tela, como abaixo. Apague a memória e digite:

```
 5 CLS(0)
10 FOR X=1 TO 200
20 SET(RND(64)-1, RND(30)+1,8)
30 NEXT X
```

Marque o planeta que seu avião deve alcançar usando:

```
40 FOR H=54 TO 63
50 FOR V=28 TO 31
60 SET(H,V,3)
70 NEXT V, H
```

Para ler a posição do joystick direito, digite:

```
100 A=JOYSTK(0)
110 B=JOYSTK(1)
120 B=B/2
130 B=INT(B)
```

"A" lê a coordenada horizontal (0-63) e "B", a coordenada vertical (0-63). Como a maior posição vertical na tela é 31, nós temos que adicionar as linhas 120 e 130.
Para movimentar todo o bloco de acordo com a posição do joystick, adicione estas linhas:

```
200 IF INT(A/2)<A/2 THEN A=A-1
210 IF INT (B/2)<B/2 THEN B=B-1
220 FOR H=A TO A+1
230 FOR V=B TO B+1
240 SET (A,V,6)
250 NEXT V, H
999 GOTO 100
```

As linhas 200 e 210 garantem que os primeiros pontos vertical e horizontal são números pares e as linhas de 200 a 250 desenham o bloco. Rode o programa e experimente girar seu joystick. A linha de cor ciano se moverá para onde você deslocar o joystick. Agora vamos ao jogo. Digite:

```
212 FOR H=A TO A+1
214 FOR V=B TO B+1
216 IF POINT(H,V)=8 THEN SOUND 128,1:T=T+1
218 NEXT V, H
```

Rode novamente. Cada vez que você acerta um ponto laranja, o Computador soará um tom. Notar que a linha 216 faz duas coisas se o ponto for laranja:

- **Soa um tom;**
- **adiciona 1 a T, o contador.**

Adicione estas linhas ao seu programa:

```
235 IF POINT (H,V)=3 THEN PRINT@0,
    "CONGRATULACOES VOCE CONSEGUIU": END
300 PRINT@28, T
310 IF T >10 THEN 1000
1000 FOR X=1 TO 40
1010 CLS(RND(8))
1020 SOUND RND (255), RND (2)
1030 NEXT X
1040 PRINT@228, "SUA ESPACONAVE EXPLODIU"
```

Você gostaria de ter as orientações na tela? Adicione mais estas linhas:

```
  80 FOR X=1 TO 8
  82 READ A$
  84 PRINT@0, A$
  86 FOR Y=1 TO 1500 : NEXT Y
  88 NEXT X
  90 R$=INKEY$ : IF R$= "" THEN 90
  92 FOR H=4 TO 63
  94 SET(H,0,8):SET(H,1,8)
  96 NEXT H
2000 DATA SUA META E' CONSEGUIR
2010 DATA GUIAR SUA ESPACONAVE
2020 DATA ATRAVES DOS ASTEROIDES
2030 DATA PARA O PLANETA AZUL
2040 DATA ACERTANDO MAIS DE DEZ ASTEROIDES
2050 DATA SUA ESPACONAVE EXPLODIU!!!
2060 DATA PRESSIONE QUALQUER TECLA QUANDO
2070 DATA ESTIVER NO CANTO SUP ES
```

USANDO MOVIMENTOS RÁPIDOS

Vamos mostrar agora uma alternativa para o seu programa e achamos que você vai gostar. Digite:

```
PRINT CHR$(128) ENTER
```

O Computador imprime um bloco preto como este: ■

Teste mais alguns números. Digite:

```
PRINT CHR$(129) ENTER
PRINT CHR$(130) ENTER
PRINT CHR$(131) ENTER
```

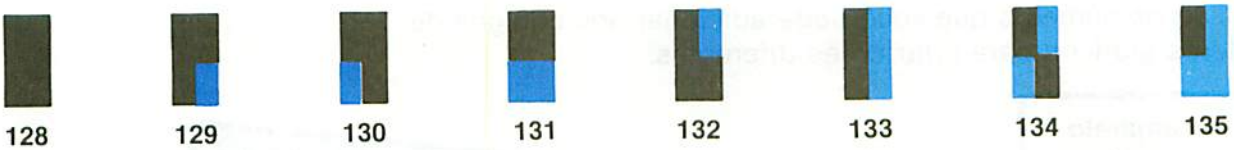
O Computador imprime 3 blocos com diferentes combinações de verde e preto:

Com fundo verde fica difícil ver o contorno dos blocos, digite então este programa. Ele imprimirá de novo o primeiro bloco em fundo cinza:

```
10 CLS(5)
20 PRINT@239, CHR$(129);
30 GOTO 30
```

O quadro de posições na tela para PRINT@ está no Apêndice E (nós explicamos como usá-lo no Capítulo 3). Não esqueça o ponto e vírgula do PRINT.

Lembra-se do CHR$ no capítulo anterior? CHR$ converte um código para o caractere que ele representa. Por exemplo, CHR$(65) converte o código 65 no caractere "A". Os códigos 128, 129, 130 e 131 são códigos para caracteres gráficos.
Veja "Localização Gráfica na Tela", no Apêndice E. Como explicamos anteriormente, as linhas escuras dividem a grade em blocos. Cada bloco contém 4 pontos. Esses pontos podem ser combinados de 16 maneiras diferentes para formar estes caracteres gráficos:



Para imprimir estes caracteres gráficos, digite e execute este programa:

```
10 CLS(5)
20 FOR C=128 TO 143
30 PRINT@0, "PRESSIONE QUALQUER TECLA PARA
   CONTINUAR"
40 PRINT@173, C;
50 PRINT@240, CHR$(C);
60 K$=INKEY$ : IF K$="" THEN 60
70 NEXT C
80 GOTO 10
```

Sabe por que é importante digitar um ponto e vírgula no fim destas linhas PRINT@? Tente com e sem ponto e vírgula.

O ponto e vírgula faz o Computador parar de imprimir assim que coloca seus caracteres na tela. Caso contrário, ele continua imprimindo com o seu costumeiro fundo verde o resto da linha.

A linha 50 imprime os caracteres gráficos para os códigos de 128 a 143 na localização 240 de sua tela. Tente algo um pouco diferente. Digite:

```
PRINT CHR$(129+16) ENTER
```

O Computador imprime o caractere gráfico para 129, exceto que a área que seria verde é amarela. Digite:

```
PRINT CHR$(129+32) ENTER
PRINT CHR$(129+48) ENTER
PRINT CHR$(129+64) ENTER
```

Estes são os números que você pode adicionar aos códigos de caracteres gráficos para criar cores diferentes:

16 — amarelo
32 — azul
48 — vermelho
64 — cinza
80 — ciano
96 — roxo
112 — laranja

*Notar que estes números são todos múltiplos de 16 (16=16*1; 32=16*2; 48=16*3; 112 = 16*7).*

Para ver todos os diferentes caracteres coloridos, adicione estas linhas e rode o programa:

```
10 CLS(1)
20 FOR X=0 TO 7
40 PRINT@170, C"+" X*16;
50 PRINT@240, CHR$(C+X*16);
75 NEXT X
```

Estes caracteres gráficos são "caracteres" como A, B, C e D. Portanto, você pode combinar e armazenar do mesmo modo que strings. Limpe a memória e digite:

```
20 A$=CHR$(135+16)+CHR$(143+16)+CHR$(131+16)
   +CHR$(131+16)
25 B$=CHR$(135+32)+CHR$(143+32)+CHR$(143+32)
30 C$=CHR$(135+96)+CHR$(139+96)
40 D$=CHR$(143+112)
```

E você pode posicioná-los no centro da tela do mesmo modo que posicionaria 3 palavras, usando PRINT@.

Você imprime caracteres gráficos usando PRINT@. Com a Localização Gráfica da Tela você coloca os pontos (SET).

Digite:

```
10 CLS(0)
50 PRINT@230,
   A$+CHR$(128)+B$+CHR$(128)+C$+CHR$(128)+D$
   +D$+D$;
60 GOTO 60
```

e rode o programa. O programa imprime a imagem de um caminhão amarelo, um caminhão azul, um carro roxo e um ônibus laranja. Limpe a memória e digite:

```
10 A=RND(7)*16 : B=RND(7)*16
20 A$=CHR$(129+A)+CHR$(131+A)
30 B$=CHR$(133+B)+CHR$(143+B)+CHR$(130+B)
40 PRINT B$, A$,: GOTO 10
```

Rode o programa.

O Computador vai criar aleatoriamente um carro e um caminhão coloridos.
Introduza essas linhas:

```
40 IF RND(2)=2 THEN VE$=A$ ELSE VE$=B$
50 PRINT VE$ : GOTO 10
```

Rode o programa. Algumas vezes você obterá um carro, outras um caminhão. O Computador cria aleatoriamente um carro ou um caminhão.

Agora você pode ver o trânsito. Digite:

```
 50 IF LEN(TR$+VE$+SP$)>32 THEN 100
 60 TR$=TR$+VE$
 70 GOTO 10
100 PRINT TR$
```

Rode o programa várias vezes. Cada vez o Computador cria aleatoriamente um tráfego de 32 caracteres de comprimento. Para fazê-lo mover-se, digite:

```
100 INPUT "VELOCIDADE (1-200)";S
110 FOR P=0 TO 480
120 PRINT@P, TR$;
130 FOR X=1 TO 200-S : NEXT X
140 CLS(0)
150 NEXT P
```

e rode. O tráfico se moverá do canto superior esquerdo ao canto inferior direito da tela. A linha 120 imprime o tráfego na localização P (0 até 480). A linha 130 coloca uma pausa no programa para a velocidade que você digitou. Para fazer o tráfego mover-se através de sua tela indefinidamente, digite:

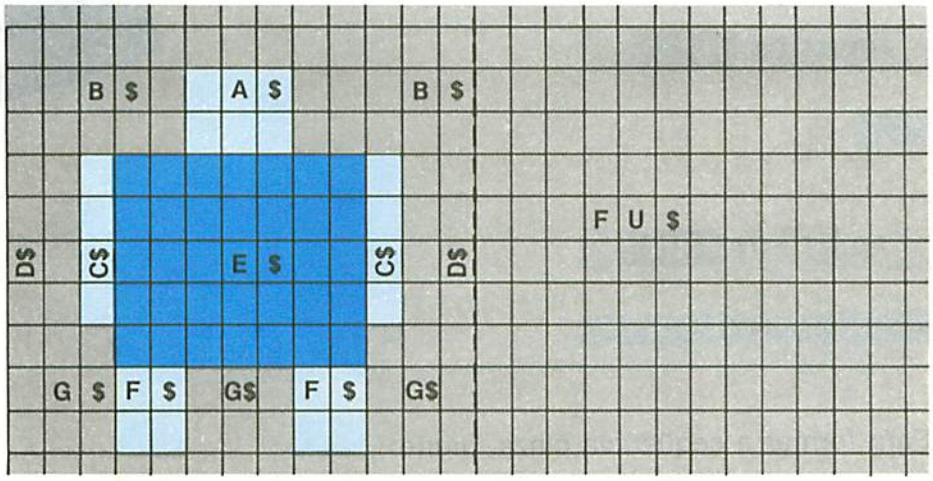
```
110 P=320
150 P=P+1
160 IF P=351 THEN 110
170 PRINT@P, LEFT$(TR$, 352-P);
180 PRINT@320, RIGHT$(TR$, P-320);
190 GOTO 130
```

Nós mostramos como usar LEFT$ e RIGHT$ no Capítulo 3.

e rode-o.

MÚSICA E MOVIMENTO

Aqui está a chance de rever o que você já aprendeu até agora. Nós não vamos ensinar nada novo. Vamos apenas fazer uma brincadeira construindo com strings gráficas um Computador dançarino.



Como isto tomará muito espaço para as strings, digite:

```
1 CLEAR 3000
```

para reservar bastante espaço.
Para formar strings feitas com caracteres gráficos pretos, digite:

```
10 D$ = CHR$(128) + CHR$(128)
20 G$ = D$ + CHR$(128)
30 B$ = G$ + D$
40 FU$ = B$ + B$ + B$ + D$ + D$
```

Na tela do Computador o azul-claro será cinza; o azul-escuro, vermelho; e a superfície cinza, preta.

Rode o programa e mande o Computador imprimir B$, D$, G$ e FU$

```
PRINT B$ ENTER
```

B$ tem cinco caracteres de comprimento.

```
PRINT D$ ENTER
```

D$ tem dois caracteres; G$ tem 3; FU$ tem 19 e A$ tem 3.

```
PRINT G$ ENTER
```

```
PRINT FU$ ENTER
```

Para formar a seqüência cinza, digite:

```
 50 C$ = CHR$(143 + 64)
 60 F$ = C$ + C$
 70 A$ = F$ + C$
 80 FOR T = 1 TO 7
 90 E$ = E$ + CHR$(143 + 48)
100 NEXT T
```

Rode o programa. Imprima A$, C$ e F$:

```
PRINT A$ ENTER
```

```
PRINT C$ ENTER
```

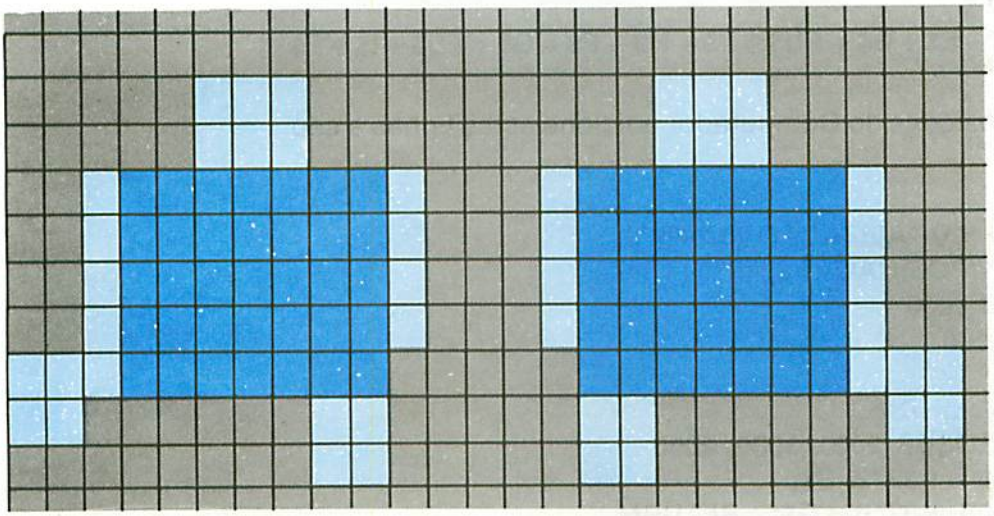
C$ tem um caractere de comprimento; F$ tem 2 e E$ tem 7.

```
PRINT F$ ENTER
```

Tente formar as seqüências para o corpo e pernas, sendo HD$ a cabeça; BD$, B1$ e B2$ corpo e braços; e L1$ as pernas. Aqui está o nosso programa:

```
110 HD$ = B$ + A$ + B$ + FU$ + B$ + A$ + B$ + FU$
115 J$ =  CHR$(128)
120 FOR X = 1 TO•4
130 BD$ = BD$ + D$ + C$ + E$ + C$ + D$ + FU$
140 NEXT X
145 B1$ = D$ + C$ + E$ + C$ + C$ + J$ + FU$
      + D$ + C$ + E$ + J$ + C$ + J$ + FU$
      + D$ + C$ + E$ + J$ + C$ + J$ + FU$
      + D$ + C$ + E$ + G$
147 B2$ = J$ + C$ + C$ + E$ + C$ + D$ + FU$
      + J$ + C$ + J$ + E$ + C$ + D$ + FU$
      + J$ + C$ + J$ + E$ + C$ + D$ + FU$
      + G$ + E$ + C$ + D$
150 L1$ = G$ + E$ + G$ + FU$ + G$ + F$ + G$ + F$ + G$ + FU$
      + G$ + F$ + G$ + F$ + G$
```

Para fazer o Computador dançar, nós daremos a ele mais duas posições para as pernas.



Adicione linhas ao seu programa para criar as seqüências L2$ e L3$.

FAÇA VOCÊ MESMO

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Nós fizemos deste jeito:

```
160 H$ = G$ + G$
170 I$ = H$ + D$
180 L2$ = G$ + E$ + A$ + FU$ + G$ + F$ + H$ + F$ + FU$
    + G$ + F$
190 L3$ = A$ + E$ + G$ + FU$ + F$ + H$ + F$ + G$ + FU$ + I$ + F$
```

Para ver as 3 posições do Computador adicione estas linhas a seu programa:

```
 500 INPUT "LOCALIZACAO (0-243)";L
 510 INPUT "POSICAO (1-3)";P
 520 GOSUB 1000
 530 GOTO 500
1000 CLS(0)
1010 PRINT@L,HD$;
1020 ON P GOSUB 2000, 3000, 4000
1025 PRINT@L + 32*2, B$;
1030 PRINT@L + 32*6, LG$; : RETURN
2000 LG$ = L1$ : B$ = BD$:RETURN
3000 LG$ = L2$ : B$ = B1$:RETURN
4000 LG$ = L3$ : B$ = B2$:RETURN
```

Rode o programa. Tente diferentes posições.
A linha 1010 imprime a cabeça na localização por você escolhida.

A linha 1020 envia o programa para uma sub-rotina que faz LG$ igual a L1$, L2$ ou L3$ e B$ igual a BD$, B1$ ou B2$ (dependendo do que você digitou). A linha 1025 imprime B$ e a linha 1030 imprime LG$. B$ será impresso duas colunas abaixo de HD$ e LG$ seis colunas abaixo de HD$ que está na posição por você escolhida. Pelo controle destas posições, você pode facilmente fazer o Computador se mover. Eis aqui como nós fizemos. Mude as linhas 500 e 510 e adicione estas:

```
495 INPUT "VELOCIDADE (1-10)";V
500 FOR X=1 TO 17
510 IF X=1 OR X=5 THEN RESTORE
515 READ L, P, T, D
525 SOUND T, V*D
527 NEXT X
5000 DATA 137, 2, 89, 1, 240, 1, 133, 2
5010 DATA 137, 3, 159, 1, 229, 1, 133, 2
5020 DATA 5, 1, 89, 1, 229, 1, 133, 2
5030 DATA 5, 1, 147, 1, 229, 1, 159, 1
5040 DATA 229, 1, 147, 1, 5, 1, 133, 1
5050 DATA 229, 1, 125, 2, 5, 1, 133, 1
5060 DATA 229, 1, 147, 2
```

Lembre-se de READ e DATA do Capítulo 5.

Rode o programa e assista-o dançar.
A linha 515 lê local, posição, tom e duração do som das linhas 5000 até 5060. À primeira passada do programa, o Computador aparecerá na localização 137, posição 2 e na linha 525, soará o tom 89 com a duração de V vezes 1. À segunda passada, o Computador aparecerá na localização 240, posição 1, e soará o tom 133 para uma duração de V vezes 2.
Como você pode ver, adicionando mais posições, pode-se deixar isto mais divertido. Use tudo que você sabe.

EXERCÍCIO DO CAPÍTULO

Tente fazer o nosso dançarino começar bem lentamente. À medida que a música toca, ele ganha velocidade, até atingir o máximo possível.

CAPÍTULO

10

um grande administrador

Você tem uma lista ou arquivo que quer que o Computador controle? Aqui, você faz com que o Computador classifique, compare, armazene e imprima sua informação mais veloz e corretamente do que você poderia fazer.
O Computador pode controlar qualquer coisa que você pense tais como: compras, imposto de renda, agenda de endereço e telefone, livros, discos, documentos, jogos e mais, muito mais...

CÁLCULO POR TABELA

Você já tentou escrever um programa para manipular informações? Pois o seu Computador foi construído para "rotular e organizar" sistemas, fazendo esta programação muito fácil.
Para começar, como conseguir que o Computador lembre as notas de provas de 15 alunos.

ALUNO	NOTA
1	94
2	20
3	10
4	63
5	45
6	35
7	70
8	100
9	70
10	70
11	20
12	10
13	55
14	90
15	85

Para lembrar o Computador sempre usa variáveis. Para conseguir que o Computador se lembre das notas das provas dos alunos, digite:

Veja o Capítulo 2 para revisar variáveis.

```
A = 94 ENTER
B = 20 ENTER
C = 10 ENTER
```

Podemos melhorar ainda mais. Digite deste modo:

```
A(1)=94 ENTER
A(2)=20 ENTER
A(3)=10 ENTER
```

Cada uma destas variáveis tem um rótulo A(1), A(2) e A(3). Apesar de serem diferentes, elas funcionam do mesmo jeito que as anteriores. Para ver isso, digite estas duas linhas:

```
PRINT A;B; C ENTER
PRINT A(1); A(2); A(3) ENTER
```

Quando usamos uma só variável para vários valores ao mesmo tempo chamamos isso de arranjo. Cada valor é um item do arranjo.

As duas linhas que você digitou fazem o mesmo, certo? Então, por que os arranjos são melhores? Dê uma olhada nestes dois programas. Ambos fazem o mesmo:

Programa 01

```
10 DATA 94, 20, 10, 63, 45
20 DATA 35, 70, 100, 70, 70
30 DATA 20, 10, 55, 90, 85
40 READ A, B, C, D, E
50 READ F, G, H, I, J
60 READ K, L, M, N, O
70 INPUT "NUMERO DO ALUNO (1-15)"; Z
75 IF Z>15 THEN 70
80 IF Z=1 THEN PRINT "NOTA" A
90 IF Z=2 THEN PRINT "NOTA" B
100 IF Z=3 THEN PRINT "NOTA" C
110 IF Z=4 THEN PRINT "NOTA" D
120 IF Z=5 THEN PRINT "NOTA" E
130 IF Z=6 THEN PRINT "NOTA" F
140 IF Z=7 THEN PRINT "NOTA" G
150 IF Z=8 THEN PRINT "NOTA" H
160 IF Z=9 THEN PRINT "NOTA" I
170 IF Z=10 THEN PRINT "NOTA" J
180 IF Z=11 THEN PRINT "NOTA" K
190 IF Z=12 THEN PRINT "NOTA" L
200 IF Z=13 THEN PRINT "NOTA" M
210 IF Z=14 THEN PRINT "NOTA" N
220 IF Z=15 THEN PRINT "NOTA" O
230 GOTO 70
```

Programa 02

```
 10 DATA 94, 20, 10, 63, 45
 20 DATA 35, 70, 100, 70, 70
 30 DATA 20, 10, 55, 90, 85
 40 DIM A(15)
 50 FOR X=1 TO 15
 60 READ A(X)
 70 NEXT X
 80 INPUT "NUMERO DO ALUNO (1-15)"; Z
 85 IF Z>15 THEN 80
 90 PRINT "NOTA"A(Z)
100 GOTO 80
```

Na verdade, é deixado espaço para 16 itens pois você pode usar o 0 como índice (o número entre parênteses).

O primeiro programa usa variáveis normais. O segundo usa arranjos. Com arranjos fica mais fácil programar uma longa lista de informações. Introduza e rode o segundo programa. Eis como ele funciona:

A linha 40 prepara uma lista de informações, uma matriz (arranjo), chamada A, com 15 itens distintos.

As linhas 50 e 70 produzem um ciclo para contar de 1 a 15. A linha 60 põe o seguinte na memória:

Memória do seu Computador

A(1) = 94	A(8) = 100	A(15) = 85
A(2) = 20	A(9) = 70	
A(3) = 10	A(10) = 70	
A(4) = 63	A(11) = 20	
A(5) = 45	A(12) = 10	
A(6) = 35	A(13) = 55	
A(7) = 70	A(14) = 90	

Isto armazena todas as notas na matriz chamada A. Esta matriz contém 15 itens indexados (identificados por um índice).

A linha 80 pede a você para introduzir um dos índices e a linha 90 imprime o que você requisitou.

Agora que você tem as notas armazenadas em uma matriz, fica fácil trabalhar com elas.

Vamos supor que a estas notas os alunos ainda podem acrescentar mais alguns pontos de conceito. Digite estas linhas:

```
92 INPUT "VAI SER ADICIONADO NOTA DE CONCEITO"; R$
94 IF R$="NAO"THEN 80
96 INPUT "QUANTOS PONTOS A MAIS"; X
97 A(Z)=A(Z)+ X
98 PRINT "O ALUNO" Z "TEM AGORA NOTA" A(Z)
```

Se você quer imprimir uma tabela com todas as notas, adicione estas linhas:

```
 72 INPUT "VOCE QUER VER TODAS AS NOTAS"; S$
 74 IF S$="SIM" THEN GOSUB 110
110 PRINT"ALUNO","NOTAS","CONCEITO"
120 FOR X=1 TO 15
130 PRINT X, A(X),
140 NEXT X
150 RETURN
```

e mude a linha 100:

```
100 GOTO 72
```

ALUNO	NOTA	CONCEITO
1	94	6
2	20	5
3	10	4
4	63	4
5	45	3
6	35	5
7	70	6
8	100	6
9	70	4
10	70	5
11	20	3
12	10	4
13	55	4
14	90	5
15	85	2

Nós podemos usar outra matriz no nosso programa para o conceito. Este programa grava os pontos das notas (Matriz A) e de conceito (Matriz B).

Dados para a Matriz A

Dados para a Matriz B

Lê dados da Matriz A

Lê dados da Matriz B

```
10 DATA 94, 20, 10, 63, 45
20 DATA 35, 70, 100, 70, 70
30 DATA 20, 10, 55, 90, 85
40 DATA 6, 5, 4, 4, 3
50 DATA 5, 6, 6, 4, 5
60 DATA 3, 4, 4, 5, 2
70 DIM A(15), B(15)
80 FOR X=1 TO 15
90 READ A(X)
100 NEXT X
110 FOR X=1 TO 15
120 READ B(X)
130 NEXT X
```

```
140 INPUT "ALUNO N."; Z
145 IF Z>15 THEN 140
150 INPUT "NOTA<1> OU CONCEITO<2>"
160 IF R$="1" THEN PRINT A(Z)
170 IF R$="2" THEN PRINT B(Z)
180 GOTO 140
```

TESTE DE MEMÓRIA

Este programa testa sua memória e a memória do seu Computador. Digite NEW para apagar o programa anterior e digite:

```
1 DIM A(5)
```

```
  5 CLS
 10 PRINT@165, "MEMORIZE ESTES NUMEROS"
 15 PRINT@230 "VOCE TEM 7 SEGUNDOS"
 20 FOR X=1 TO 460*2: NEXT X
 25 CLS
 30 FOR X=1 TO 5
 35 A(X)=RND(100)
 40 NEXT X
 45 PRINT@71,A(1)
 50 PRINT@88,A(2)
 55 PRINT@238,A(3)
 60 PRINT@391,A(4)
 65 PRINT@408,A(5)
 70 FOR Y=1 TO 460*7:NEXT Y
 75 CLS
 80 PRINT@8, "QUAIS SAO OS NUMEROS?"
 85 PRINT@71,;:INPUT R(1)
 90 PRINT@88,;:INPUT R(2)
 95 PRINT@238,;:INPUT R(3)
100 PRINT@391,;:INPUT R(4)
105 PRINT@408,;:INPUT R(5)
110 FOR V=1 TO 460*2:NEXT V
115 IF R(1)=A(1) THEN PRINT@100,"CORRETO" ELSE
    PRINT@96,"O CERTO E'"A(1)
120 IF R(2)=A(2) THEN PRINT@117, "CORRETO" ELSE
    PRINT@115,"O CERTO E'" A(2)
125 IF R(3)=A(3) THEN PRINT@268, "CORRETO" ELSE
    PRINT@265,"O CERTO E'" A(3)
130 IF R(4)=A(4) THEN PRINT@420, "CORRETO" ELSE
    PRINT@416, "O CERTO E'" A(4)
135 IF R(5)=A(5) THEN PRINT@437, "CORRETO" ELSE
    PRINT@435, "O CERTO E'" A(5)
```

Na verdade, você não precisa desta linha DIM, se nenhum dos seus itens da matriz usar um índice maior que 10. Entretanto, é uma boa idéia colocar esta linha em seu programa, para reservar a quantidade necessária da memória.

Lembre-se de que você pode colocar várias instruções em uma linha, separando-as por dois pontos.

A linha 1 reserva espaço para uma matriz chamada:
A com 5 itens.

Da linha 30 à 40 são designados 5 números aleatórios para a matriz.
Da linha 45 à 65 estes números são impressos na tela.
Da linha 80 à 105 o Computador pergunta a você quais são os números.
Da linha 115 à 130 ele compara com os números corretos e dá a resposta.

Você pode aumentar a dificuldade deste jogo aumentando o argumento de RND.

NORMAS PARA MATRIZ

1. Há dois tipos de variáveis:
   - **A.** Variáveis simples, tais como A, B, C e D.
   - **B.** Variáveis "rotuladas" ou itens de matriz, tais como A(5), A(3), B(2) e B(6).
2. Uma matriz é um grupo de itens rotulados e cada um tem o mesmo nome de variável. Por exemplo, m(2), m(4), m(5), m(6), todos pertencem a uma mesma matriz chamada m.

DÊ UMA FORÇA NA SUA GRAVAÇÃO

No item anterior nós somente usamos matriz para listas de números. Mas a matriz é também usada para palavras. Não exatamente uma simples lista de palavras, mas como você verá neste capítulo seu Computador saberá editar e imprimir um texto cheio de palavras.
Iniciaremos com uma coisa bem simples — uma lista de compras:

1 — ARROZ	7 — PEIXE
2 — FEIJÃO	8 — SAL
3 — BATATAS	9 — AÇÚCAR
4 — TOMATES	10 — PÃO
5 — OVOS	11 — LEITE
6 — ALFACE	12 — QUEIJO

Para fazer com que o Computador se lembre de cada um desses itens, nós designaremos cada um a uma variável de string indexada (arranjo string). Para os três primeiros itens, você poderia digitar:

```
S$(1) = "ARROZ" ENTER
S$(2) = "FEIJAO" ENTER
S$(3) = "BATATAS" ENTER
```

Não esqueça o sinal $. É toda a diferença entre estas variáveis e as do último capítulo.

Para que o Computador imprima estes três primeiros itens, digite:

```
PRINT S$(1), S$(2), S$(3) ENTER
```

Aqui está como colocá-los no programa:

```
  5 DIM S$(12)
 10 DATA ARROZ, FEIJAO, BATATAS, TOMATES
 20 DATA OVOS, ALFACE, PEIXE, SAL
 30 DATA ACUCAR, PAO, LEITE, QUEIJO
 40 FOR X=1 TO 12
 50 READ S$(X)
 60 NEXT X
 70 PRINT "LISTA DE COMPRAS"
 80 FOR X=1 TO 12
 90 PRINT X ; S$(X)
100 NEXT X
```

Lê itens na matriz S$

Imprime a matriz S$

Este programa coloca todos os 12 itens em uma matriz (arranjo) chamada S$ e imprime a lista. Vamos acrescentar algumas linhas neste programa para que você possa mudar qualquer um destes itens nesta lista.

```
110 INPUT "QUER MUDAR ALGUM ITEM?";R$
120 IF R$< >"SIM" THEN 200
130 INPUT "DIGITE O NUMERO DO ITEM"; N
140 IF N>12 THEN 130
150 INPUT "QUAL E' O ITEM SUBSTITUTO"; S$ (N)
160 GOTO 70
200 END
```

ESCREVENDO UM ENSAIO

Aqui está um programa melhor para ajudá-lo nos seus escritos. Use-o junto com o que aprendeu no Capítulo 5 e você terá um programa processador de textos:

```
  1 CLEAR 1000
  5 DIM A$(50)
 10 PRINT "DIGITE UM TEXTO"
 20 PRINT "PRESSIONE / QUANDO TERMINAR"
 30 X=1
 40 A$=INKEY$
 50 IF A$="" THEN 40
 60 PRINT A$;
 70 IF A$="/" THEN 110
 80 A$(X)=A$(X)+ A$
 90 IF A$="." THEN X=X+1
100 GOTO 40
110 CLS
120 PRINT "SEUS PARAGRAFOS"
130 PRINT
140 FOR Y=1 TO X
150 PRINT A$(Y);
160 NEXT Y
```

Digite e rode o programa. Antes de ver como ele funciona, faça algumas experiências com ele. Digite:

```
PRINT A$(1) ENTER
PRINT A$(2) ENTER
PRINT A$(3) ENTER
```

Teve uma idéia de como funciona? Aqui está sua execução:

- A linha 1 limpa uma grande quantidade de espaços para o Computador usar.
- A linha 5 reserva espaço para uma matriz chamada A$ que pode ter até 50 parágrafos.
- A linha 30 faz X igual a 1. X será usado para rotular todos os parágrafos.
- A linha 40 verifica que tecla você está pressionando. Se não estiver pressionando nenhuma, a linha 50 envia o Computador de volta à linha 40.
- A linha 60 imprime a tecla que você pressionou.
- A linha 70 envia o Computador para a linha que imprime seu texto quando você pressionar a tecla "/".
- A linha 80 constrói cada período e o identifica com o número X. X é igual a 1 até você pressionar o ponto final. Então a linha 80 faz X igual a X+1.

Por exemplo, se a primeira letra que você pressionar for "P": A$(1) será igual a "P".
Se a segunda letra que você pressionar for "R":

A$(1) será igual a A$(1) + "R", que é o "P" + "R" ou "PR".
Isto pode continuar até que A$(1) seja igual a "PRIMEIRO PARAGRAFO". Aí você pressiona ".". A próxima letra que pressionar estará junto com A$(2). As linhas 140 até 160 imprimem seus parágrafos.

PROCESSADOR DE TEXTOS

Aqui está um programa para aqueles envolvidos com o processamento de textos. Faça um programa processador de textos completo que:
1. Imprima qualquer sentença que você quiser.
2. Deixe você corrigir sua sentença.
Para ajudá-lo, ver o programa que nós mostramos no final do Capítulo 5.

CÓPIAS IMPRESSAS

Se você tem uma impressora, então deve já estar pronto para colocá-la em uso.
Conectar o cabo da impressora ao conector marcado "RS" na traseira do seu teclado. Ligá-la e colocar o papel . O manual que vem com a impressora mostra como fazê-lo.
Pronto? Digite este pequeno programa:

```
10 INPUT A$
20 PRINT#-2, A$
```

Agora digite:

```
LLIST ENTER
```

e veja a impressora trabalhar. Ela fará facilmente a listagem dos programas longos. Se seu programa não lista na impressora, verifique se ela está ligada, preparada e conectada ao seu Computador. Teste, digitando LLIST novamente.
Agora rode o programa. Introduza tudo que você quiser e assista a impressora trabalhar.
PRINT#-2 manda o Computador imprimir, não na tela mas no dispositivo#-2, que é a impressora.
Pressione SHIFT e 0 (zero) simultaneamente para que as letras que você digitar apareçam de cores trocadas na tela (verde com fundo preto). Você terá agora na impressora letras maiúsculas/minúsculas. As letras de cores trocadas na tela serão as minúsculas na impressora.
Digite uma letra enquanto estiver com SHIFT apertado. Ela será uma letra maiúscula e vai aparecer em cor normal.
Rode seu programa, pressionando SHIFT enquanto você digita RUN:

```
RUN ENTER
```

Introduza uma sentença com ambos os casos, letras maiúsculas e minúsculas. Digite:

```
MINHA IMPRESSORA IMPRIME MINUSCULAS
```

Uma impressora com letras maiúsculas e minúsculas seria ótimo para um programa processador de textos.
Veja aquele que discutimos anteriormente neste capítulo. Como você muda as linhas 140-160 para que seu parágrafo seja impresso na impressora ao invés de na tela?
Pronto? Você simplesmente mudaria a linha 150 para:

```
150 PRINT#-2, A$(Y);
```

Todas as letras em RUN devem aparecer em cores normais (não trocadas).

GRAVE SUA COLEÇÃO DE LIVROS

Você sabe que pode armazenar programas em fita. Mas pode também usar a fita para armazenar qualquer lista que você queira. Cada vez que você tiver uma lista na fita você pode economizar tempo usando a força do Computador para imprimi-la, mudá-la, aumentá-la ou analisá-la a qualquer hora.
Pronto para começar a se organizar? Vamos começar com seus livros. Aqui estão alguns de sua biblioteca.

1. O Nome da Rosa — Umberto Eco
2. Eu, Robô — Isaac Asimov
3. Não — Celso Furtado
4. Vidas Secas — Graciliano Ramos
5. Incidente em Antares — Érico Veríssimo

Para colocar esta lista na fita e lê-la de volta na memória de seu Computador, você precisa de um programa. Você tem que digitá-lo todo antes de ver como ele funciona. Portanto, paciência. Comece a digitar:

```
10 OPEN "O",#-1, "LIVROS"
```

O "0" indica SAÍDA (output, em inglês).

Isto diz ao Computador para abrir as linhas de comunicação com o dispositivo #-1, o gravador. Nós estaremos enviando (saindo) um arquivo de informação e armazenando-o todo sob o nome LIVROS.

Um arquivo é um conjunto de informações — tais como livros — armazenadas sob um mesmo nome.

Agora digite:

```
15 CLS: PRINT "INTRODUZA SEUS LIVROS — DIGITE
   <XX> QUANDO TIVER TERMINADO"
20 INPUT "TITULO"; T$
30 PRINT#-1, T$
40 GOTO 15
```

Isto permite que você introduza em T$ o título do livro. Cada vez que você introduzir em T$, o Computador imprime-o — não na tela, mas no dispositivo #-1, que é o gravador.
Acrescente estas linhas:

```
25 IF T$="XX" THEN 50
50 CLOSE#-1
```

Isto permite digitar XX quando você tiver terminado de digitar todos os títulos de seus livros. O Computador então fecha a comunicação com o dispositivo #-1, o gravador de fita.
Coloque mais estas 3 linhas:

```
1 CLS
2 PRINT "COLOQUE A FITA E PRESSIONE PLAY/RECORD"
4 INPUT "PRESSIONE <ENTER> QUANDO PRONTO"; R$
```

Este é o programa. Antes de rodá-lo, você precisa:
- Conectar seu gravador. O Capítulo 7 mostra como.
- Colocar uma fita no gravador e retrocedê-la até o começo.
- Pressionar RECORD e PLAY juntas.

Pronto? Liste seu programa para ver se ele ainda parece com o nosso:

```
 1 CLS
 2 PRINT "COLOQUE A FITA E PRESSIONE PLAY/RECORD"
 4 INPUT "PRESSIONE <ENTER> QUANDO PRONTO"; R$
10 OPEN "O",#-1, "LIVROS" → (abre a linha com o gravador)
15 CLS : PRINT "DIGITE SEUS LIVROS OU XX NO FINAL"
20 INPUT "TITULO"; T$
25 IF T$ = "XX" THEN 50
30 PRINT #-1, T$
40 GOTO 15
50 CLOSE #-1
```

Imprimirá título na fita

Desliga a linha com o gravador

Já está digitado? Então rode.
Note que assim que você pressiona ENTER, o motor do gravador liga. O Computador está abrindo um "arquivo" na fita e chamando-o "LIVROS".
Quando ele pedir por títulos, introduza os 5 títulos a seguir e então digite XX:

```
TITULO?— O NOME DA ROSA
TITULO?— EU, ROBO
TITULO?— NAO
TITULO?— VIDAS SECAS
TITULO?— INCIDENTE EM ANTARES
TITULO?— XX
```

O Computador limpará a tela depois de cada título.

Cada vez que você introduzir um título, o Computador o imprime em um lugar especial na memória reservado ao gravador de fita. Quando você tiver terminado, o motor do gravador andará novamente. O Computador está gravando todos os títulos na fita (linha 30) e então fechando a comunicação com o gravador (linha 50). Agora todos os títulos estão na fita. Para carregá-los de volta digite:

```
60 CLS : PRINT "REBOBINE A FITA E PRESSIONE PLAY"
70 INPUT "PRESSIONE <ENTER> QUANDO PRONTO"; R$
80 OPEN "I",#-1, "LIVROS"
```

O "I" indica ENTRADA (input, em inglês).

A linha 80 abre a comunicação com o gravador para um arquivo de informação chamado "LIVROS". Desta vez, ao invés de estar aberta para a saída, a comunicação está aberta para a entrada no Computador. Acrescente estas linhas:

```
 90 INPUT#-1, T$
100 PRINT T$
```

A linha 90 pega o primeiro título — T$. Novamente, T$ não está entrando pelo teclado. Ela está vindo do gravador. A linha 100 imprime T$ em sua tela. Agora, adicione estas linhas:

```
 85 IF EOF(-1) THEN 120
110 GOTO 85
120 CLOSE#-1
```

A linha 85 diz que se você estiver no fim do arquivo (End-Of File) dos livros, então vai para a linha 120, que fecha a comunicação com o gravador.

Você está curioso em saber o que o -1 significa? A função EOF resulta em -1 se você alcançou o final do arquivo.

O certo é colocar a linha EOF(-1) antes da linha INPUT#-1. De outro modo, você obterá o IE ERRO que indica entrada depois do final do arquivo. Liste esta última parte do programa, digitando:

```
LIST 60 -
```

Deverá aparecer isto:

```
 60 CLS : PRINT "REBOBINE A FITA E PRESSIONE PLAY"
 70 INPUT "PRESSIONE <ENTER> QUANDO PRONTO"; R$
 80 OPEN "I",#-1, "LIVROS"
 85 IF EOF(-1) THEN 120
 90 INPUT#-1, T$
100 PRINT T$
110 GOTO 85
120 CLOSE#-1
```

Abre a linha para o gravador

Imprime títulos na fita

Fecha linhas para o gravador

Não pressione RECORD — apenas PLAY. Além disso, não esqueça de rebobinar a fita.

Agora rode-o. Digite:

```
RUN 60
```

Se o seu Computador suspender a comunicação com o gravador, você pode recuperar o controle pressionando as duas teclas RESET ao mesmo tempo. Depois então verifique em quais linhas aparecem erros.

Quando pressionar ENTER, note que o motor do gravador começa a rodar. O Computador pega seus itens da fita. Assim que eles entrarem, o Computador vai imprimir os 4 itens na sua tela.

```
 1 CLS
 2 PRINT "COLOQUE A FITA E PRESSIONE PLAY/RECORD"
 4 INPUT "PRESSIONE <ENTER> QUANDO PRONTO"; R$
10 OPEN "O",#-1, "LIVROS"
15 CLS : PRINT "DIGITE SEUS LIVROS OU XX NO FINAL"
20 INPUT "TITULO"; T$
25 IF T$="XX" THEN 50
30 PRINT#-1, T$
40 GOTO 15
50 CLOSE#-1
60 CLS : PRINT "REBOBINE A FITA E PRESSIONE PLAY"
70 INPUT "PRESSIONE <ENTER> QUANDO PRONTO"; R$
80 OPEN "I",#-1, "LIVROS"
85 IF EOF(-1) THEN 120
90 INPUT#-1, T$
100 PRINT T$
110 GOTO 85
120 CLOSE#-1
```

Abre a linha para o gravador

Imprime títulos na fita

Fecha linhas para gravador

Entra títulos da fita

Na linha 30 nós gravamos T$ (o título dos livros) na fita. Para fazer isto, nós temos que abrir comunicação com o gravador para saída. Depois de terminar a "saída", temos que fechar a comunicação com o gravador.
Na linha 90 nós pegamos T$ do gravador. Para fazer isto, nós temos que abrir comunicação com o gravador para entrada. Depois de terminada a "entrada" fechamos a comunicação.
Entendeu? Pense nas respostas destas três questões...

QUESTÕES

1 — O que aconteceria se você tirasse a linha 50 e rodasse o programa?
R — Sem a linha 50, a comunicação com o gravador permanece aberta (OPEN) para a saída (OUTPUT). Como já está aberto, o Computador não deixará você abri-lo novamente para a entrada (INPUT). Por isso a linha 80 força um "AO ERRO" — tentativa de abrir um arquivo que já foi aberto.
2 — Teria problema se tirássemos as linhas 50 e 80 e rodássemos o programa?
R — Sem as linhas 50 e 80, a comunicação permanece aberta para a saída. Quando a linha 90 pedir ao Computador uma entrada do gravador, você obterá um "NO ERRO" — o arquivo não está aberto (OPEN) para entrada (INPUT).
3 — Funcionaria se você mudasse as linhas 90 e 100 como abaixo?

```
90 INPUT#-1, X$
100 PRINT X$
```

R — Sim, funciona. O Computador não leva em consideração que você chamou o título de T$ quando os colocou na fita. Quando você os pede novamente, o Computador simplesmente olha para uma variável string na fita e a coloca em X$.

UM CATÁLOGO ELETRÔNICO

Como mudar o programa para que você possa colocar todos estes itens na fita:

TÍTULO	AUTOR	ASSUNTO
O Nome da Rosa	Umberto Eco	Romance
Eu, Robô	Isaac Asimov	Ficção científica
Não	Celso Furtado	Economia
Vidas Secas	Graciliano Ramos	Romance
Incidente em Antares	Érico Veríssimo	Romance

Primeiro trabalharemos com a primeira metade do programa — a parte que sai da fita.
Acrescente estas linhas:

```
26 INPUT "AUTOR"; A$
28 INPUT "ASSUNTO"; S$
29 IF A$="XX"OR S$="XX" THEN 50
```

Para gravar todas na fita, simplesmente mude a linha 30:

```
30 PRINT#-1, T$, A$, S$
```

Agora, a segunda metade do programa. Como você pode mudar as linhas de 90 a 100 para que o Computador entre na fita e grave o título, autor e assunto? Digite:

```
 90 INPUT#-1, T$,A$,S$
100 PRINT "TITULO:" T$
102 PRINT "AUTOR:" A$
104 PRINT "ASSUNTO:" S$
```

Como dissemos anteriormente, você não precisa usar os mesmos nomes de variáveis. Assim também funcionaria:

```
 90 INPUT#-1, X$,Y$, Z$
100 PRINT "TITULO:" X$
102 PRINT "AUTOR:" Y$
104 PRINT "ASSUNTO:" Z$
```

CLASSIFICAÇÃO

Agora, você pode ter vantagens com toda esta organização. Por exemplo: você pode querer que o Computador imprima uma lista de livros pelo assunto. Adicione estas linhas a seu programa:

```
130 CLS
140 INPUT "QUAL ASSUNTO"; C$
150 PRINT "RETROCEDA A FITA - PRESSIONE PLAY"
160 INPUT "PRESSIONE <ENTER> QUANDO PRONTO"; E$
170 CLS : PRINT C$ "LIVROS": PRINT
180 OPEN "I",#-1, "LIVROS"
190 IF EOF(-1) THEN 230
200 INPUT#-1, T$, A$, S$
210 IF S$ = C$ THEN PRINT T$, A$
220 GOTO 190
230 CLOSE#-1
```

Rode o programa digitando RUN 130. Se você pedir ficção aparecerá:

```
QUAL O ASSUNTO: FICCAO CIENTIFICA
RETROCEDER A FITA - PRESSIONE PLAY
PRESSIONE <ENTER> QUANDO PRONTO
LIVROS DE FICCAO CIENTIFICA
EU, ROBO   ISAAC ASIMOV
```

CONTROLANDO O TALÃO DE CHEQUES

Vamos dizer que você tem a seguinte relação de cheques:

Nº	DATA	PAGO A	CONTA	QUANTIA
101	13/05	BOM PÃO	ALIMENTO	Cr$ 520,00
102	13/05	BELOCAR	CARRO	Cr$ 8.400,00
103	14/05	A CHAPA	ALIMENTO	Cr$ 2.500,00
104	17/05	VIAÇÃO RAIO	FÉRIAS	Cr$ 25.870,00
105	19/05	HOTEL MARAVILHA	FÉRIAS	Cr$ 19.000,00

Vamos escrever um programa para controlar esses cheques. O programa deve permitir a introdução dos dados, gravá-los em fita, recuperá-los mais tarde e permitir que você verifique uma determinada conta, como alimentos, por exemplo. Você ainda se lembra como gravar informação na fita?

NORMAS PARA MANDAR DADOS PARA A FITA

Para gravar dados na fita você deve:

1. Abrir comunicação com a fita para saída;
2. Imprimir dados na fita;
3. Continuar imprimindo dados na fita até terminar, e, então,
4. Fechar comunicação com a fita.

NORMAS PARA ENTRADA DE DADOS DA FITA

Para entrar dados da fita você deve:

1. Abrir comunicação com a fita para entrada;
2. Usar EOF(-1) para ver se já alcançou o fim do arquivo na fita;
3. Entrar dados da fita;
4. Continuar entrando dados até que você tenha terminado ou alcançado o fim do arquivo, e, então,
5. Fechar comunicação com a fita.

Digite:

```
  5 CLS : PRINT "COLOQUE A FITA E PRESSIONE PLAY"
  7 INPUT "PRESSIONE <ENTER> QUANDO PRONTO"; R$
 10 OPEN "O",#-1, "CHEQUES"
 15 CLS : PRINT "INTRODUZA CHEQUES E XX NO FINAL"
 20 INPUT "NUMERO:"; N$
 25 IF N$ = "XX" THEN 90
 30 INPUT "DATA"; D$
 40 INPUT "PAGO A:"; P$
 50 INPUT "CONTA:"; S$
 60 INPUT "QUANTIA:Cr$"; A
 70 PRINT#-1, N$, D$, P$, S$, A
 80 GOTO 15
 90 CLOSE#-1
 92 CLS : T=0
 95 INPUT "QUAL CONTA"; B$
100 PRINT "RETROCEDA A FITA - PRESSIONE PLAY"
110 INPUT "PRESSIONE <ENTER> QUANDO PRONTO" :R$
120 OPEN "I",#-1, "CHEQUES"
130 IF EOF(-1) THEN 170
140 INPUT#-1, N$, D$, P$, S$, A
150 IF B$ = S$ THEN T=T+A
160 GOTO 130
170 CLOSE#-1
180 PRINT "O TOTAL GASTO COM" B$, "FOI $" T
```

No Capítulo 20 forneceremos o programa sobre uma lista de compras. Ele lhe dará uma melhor idéia de como "arquivar" na fita.

ARQUIVAR: TÃO FÁCIL QUANTO O ABECÊ

Arquivar em ordem alfabética é muito fácil, mas requer muita paciência. Portanto, vamos fazer com que o Computador faça isso para você.

```
10 INPUT "DIGITE 2 PALAVRAS"; A$, B$
20 IF A$<B$ THEN PRINT A$ "VEM ANTES DE" B$
30 IF A$>B$ THEN PRINT A$ "VEM DEPOIS DE" B$
40 IF A$=B$ THEN PRINT "AS PALAVRAS SAO AS
   MESMAS"
50 GOTO 10
```

Rode o programa. Introduza várias palavras até você se convencer de que o Computador já sabe colocar em ordem alfabética. Com strings, os sinais maior que, menor que e igual, que discutimos no Capítulo 5, têm agora um novo significado. Eles informam se uma string vem antes que a outra na ordem alfabética:

< precede alfabeticamente
< = precede ou é a mesma alfabeticamente
> segue alfabeticamente
> = segue ou é a mesma alfabeticamente
= é a mesma

Como o Computador pode colocar em ordem alfabética, você pode escrever um programa para ordenar uma longa lista de palavras. Aqui está o nosso:

```
 10 DIM A$(5)
 20 FOR I=1 TO 5
 30 INPUT "DIGITE UMA PALAVRA"; A$(I)
 40 NEXT I
 50 X=0
 60 X=X+1
 70 IF X>5 THEN GOTO 70
 80 IF A$(X)="ZZ     " THEN 60
 90 FOR Y=1 TO 5
100 IF A$(Y)<A$(X) THEN X=Y
110 NEXT Y
120 PRINT A$(X)
130 A$(X)="ZZ     "
140 GOTO 50
```

Você pode facilmente fazer o Computador ordenar mais palavras trocando o 5 nas linhas 10, 20, 70 e 90.

Digite e rode este programa.

Antes de explicar como ele funciona, mostraremos o que está acontecendo quando o programa está sendo executado.
Digite:

```
 30 READ A$(1)
200 DATA PAULO, ROSELY, JOSE, ANTONIO, FILOMENA
```

Cancele a linha 120 e digite:

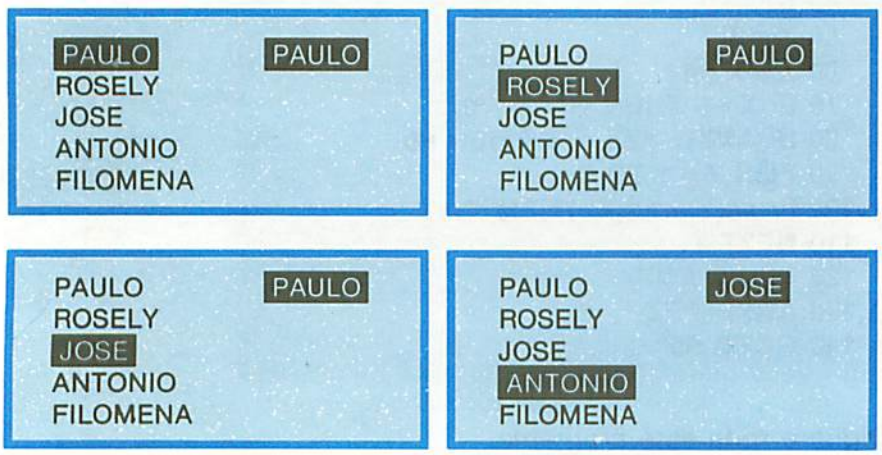
```
120
  5 CLS
 35 PRINT A$(I)
 85 V=V+1
105 PRINT@15+32*(V-1), A$(X)
135 GOSUB 500
500 FOR I=1 TO 5
510 PRINT@0+32*(I-1), A$(I);
520 NEXT I
530 RETURN
```

Estas linhas são só para organizar a tela — assim você pode ver melhor o que está acontecendo. Rode o programa.
Está muito rápido? Digite esta linha para diminuir a velocidade, assim você pode ver o que está acontecendo.

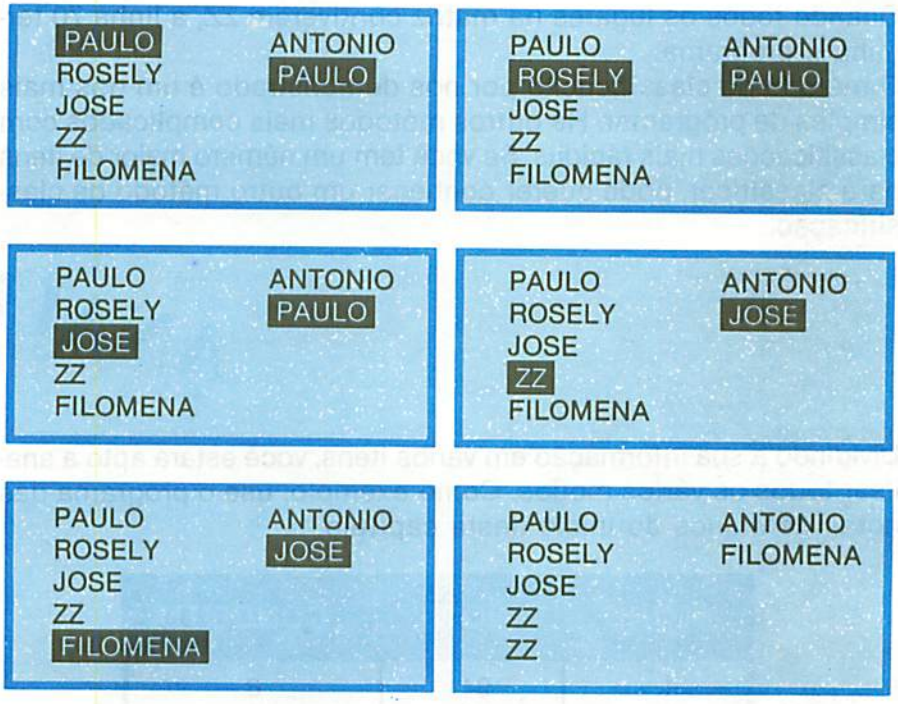
```
107 FOR T=1 TO 600 : NEXT
```

Rode o programa novamente. Veja com atenção a segunda coluna. Veja como o primeiro nome muda de PAULO para JOSE e para ANTONIO. Depois note o que acontece com ANTONIO na primeira coluna. Ele se torna ZZ.
Aqui estão os quadros de como o Computador determina a primeira e a segunda posição:

1.ª POSIÇÃO



2.ª POSIÇÃO



Quando o programa começa, PAULO é comparado com PAULO para ver qual antecede o outro alfabeticamente. PAULO permanece no alto e é comparado com ROSELY. PAULO ainda permanece no alto. Em seguida, PAULO é comparado com JOSE. Como este antecede PAULO, ele agora assume a posição de PAULO no alto da 2.ª coluna. Agora JOSE é comparado com ANTONIO que vem para cima e é comparado com FILOMENA. ANTONIO permanece no alto.

Agora que todos os nomes foram comparados com a primeira posição, o Computador repete o ciclo para determinar as demais posições. ANTONIO se torna ZZ para não mudar de posição.

As linhas 50 e 60 ajustam o valor de X. Na primeira passada do programa, X é igual a 1.

Assim, as linhas 90 até 110 comparam A$(1) — PAULO — com cada um dos outros nomes na matriz A$, até alcançar a palavra que antecede A$(1).

Em nosso exemplo, a terceira palavra — JOSE — o antecede. A linha 100, então, faz A$(X) igual a A$(3) — o lugar de JOSE na matriz. Quando JOSE é comparado com a primeira palavra — ANTONIO — A$(X) se torna A$(4).

A linha 120 imprime A$(4) — ANTONIO —, a linha 130 faz A$(4) igual a ZZ.
Neste ponto, as linhas 50 e 60 fazem X igual a 1 novamente. A$(1) — PAULO — será novamente comparado com outros nomes na matriz.
Quando a posição de PAULO na matriz se torna igual a ZZ, a linha 80 enviará o Computador de volta à linha 60 que faz X igual a 2. A$(2) — ROSELY — é então comparado com todos os nomes na matriz.
Quando todos os lugares na matriz contiverem ZZ, a linha 70 termina o programa.
O método de classificação por nós demonstrado é um dos mais simples de programar. Há outros métodos mais complicados com classificações mais rápidas. Se você tem um número maior de itens para classificar, pode querer conhecer um outro método de classificação.

ANALISANDO

Dividindo a sua informação em vários itens, você estará apto a analisar todos os vários modos. Como exemplo, use o programa das notas de alunos do início deste capítulo.

ALUNO	NOTA	CONCEITO
1	94	6
2	20	5
3	10	4

Para simplificar, usamos três alunos apenas.
Dando a cada item dois índices, nós podemos colocar tudo em uma única matriz.

MATRIZ P		
	NOTA	CONCEITO
aluno 1	P(1, 1) 94	P(1, 2) 6
aluno 2	P(2, 1) 20	P(2, 2) 5
aluno 3	P(3, 1) 10	P(3, 2) 4

No exemplo anterior nós criamos a Matriz A para a nota e a Matriz B para o conceito. Agora eles serão colocados numa mesma Matriz P.

O primeiro índice indica o aluno e o segundo, a nota ou o conceito relacionado a ele. Por exemplo, usamos P(1, 2) para indicar os 6 pontos de conceitos para o aluno 1.

Vamos colocar todos em um programa. Digite:

```
  5 DIM P(3,2)
 10 DATA 94, 6, 20, 5, 10, 4
 20 FOR A=1 TO 3
 30 FOR P=1 TO 2
 40 READ P(A,P)
 50 NEXT P
 60 NEXT A
 70 INPUT "ALUNO N. (1-3)"; A
 80 IF E<1 OR E>3 THEN 70
 90 INPUT "NOTA OU CONCEITO" (1-2)";P
100 IF P<0 OR P>2 THEN 90
110 PRINT P (A, P)
120 GOTO 70
```

A linha 5 reserva um espaço na memória para uma matriz chamada P. Cada item pode ter 2 índices. O primeiro índice não pode ser maior que 3, nem o segundo maior que 2.
Rode o programa. Todos os pontos foram indexados. Tente diferentes combinações de número de alunos, notas e conceitos.
As linhas 20 até 60 lêem os pontos para notas e conceitos dos alunos 1, 2 e 3, e, antes de colocá-los na memória do Computador, identifica toda a matriz. Assim:

```
P(1,1) 94   P(1,2) 6
P(2,1) 20   P(2,2) 5
P(3,1) 10   P(3,2) 4
```

Agora que você tem dois índices para identificar os pontos, seu programa pode ficar mais poderoso.
Cancele as linhas 70-120 e digite:

```
  70 INPUT "DIGITE <1> ALUNO OU <2> PONTOS"; R
  80 IF R<1 OR R>2 THEN 70
 100 ON R GOSUB 1000, 2000
 110 GOTO 70
1000 INPUT "ALUNO N.(1-3)"; A
1010 IF A<1 OR A>3 THEN 1000
1015 CLS
1020 PRINT@132, "PONTOS DO ALUNO" A
1030 PRINT
1040 FOR P=1 TO 2
1050 PRINT "PONTOS" P
1060 PRINT P (A, P)
1070 NEXT P
```

```
1080 RETURN
2000 INPUT "NOTAS <1> OU CONCEITO <2>"; P
2010 IF P<1 OR P>2 THEN 2000
2015 CLS
2020 PRINT@132, "PONTOS DO ALUNO" P
2030 PRINT
2040 FOR A=1 TO 3
2050 PRINT "ALUNO" A
2060 PRINT P (A, P)
2070 NEXT A
2080 RETURN
```

Rode o programa. O que você acaba de programar é uma matriz bidimensional. Ela é chamada bidimensional porque todos os seus itens têm dois índices — notas e conceitos. No capítulo anterior nós programamos matrizes unidimensionais — seus itens tinham apenas um índice.

A TERCEIRA DIMENSÃO

Nós agora estamos prontos para considerar os dois semestres e o exame — a matriz de 3 dimensões. As estatísticas dos pontos mostrados acima eram todos do mesmo semestre. Aqui está a informação:

1° SEMESTRE (1)		
	NOTA	CONCEITO
aluno 1	94	6
aluno 2	20	5
aluno 3	10	4

2° SEMESTRE (2)		
	NOTA	CONCEITO
aluno 1	50	5
aluno 2	81	1
aluno 3	75	2

EXAME (3)		
	NOTA	CONCEITO
aluno 1	40	1
aluno 2	52	3
aluno 3	70	4

Eis aqui como nós identificamos todos estes pontos na Matriz P.

MATRIZ P		
1° SEMESTRE		
	NOTA	CONCEITO
aluno 1	P(1,1,1) 94	P(1,1,2) 6
aluno 2	P(1,2,1) 20	P(1,2,2) 5
aluno 3	P(1,3,1) 10	P(1,3,2) 4

2° SEMESTRE		
	NOTA	CONCEITO
aluno 1	P(2,1,1) 50	P(2,1,2) 5
aluno 2	P(2,2,1) 81	P(2,2,2) 1
aluno 3	P(2,3,1) 75	P(2,3,2) 2

EXAME		
	NOTA	CONCEITO
aluno 1	P(3,1,1) 40	P(3,1,2) 1
aluno 2	P(3,2,1) 52	P(3,2,2) 3
aluno 3	P(3,3,1) 70	P(3,2,2) 4

Para ter tudo isso na memória de seu Computador, apague seu programa e digite:

```
 5 DIM P(3,3,2)
10 DATA 94, 6, 20, 5, 10, 4
20 DATA 50, 5, 81, 1, 75, 2
30 DATA 40, 1, 52, 3, 70, 4
40 FOR S=1 TO 3
50 FOR A=1 TO 3
60 FOR P=1 TO 2
70 READ P (S,A,P)
80 NEXT P
90 NEXT A
100 NEXT S
110 INPUT "SEMESTRE <1>, SEMESTRE <2>, EXAME
    <3>"; S
120 IF S<1 OR S>3 THEN 110
130 INPUT "SEMESTRE (1-3)"; A
```

```
140 IF A<1 OR A>3 THEN 130
150 INPUT "NOTAS <1>, CONCEITOS <2>"; P
160 IF P<1 OR P>2 THEN 150
170 PRINT P (S,A,P)
180 GOTO 110
```

Rode o programa e teste todos esses índices. As linhas 40 a 100 colocam isto na memória do seu Computador:

MEMÓRIA DE SEU COMPUTADOR

P(1,1,1)→94	P(1,1,2)→6
P(1,2,1)→20	P(1,2,2)→5
P(1,3,1)→10	P(1,3,2)→4
P(2,1,1)→50	P(2,1,2)→5
P(2,2,1)→81	P(2,2,2)→1
P(2,3,1)→75	P(2,3,2)→2
P(3,1,1)→40	P(3,1,2)→1
P(3,2,1)→52	P(3,2,2)→3
P(3,3,1)→70	P(3,3,2)→4

Leve vantagem de todas as três dimensões, anule as linhas 110 a 180 e digite:

```
 110 PRINT : PRINT "DIGITE <1> PARA PERIODO"
 120 PRINT "<2> P/ ALUNO OU <3> P/ PONTOS"
 130 L=224 : INPUT R
 140 ON R GOSUB 1000, 2000, 3000
 150 GOTO 110
1000 INPUT "SEMESTRE <1>, SEMESTRE <2>, EXAME
     <3>"; S
1010 IF S<1 OR S>3 THEN 1000
1020 CLS
1030 PRINT@102, "PONTOS DO PERIODO" S
1040 PRINT@168, "NOTAS"
1050 PRINT@176, "CONCEITOS"
1060 FOR A=1 TO 3
1070 PRINT@L, "ALUNO" A
1080 FOR P=1 TO 2
1100 PRINT@L+8*P, P(S,A,P);
1110 NEXT P
1120 L=L+32
1130 NEXT A
1140 RETURN
```

```
2000 INPUT "ALUNO (1-3)"; A
2010 IF A<1 OR A>3 THEN 2000
2020 CLS
2030 PRINT@102, "PONTOS DO ALUNO" A
2040 PRINT@168, "NOTAS"
2050 PRINT@176, "CONCEITOS"
2060 FOR S=1 TO 3
2070 PRINT@L, "PERIODO" S
2080 FOR P=1 TO 2
2100 PRINT@L+8*P, P(S,A,P);
2110 NEXT P
2120 L=L+32
2130 NEXT S
2140 RETURN
3000 INPUT "NOTA <1>, CONCEITO <2>"; P
3010 IF P<1 OR P>2 THEN 3000
3020 CLS
3030 PRINT@102, "PONTOS DO TIPO" P
3040 PRINT@168, "ALUNO 1"
3050 PRINT@176, "ALUNO 2"
3060 PRINT@184, "ALUNO 3"
3070 FOR S=1 TO 3
3080 PRINT@L, "PERIODO" S
3090 FOR A=1 TO 3
3100 PRINT@L+8*A,P(S,A,P);
3110 NEXT A
3120 L=L+32
3130 NEXT S
3140 RETURN
```

EXERCÍCIO DO CAPÍTULO

Escrever um programa para distribuir as cartas usando uma matriz de 2 dimensões. Uma dimensão pode ser um dos 4 naipes; a outra, os valores das cartas (1-13).

SEÇÃO

III

PROGRAMAÇÃO AVANÇADA

Agora que você já passou pela Seção II e conhece os fundamentos de programação e sabe utilizar as várias instruções da linguagem BASIC, você está preparado para descobrir as características avançadas do COLOR BASIC do seu CP 400.
O que você vai aprender agora lhe ajudará a melhorar ainda mais seus programas, tornando-os mais simples e eficientes, permitindo que você mostre toda a sua criatividade.

CAPÍTULO 11

pontos e linhas

Uma das características mais excitantes do COLOR BASIC é sua capacidade de mostrar desenhos (tais como linhas e círculos) mais precisos, mais variados e mais fáceis de usar do que os mostrados na seção anterior.
Vamos começar com o elemento gráfico mais básico — um simples ponto — e construir a partir daí.
Com o COLOR BASIC fica realmente simples colocar um ponto sobre a tela. Digite o seguinte programa e você perceberá o que queremos dizer:

```
 5 PMODE 1,1
10 PCLS
20 SCREEN 1,1
30 PSET (10,20,8)
40 GOTO 40
```

Não se preocupe com qualquer das novas palavras. Por exemplo, PMODE e SCREEN determinam o grau de detalhe e o conjunto de cores utilizadas. Isso será explicado mais à frente.

Agora rode o programa. A tela deve estar cinza e, se olhar cuidadosamente, você verá um ponto alaranjado no canto superior esquerdo. Esse ponto foi colocado lá por PSET na linha 30 do programa.
Esta instrução permite a você colocar um ponto de uma determinada cor em qualquer lugar sobre a tela. PSET tem o seguinte formato:

PSET (x, y, c)

x especifica uma posição sobre o eixo X (horizontal) e é uma expressão numérica cujo resultado deve estar entre 0 e 255.
y especifica uma posição sobre o eixo Y (vertical) e é uma expressão numérica entre 0 e 191.
c especifica a cor do ponto (código de cor) e é uma expressão numérica entre 0 e 8.
Nota: *A coordenada X deve vir sempre antes da coordenada Y.*

Nós chamamos isto de "Quadro de sintaxe". Estes quadros aparecerão muito nesta seção do livro. Em geral, eles explicam instruções tais como a PSET. Preste muita atenção na sua organização e pontuação. Você substituirá os parâmetros x, y e c pelos valores que você precisar.

O jeito que seu Computador faz linhas grossas ou finas é muito simples. Na resolução mais alta (linhas mais finas), o seu Computador usa um pequeno retângulo na tela para colocar um ponto.

Na resolução mais baixa, o Computador combina quatro desses retângulos para cada ponto. Conseqüentemente, sempre que você ativa um dos retângulos com uma certa cor, os outros três também são ativados. Entretanto, não importa a resolução que você use, ainda assim devem ser especificadas as coordenadas x de 0 a 255 e y de 0 a 191. Mais tarde, voltaremos a tocar nesse assunto.

Mesmo que você não possa ver, o Computador tem a tela dividida em aproximadamente 50 mil pontos. Cada um desses pontos pode fazer parte de uma grade, se você pensar na sua tela como uma folha de papel quadriculado de 256 por 192 pontos. Você precisará recorrer às coordenadas (X, Y) para definir exatamente a posição do ponto que você tem em mente. No Apêndice E você encontrará um quadro com as posições gráficas na tela, dividido em 256 x 192 pontos. Isso será valioso para localizar os pontos, usando quaisquer das funções gráficas. Em nosso exemplo, a linha 30 indica o ponto laranja:

```
30 PSET (10,20,8)
```

Você pode ver como PSET especifica a posição do ponto contando até 10 no eixo X e até 20 no eixo Y. Note que Y é contado de cima para baixo, enquanto X, da esquerda para a direita.

Você acha que consegue colocar um segundo ponto sobre a tela da mesma cor que o primeiro? Como colocar no canto inferior direito? Acrescente uma outra linha de programa para colocar esse ponto:

```
35 PSET (255,191,8)
```

Agora rode o programa. Deve aparecer um outro ponto no canto inferior direito da tela. Liste seu programa:

```
 5 PMODE 1,1
10 PCLS
20 SCREEN 1,1
30 PSET (10,20,8)
35 PSET (255,191,8)
40 GOTO 40
```

MARCANDO A COR

Até aqui, nós falamos duas coisas sobre o *c*:
1. Ele especifica a cor do ponto.
2. Ele pode ser qualquer número de 0 a 8.

Você deve se lembrar do Capítulo 2 que cada uma das 9 cores disponíveis tem um código numérico. Nas linhas 30 e 35 do programa, nós especificamos o laranja (código 8) como a cor do ponto.

Dentro de certos limites, você pode mudar a cor do ponto alterando o código de cor.

Entretanto (e isso é muito importante!), você pode não obter a cor que especificou. Há uma boa razão para isso. Nós discutiremos isto no próximo capítulo quando falarmos sobre os "modos" diferentes de gráficos; mas, por ora, não fique preocupado se não conseguir a cor que você quer.

Nota muito importante! *O CP 400 é capaz de gerar 9 cores diferentes. São elas: preto, verde, amarelo, azul, vermelho, cinza, ciano, roxo e laranja. Entretanto, os tons coloridos reais produzidos pelo seu aparelho e a diferença entre os tons dependerão da qualidade e ajuste de cor de sua televisão — não do Computador. Sugerimos que você rode o programa de ajuste e teste de cor que está na seção I e ajuste a cor da maneira que mais lhe agradar antes de rodar seus programas.*

APAGUE ESTE PONTO!

É tão fácil apagar um pingo quanto ativar um. Você imagina como fazê-lo? Lembra-se de como você fez o Computador piscar no *Capítulo 4*? Você usou RESET (que torna um ponto específico na cor de fundo). Você pode fazer a mesma coisa no COLOR BASIC usando PRESET.

PRESET (x, y)

x especifica a posição sobre o eixo X e é uma expressão numérica entre 0 e 255.
y especifica a posição sobre o eixo Y e é uma expressão numérica entre 0 e 191.

Observe que você não precisa especificar a cor com PRESET, pois o Computador usa automaticamente a cor do fundo.

A MENOR DISTÂNCIA ENTRE DOIS PONTOS

Tudo certo, você já sabe como colocar um ponto em qualquer lugar da tela e até mais de um. Muito bem. E o que você pode fazer com isso?
Se você desenhar dois pontos sobre um pedaço de papel, qual a primeira coisa que você pensa fazer? Na certa vai querer ligá-los com uma reta. Por que não tentar isso aqui? Antes porém...

Você se lembra como usou uma série de blocos para fazer uma linha no começo deste livro? Isso funcionou para uma linha vertical ou horizontal (mesmo sendo um pouco larga para certos desenhos), mas uma diagonal pareceria mais com uma escada do que uma reta (isso sem mencionar como ficaria um círculo!).
Agora vamos mostrar como seu Computador pode traçar uma linha horizontal ou vertical muito mais fina, e ainda uma verdadeira diagonal. E, mais ainda, agora você pode variar a espessura da linha.
Um jeito de fazer isso é usar a instrução LINE do COLOR BASIC. Para ver a LINE funcionando, use o programa dos pontos, porém um pouco modificado (e, para facilitar, no futuro nós o chamaremos "Linhas"). Mude primeiro a linha 30 do programa para:

```
30 LINE (0,0)-(255,191),PSET
```

Agora apague a linha 35:

```
35 ENTER
```

O seu programa fica assim:

```
 5 PMODE 1,1
10 PCLS
20 SCREEN 1,1
30 LINE (0,0)-(255,191),PSET
40 GOTO 40
```

Na verdade, nada mudou. Você ainda usa blocos (ou pontos) para fazer uma linha, porém os gráficos de alta resolução utilizam pontos menores.

Agora rode o programa. Você deve ter uma linha alaranjada que corre do topo esquerdo do mostrador para o canto inferior direito, enquanto o fundo deve ser cinza. Muito bom!
Como mudar a *direção* da linha, digamos, do canto inferior esquerdo para o superior direito?
Complicado? Então experimente mudar a linha 30 para:

```
30 LINE (0,191)-(255,0), PSET
```

Isso dá a você uma idéia de como funciona a instrução LINE?

E SE EU QUISER VÁRIAS LINHAS?

Por que você não recoloca a instrução da linha 30 anterior (como linha 25, por exemplo) e roda o programa? Sua TV mostra duas linhas alaranjadas que se cruzam no centro da tela?

Na verdade, você pode traçar tantas linhas quanto quiser — desde que você tenha aprendido o formato. Como uma linha reta é definida como a menor distância entre dois pontos, a primeira coisa que se precisa fazer é definir onde devem ficar esses pontos. Os pontos, onde começa e termina nossa linha original, foram colocados dentro dos parênteses:

```
25 LINE (0,0)-(255,191),PSET
```

Esses pontos são identificados pelas coordenadas (X,Y). Cada um desses pontos pode ser localizado em um quadro com as posições gráficas na tela, no Apêndice E. Esses pontos devem, então, ser incluídos no seu comando LINE no seguinte formato:

LINE (x1,y1)-(x2,y2),a,b

(x1,y1) especifica o ponto inicial da linha. x1 é uma expressão numérica entre 0 e 255; y1, entre 0 e 191.
(x2,y2) especifica o ponto final da linha. x2 é uma expressão numérica entre 0 e 255; y2, entre 0 e 191.
a é PSET ou PRESET.
b é B ou BF. Isso é opcional.

Falaremos sobre os parâmetros **a** *e* **b** *mais tarde.*

Experimente usar o quadro com as posições gráficas para plotar (localizar) os pontos que usamos para criar as linhas concorrentes em nosso programa "Linhas". Lembre-se de que o primeiro conjunto de coordenadas especifica o ponto inicial da linha. O segundo (após o hífen) especifica o ponto final.
Observe também que nem sempre é necessário especificar o ponto inicial da linha (o primeiro conjunto de parênteses). Se você não especificar o ponto inicial, o Computador começará automaticamente no último ponto final. Se você ainda não tiver usado a instrução LINE no seu programa, o Computador começa a linha em (128,96). Por exemplo:

Nota: *Esse PSET não é o mesmo do começo do capítulo. Ele não posiciona um ponto especificado sobre a tela ou especifica um código de cor. Ele tem uma função totalmente diferente. Mais à frente você verá que não é tão difícil identificar esse PSET.*

```
20 LINE (0,0)-(255,191),PSET
30 LINE -(191,0),PSET
```

A linha 20 do programa traça uma linha a partir de (0,0) estendendo-se até (255,191). A linha 30 do programa desenha uma linha começando em (255,191) estendendo-se até (191,0).
Se você não especificar um ponto inicial, não se esqueça de colocar o hífen (-) antes do ponto final.

DESENHAR OU APAGAR?

Vamos olhar as linhas do programa que criaram as linhas concorrentes:

```
25 LINE (0,0)-(255,191),PSET
30 LINE (0,191)-(255,0),PSET
```

Até aqui nós só falamos do que está dentro dos parênteses, mas, se você der uma olhada no bloco de sintaxe outra vez, verá que é possível escolher o próximo parâmetro — PSET ou PRESET. Antes de explicarmos a diferença entre PSET e PRESET, experimente mudar PSET na linha 25 para PRESET e rode o programa:

```
25 LINE (0,0)-(255,191),PRESET
```

O que aconteceu? A tela de sua TV deve mostrar uma única linha alaranjada (criada na linha 30) que corre do canto inferior esquerdo da tela para o canto superior direito. Você já sabe o que acontece quando PSET é trocado por PRESET?
Substitua PSET na linha 30 por PRESET e veja o que acontece. Sua tela ficou limpa?

VAMOS DAR MAIS UMA CHECADA

- **PSET** traça a linha com a cor predeterminada para primeiro plano.
- **PRESET** utiliza a cor predeterminada como de fundo para traçar a linha.

Isso equivale, na prática, a "apagar" a linha.
No nosso programa estamos usando o laranja como cor em primeiro plano (a cor da linha, certo?) e o cinza como cor de fundo (a cor da tela em que você está desenhando).
Você pode só distinguir entre as cores de fundo e de primeiro plano se pensar numa linha como uma cor mostrada sobre a cor que está na tela. É a mesma idéia de usar diferentes cores de giz para escrever num quadro. Alguns padrões são verdes, outros pretos etc. A cor do quadro seria a cor de fundo; o giz, a cor de primeiro plano.

Se você tentar escrever num quadro verde com giz verde, você não será capaz de ver o que está escrevendo. Isso é exatamente o que aconteceu quando você colocou o PRESET na linha 25. Você instruiu o Computador para mudar a cor da linha para a cor de fundo! Por isso você não pôde ver nada.

A especificação das cores de fundo e de primeiro plano não são feitas no comando LINE. Nós voltaremos a falar sobre isso mais tarde. Por ora, concentre-se no LINE.

LINHAS E RETÂNGULOS

Ainda existem alguns itens para estudarmos sobre o comando LINE. O B, por exemplo, vem do inglês "box", que quer dizer caixa. O COLOR BASIC facilita o traçado de uma caixa (retangular ou quadrada). Tudo o que você tem a fazer é especificar uma das diagonais da caixa e colocar o parâmetro B. Assim, quando você rodar o programa, seu Computador traçará uma caixa ao invés de uma linha.
Para ilustrar isso, vamos chamar o programa "Linhas" de volta.

```
 5 PMODE 1,1
10 PCLS
20 SCREEN 1,1
25 LINE (0,0)-(255,191),PSET
30 LINE (0,191)-(255,0),PSET
40 GOTO 40
```

Apague a linha 30 e adicione o sufixo ",B" à linha 25. Assim:

```
25 LINE (0,0)-(255,191),PSET,B
```

Agora rode o programa. Que tal, entendeu?

Em todas as modificações das linhas de seu programa você está usando o modo de Edição? Lembra-se dele? Capítulo 6.

COMO PINTAR A CAIXA

Se você consultar o bloco de sintaxe para LINE, você verá que tem a opção de adicionar "F" ao sufixo opcional ",B". Esta opção permite que você pinte a caixa com a cor de primeiro plano. Experimente. Mude a linha 25 para:

```
25 LINE (0,0)-(255,191),PSET,BF
```

Que tal? (Você conseguiu uma grande caixa alaranjada (256 × 192) sobre um fundo cinza.)

PARA NÃO DIZER QUE NÃO FALEI DAS CORES

Como já faz algum tempo que estamos conversando sobre cores de fundo e de primeiro plano, já é tempo de você saber pelo menos parte do modo como se controla essas cores.
Dentro de certos limites, a propriedade gráfica COLOR permite a você determinar as cores de fundo/primeiro plano. Esses limites serão discutidos mais tarde.

COLOR primeiro plano, fundo

Primeiro plano é um número de 0 a 8 e representa um código de cor.
Fundo é um número de 0 a 8 e representa um código de cor.

Até agora, nós somente mencionamos a cor cinza (5) e laranja (8), usadas no nosso programa. Por que foram usadas estas duas? Quando você não especifica as cores de fundo e primeiro plano, o seu Computador escolhe automaticamente o código de cor disponível mais alto para a cor de primeiro plano e o mais baixo para a cor de fundo. É por isso que as linhas eram laranja (8) sobre o fundo cinza (5) no programa "Linhas".
Entretanto, com COLOR você pode alterar essa combinação de cores (entre as opções disponíveis). Coloque a linha 6 no seu programa e apague a linha 25:

```
6 COLOR 5,7
```

No próximo capítulo falaremos sobre as várias combinações de cor para essa nova instrução COLOR.

EXERCÍCIO DO CAPÍTULO

Vamos voltar à infância. Lembra-se de quando você desenhava casinhas? Então, que tal, usando esses novos recursos de seu CP 400, desenhar uma casinha, com montanhas ao fundo e tudo mais que sua imaginação permitir?
O nosso programa está no Apêndice A.

CAPÍTULO 12

na língua do p

Quando você envia qualquer tipo de informação para a TV, ela é armazenada numa parte da memória chamada "RAM de vídeo". O circuito de TV do Computador lê essa "parte da memória" para saber o que vai aparecer na tela. A área "normal" de memória é suficientemente grande para texto (letras e números), mas não para gráficos (círculos, linhas etc.). Conseqüentemente, o Computador precisa de uma área (parte) maior para uso com gráficos.
Esta parte da RAM de vídeo pode conter até 8 "páginas" de memória, e, dependendo de suas necessidades, você pode separar de uma a oito páginas com PCLEAR. (Uma "página" contém 1 536 posições de memória.) PCLEAR determina o tamanho da memória disponível para gráficos. Quando você não usar PCLEAR, o Computador determinará automaticamente quatro páginas.
O número de páginas que você precisa depende da quantidade de detalhes ou número de cores que você quer nos seus gráficos. Quanto maior a necessidade de detalhamento e seleção de cores, mais memória é exigida.

DO MODO QUE VOCÊ QUISER

Modifique o programa "Linhas" para que novamente ele crie linhas que se cruzem:

```
 5 PMODE 1,1
10 PCLS
20 SCREEN 1,1
25 LINE (0,0)-(255,191), PSET
30 LINE (0,191)-(255,0), PSET
40 GOTO 40
```

Agora a linha 5 contém uma das mais importantes funções gráficas do seu Computador — PMODE.

```
5 PMODE 1,1
```

PMODE permite a você escolher a resolução (quantidade de detalhes) e selecionar em que "página" da memória seu gráfico vai começar.

PMODE MODO, PÁGINA DE INÍCIO

Modo é um número de 0 a 4. É opcional; se omitido, é usado o valor anterior. Se ainda não foi especificado nenhum *modo*, será usado o 2.
Página de início é um número de 1 a 8 e especifica em que página da memória você deseja iniciar. Também é opcional; se omitido, será usada a página já definida anteriormente. Se for a primeira vez que PMODE é executado, será considerado o valor 1 (ou seja, a primeira página).

Já foi visto que quando você trabalha com gráficos, o Computador divide a tela da TV em 256 pontos sobre o eixo X e em 192 sobre o eixo Y. Um quadriculado de 256 × 192 equivale ao maior nível de resolução disponível (o mais detalhado) no COLOR BASIC. Por mera casualidade, chamamos este modo de "PMODE 4". O seu Computador tem 5 modos (0-4), cada um com características distintas. Você já sabe que a única diferença entre alta e baixa resolução (PMODE 4 e PMODE 0, respectivamente) é o tamanho do ponto: a alta resolução usa um único ponto para formar cada ponto na tela, enquanto a baixa resolução combina 4 desses pontos para um maior.
Mesmo que o tamanho do quadriculado varie (com a combinação entre os pontos), todos os modos são mapeados sobre uma tela de 256 × 192 e você deve especificar coordenadas dentro dessa faixa. Por exemplo, (128,96) é o centro da tela, não importando em que modo você esteja. Da mesma forma, (255,191) é o canto inferior direito tanto para o PMODE 0 como para o PMODE 4.

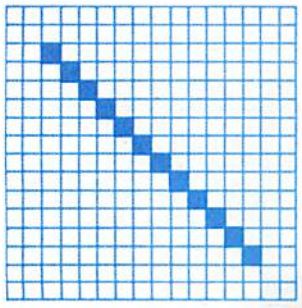
PMODE não é necessário quando você está usando a tela para programas de "texto".

TABELA 1

PMODE	PONTOS NA TELA	MODO DE COR	PÁGINAS USADAS	TAMANHO DO PONTO
4	256 × 192	2 cores	4	□
3	128 × 192	4 cores	4	▭
2	128 × 192	2 cores	2	
1	128 × 96	4 cores	2	⊞
0	128 × 96	2 cores	1	

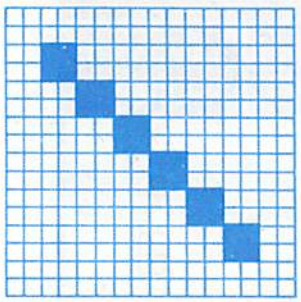
Na Tabela 1, você pode ver porque o PMODE 4 lhe dá a mais alta resolução (mais detalhes). Simplesmente porque tem os menores pontos. Esse é o motivo pelo qual vamos nos referir daqui para frente ao modo 4 como "alta resolução".
Em seguida, vêm os modos 3 e 2. Como os pontos desses modos são duas vezes o tamanho de modo 4, a resolução não é tão alta. Nós os chamamos "média resolução".
Os modos 1 e 0 têm pontos 4 vezes o tamanho do modo 4 e duas vezes os modos 3 e 2. Como esta é a mais baixa resolução dos modos gráficos, nós os chamamos "baixa resolução".
Você também pode ver a coisa do ponto de vista de que a alta resolução necessita de 4 vezes o número de pontos que a baixa resolução precisa para encher a tela.
Os exemplos a seguir ilustram a diferença entre os modos 4 e 0 em relação ao tamanho do ponto e resolução.
Você se lembra quando estava na escola e precisava usar o lápis grande para fazer linhas grossas sobre o papel?
Ocorre o mesmo com o seu Computador. Pode-se pensar na baixa resolução como um lápis grande, com um ponto duas vezes maior que o ponto fino da alta resolução. Isso porque, com alta resolução, você pode criar linhas diagonais que não lembrem uma escada, para não falar de círculos que não são realmente redondos.

Não se esqueça de que a tela de gráfico está sempre cheia de "pontos". A questão simplesmente é saber o tamanho, a cor e quantos pingos estão lá.



ALTA RESOLUÇÃO

Mesmo que chamemos isto de baixa resolução, ela ainda tem mais detalhes (maior resolução) que os gráficos SET/ RESET 64 × 32 na tela de texto normal.



BAIXA RESOLUÇÃO

Vamos voltar ao programa "Linhas" e estudar a linha 5:

```
5 PMODE 1,1
```

Mude-a para a alta resolução (PMODE 4) e rode o programa:

```
5 PMODE 4,1
```

Você deve observar duas diferenças:
1. A linha fica mais fina porque está em alta resolução.
2. Também deve ter havido uma mudança de cor, pois você pulou do modo de 4 cores para o de 2 cores.

Assim, você pode ver no programa acima que nós sempre precisamos verificar se o modo escolhido tem o conjunto de cores que queremos com o nível de detalhamento que precisamos.
Isso significa que sempre que você está no modo 4, por exemplo, você está limitado a uma combinação de 2 cores. E só vai poder usar uma combinação de preto com verde ou de preto com cinza — nenhuma outra.
Na tabela estão as cores que nós chamamos de "cores disponíveis". Isto é, as cores que estão disponíveis para cada modo que você pode escolher.

TABELA 2
CONJUNTO DE CORES

PMODE	CONJUNTO DE CORES (VER SCREEN)	COMBINAÇÃO DE DUAS CORES	COMBINAÇÃO DE QUATRO CORES
4	0	preto/verde	—
	1	preto/cinza	—
3	0	—	verde/amarelo/azul/ vermelho
	1	—	cinza/ciano roxo/alaranjado
2	0	verde/preto	—
	1	preto/cinza	—
1	0	—	verde/amarelo/azul/ vermelho
	1	—	cinza/ciano roxo/alaranjado
0	0	preto/verde	—
	1	preto/cinza	—

O modo 4 permite alta resolução mas uma escolha limitada de cores. Por exemplo, se você quiser usar a cor vermelha, não pode usar o modo 4, você terá de escolher uma resolução mais baixa, digamos, o modo 3, com o conjunto de cores 0.

Abaixo você tem um programa que mostra um triângulo nos diferentes modos:

- Observe como seus lados ficam mais firmes e contínuos, e também como mudam as cores quando o triângulo passa através dos diferentes modos.

```
 5 FOR MODO = 0 TO 4
10 PMODE MODO,1
20 PCLS
30 SCREEN 1,1
40 LINE (70,150)-(186,150), PSET
50 LINE -(128,41), PSET
60 LINE -(70,150), PSET
70 FOR Y = 0 TO 500: NEXT Y
80 NEXT MODO
90 GOTO 5
```

PASSANDO PARA UMA NOVA PÁGINA

Como já vimos, o COLOR BASIC dividiu a memória gráfica de vídeo em 8 "páginas" de memória.
O segundo parâmetro de PMODE, a *página de início*, permite selecionar em qual das páginas você quer o início dos gráficos. Se você não especificar qual página, o Computador começa automaticamente na página 1 ou na última página especificada.

Essa tabela mostra o número de páginas exigidas para cada PMODE. Observe que quanto maior a resolução e mais cores disponíveis, mais memória será exigida pelo Computador (e, conseqüentemente, sobrará menos para o seu programa). Em outras palavras, o PMODE 0 requer uma página, o PMODE 2 requer duas páginas etc.

Rodando o programa, você terá uma idéia melhor do que nós queremos dizer.

```
 5 PMODE 3,1
10 PCLS
20 SCREEN 1,1
30 LINE (110,20)-(120,30),PSET,B
40 GOTO 40
```

TABELA 3		
	USAR...	PÁGINAS RESERVADAS COM ESTE...
n = 1 a 5	PMODE 4,n	PCLEAR 4 + (n – 1)
	PMODE 3,n	PCLEAR 4 + (n – 1)
n = 1 a 7	PMODE 2,n	PCLEAR 2 + (n – 1)
	PMODE 1,n	PCLEAR 2 + (n – 1)
n = 1 a 8	PMODE 0,n	PCLEAR 1 + (n – 1)

Nota: *Cada página reservada ocupa 1 536 posições de memória (bytes). Nada mal, hein? Um pequeno quadrado próximo ao topo de sua tela de TV.*

Agora mude a linha 5 e rode o programa novamente:

```
5 PMODE 3,2
```

De onde apareceu o ?FC ERRO?
Mudar a *página de início* de 1 para 2 foi o que ocasionou o erro. O modo 3 exige 4 páginas de memória para gráficos. Quando você começa na página 2, isso significa que foram requeridas as páginas 2, 3, 4 e 5. Porém você reservou apenas as páginas 1-4. Assim, a página 5 não está disponível!
Depois nós mostraremos como contornar esse problema de modo que você possa usar o PMODE 3,2.

O seu programa "Linhas" ainda está assim:

```
 5 PMODE 1,1
10 PCLS
20 SCREEN 1,1
25 LINE (0,0)-(255,191),PSET
30 LINE (0,191)-(255,0),PSET
40 GOTO 40
```

Agora vamos examinar o PCLS, a segunda instrução do programa. O PCLS apenas limpa a tela de gráfico. Em outras palavras, o PCLS tem a mesma função para a tela de gráficos que o CLS tem para a tela de texto.
A cor de fundo atual é usada para limpar a tela de gráfico, salvo se você especificar uma outra (disponível):

PCLS cor
A cor é um número de 0 a 8 e representa um dos códigos de cor disponíveis e é opcional: se omitido, é usada a cor de fundo.

No programa "Linhas", a opção de cor PCLS foi omitida de modo que o Computador usa automaticamente a cor de fundo, o cinza. Mude a linha 10 para:

```
10 PCLS 6
```

A sua TV deve mostrar linhas alaranjadas sobre um fundo ciano (código de cor 6).
Não se esqueça de que você só pode limpar a tela de gráficos com uma das cores disponíveis.

É HORA DE CLAREAR AS COISAS

Ao trabalhar páginas de memória, você precisará eventualmente de PCLEAR para reservar um determinado número de páginas para seus gráficos.
Como PCLEAR determina o tamanho da RAM de vídeo, ele deve ser a primeira ou segunda instrução no seu programa (após CLEAR, se esta for usada).

PCLEAR n

n é um número de 1 a 8 e especifica o número de páginas de gráfico reservadas.
Nota: *No COLOR BASIC PCLEAR 4 é feito automaticamente. Você precisa do PCLEAR apenas quando você quer reservar um número diferente de páginas.*

Provavelmente você está pensando porque não deixar simplesmente o Computador usar PCLEAR 8 o tempo todo. A razão é: PCLEAR 8 diminui a quantidade de memória disponível para seu uso. Há vezes em que você precisará muito espaço de memória para seu programa e outras em que você precisará mais páginas de memória para gráficos. PCLEAR permite que você faça este ajuste.
Agora já entendemos a função de PCLEAR. Porém, como você pode usar todas essas páginas extras que ficaram disponíveis? A resposta está no segundo parâmetro do PMODE, a *página de início*.
Suponha que você queira movimentar a figura do gráfico. Uma maneira é usar mais de uma imagem, alternadas na tela para conseguir assim o efeito de animaçãc.

SOBE E DESCE

Você pode pensar provavelmente que o seu Computador é um pouco louco, mas agora nós mostraremos a você o que é um ioiô verdadeiro. Na verdade você pode chamar este programa de ioiô. Introduza-o e rode-o.

Se você ainda confundir as exigências da memória do programa com as exigências da memória de vídeo, você obterá um ?OM ERRO (Insuficiência de Memória).

Com uma exceção (você verá Circle no capítulo seguinte), o programa a seguir contém todas as propriedades as quais você foi apresentado até agora.

```
10 PCLEAR 8
20 FOR P = 1 TO 7 STEP 2 'P=PAGINA, USA TODAS AS
   8 PAGINAS
30 PMODE 1,P 'USA DUAS PAGINAS POR VEZ
40 PCLS
50 LINE (128,0)-(148,10+(P-1)*25),PSET
60 CIRCLE (128,P*25),25
70 CIRCLE (128,P*25),13
80 NEXT P
90 FOR P = 1 TO 7 STEP 2: GOSUB 120: NEXT P
100 FOR P = 5 TO 1 STEP -2: GOSUB 120: NEXT P
110 GOTO 90
120 PMODE 1,P
130 SCREEN 1,1
140 SOUND 100+P*10,1
150 RETURN
```

Como resultado, nós acrescentamos animação aos gráficos de Computador. É o mesmo conceito que está atrás do desenho animado — desenhar uma série de imagens estáticas e alterná-las rapidamente para dar a impressão de movimento.

COPIANDO

Com PCOPY, você pode copiar os conteúdos de uma página da memória numa outra página.

PCOPY fonte TO destino

fonte e **destino** são números entre 1 e 8 especificando as páginas de memória.
Nota: *Não use PCOPY para uma página que não foi reservada. Por exemplo, após PCLEAR 4, você pode copiar apenas as páginas de 1 a 4.*

Se você quiser copiar os gráficos da página 3 na página 8, digitará:

```
PCOPY 3 TO 8
```

A vantagem do PCOPY é que ele pode encurtar seus programas, eliminando a repetição das instruções de montagem dos desenhos.

VAMOS DIRETO AO PONTO

A última função P que você pode precisar, a PPOINT, testa a cor de um determinado ponto do gráfico. O seu programa pode assim usar esta informação.

PPOINT (x,y)

x especifica a coordenada X do ponto e é um número de 0 a 255.
y especifica a coordenada Y do ponto e é um número de 0 a 191.
Nota: *Use PPOINT do mesmo modo que você usou PMODE para ajustar os pontos do gráfico, caso contrário, os resultados podem ser errados.*

O programa abaixo mostra como o Computador testa 4 células sucessivas.

```
 5 PMODE 3,1
10 PCLS
20 SCREEN 1,1
30 PSET (10,11,5)
35 PSET (10,12,6)
40 PSET (10,13,7)
45 PSET (10,14,8)
50 PRINT PPOINT (10,11)
55 PRINT PPOINT (10,12)
60 PRINT PPOINT (10,13)
65 PRINT PPOINT (10,14)
```

Quando você rodar o programa, a tela deve mostrar:

```
5
6
7
8
```

que lhe diz o código de cor das células especificadas.
Apenas para se divertir, mude o PMODE na linha 5 para:

```
5 PMODE 1,1
```

Você sabe o que o seu Computador vai mostrar agora?

```
5
7
7
8
```

Qual resolução você está usando — alta ou baixa? Lembra-se, nós já vimos que a baixa resolução combina 4 pequenos pontos para fazer um maior. O que aconteceu aqui é que (10,12) e (10,13) formam o mesmo ponto na tela. Dissemos também que quando for colocado um dos pequenos pontos, todos os outros que com ele se combinam devem ser da mesma cor. E é exatamente o que acontece aqui.
Agora vamos nos concentrar sobre o comando SCREEN da linha 20, do programa "Linhas":

```
20 SCREEN 1,1
```

SCREEN dá duas instruções ao Computador.
A primeira coisa que ele diz ao Computador é se você quer usar a tela da TV para gráficos (linhas, círculos etc.) ou texto (letras, números ou mesmo blocos SET/RESET).
A segunda é que SCREEN diz ao Computador qual conjunto de cores você quer.

SCREEN dígito; conjunto de cor

dígito é: 0 para a tela de texto ou 1 para a tela de gráficos.
conjunto de cores é 0 ou 1.

Nota: *Se dígito ou conjunto de cores for um número positivo maior do que 1, seu Computador interpreta e trata esse número como se fosse 1.*

Uma expressão numérica pode ser substituída por 0 ou 1.

O primeiro parâmetro de SCREEN diz ao Computador qual tipo de tela ele deve dar a você (gráfico ou texto). No programa "Linhas", experimente mudar a linha 20 para:

```
20 SCREEN 0,1
```

Em seguida, rode o programa. O seu Computador parece em dificuldades? (Aperte BREAK para recuperar o controle.) Realmente ele não está em dificuldades. O que aconteceu é que o Computador executava o programa, mas a tela não podia mostrar quaisquer dos gráficos porque você disse ao Computador para lhe dar uma tela de *texto*.

O segundo parâmetro informa o Computador sobre o conjunto de cores que você quer. Agora mude a linha 20 para:

Nota: *Toda vez que seu programa pede texto (PRINT, INPUT), o Computador realizará automaticamente um comando SCREEN 0,0.*

```
20 SCREEN 1,0
```

Observe que você tem a tela de gráfico novamente, mas dessa vez o conjunto de cores foi alterado.
A tabela a seguir relaciona os conjuntos de cores disponíveis.

	MODO DUAS CORES	MODO QUATRO CORES
Se o conjunto de cores é 0, então	preto/verde	verde/amarelo/azul/vermelho
Se o conjunto de cores é 1, então	preto/cinza	cinza/ciano/roxo/laranja

Não se iluda acreditando que SCREEN não pode afetar a tela de texto. Sempre que você escolhe tela de texto, o Computador assume que você quer o conjunto de cores 0 (preto/verde). Porém, você pode mudar a cor de uma tela de texto especificando:

```
SCREEN 0,1
```

Usando SCREEN dessa maneira, você pode alterar a apresentação de preto sobre verde para vermelho sobre laranja. Entretanto, o Computador voltará automaticamente para verde sobre preto sempre que ele colocar qualquer coisa na tela (com PRINT, INPUT ou LINE INPUT). O programa seguinte ilustra isso.

```
 5 CLS
10 PRINT 195, "ISTO E' VERMELHO SOBRE LARANJA"
20 SCREEN 0,1
30 SOUND 100,10
40 CLS
50 PRINT 195, "ISTO E' PRETO SOBRE VERDE"
60 SOUND 130,10
70 GOTO 5
```

EXERCÍCIO DO CAPÍTULO

Vamos fixar esses conceitos?

Que tal então, usando o programa do capítulo anterior (o da casinha), você aplicar as variações de PMODE, SCREEN, PCLS ou mesmo PCOPY?

CAPÍTULO

13

figuras e cores

Tudo isso que foi falado sobre SCREEN, PMODE e PCLEAR o está deixando confuso? Nesse caso, prepare-se, pois você não viu nada ainda!
Por exemplo, você pode criar um círculo inteiro, parte de um círculo ou mesmo uma elipse usando uma única instrução, a CIRCLE.

CIRCLE (x,y),r,c,hw,início,fim

x especifica a coordenada X do centro do círculo e é um número de 0 a 255.
y especifica a coordenada Y do centro do círculo e é um número de 0 a 191.
r especifica o raio do círculo. Um ponto (posição da tela) sobre a tela é igual a uma unidade de medida.
c especifica um código de cor disponível e é um número de 0 a 8. Isto é opcional; se omitido, é usada a cor de primeiro plano.
hw especifica a razão altura/largura e é um número de 0 a 255. Isto é opcional; se omitido, é usado 1.
início especifica o ponto de partida do círculo e é um número entre 0 e 1. Isto é opcional; se omitido, é usado 0.
fim especifica o ponto final do círculo e é um número entre 0 e 1. Isto é opcional; se omitido, é usado 1.

Como, basicamente, precisamos apenas do raio e do centro do círculo, nós podemos começar com estes parâmetros.
Igual às outras funções gráficas, primeiro você tem de especificar as coordenadas (X,Y).
Neste caso, entretanto, você tem apenas um par de coordenadas para se preocupar. Elas indicam o centro do círculo. Você apenas indica o centro do círculo, usando o quadro do Apêndice E. Feito isto você tem de indicar o raio. O maior círculo que cabe na tela tem um centro em (128,96) e um raio de 95. Se o raio for maior do que 95, o círculo se achatará contra o canto da tela.
É hora de chamar de volta o programa "Linhas".

```
 5 PMODE 1,1
10 PCLS
20 SCREEN 1,1
25 LINE (0,0)-(255,191),PSET
30 LINE (0,191)-(255,0),PSET
40 GOTO 40
```

Apague a linha 25 e mude a linha 30 do programa para:

```
30 CIRCLE (120,96),95
```

Rode o programa. A sua TV deve mostrar um círculo alaranjado sobre um fundo cinza, meio torto. Porém você pensou que o COLOR BASIC faria um círculo redondo, verdadeiramente redondo. Observe a linha 5 do programa e você verá que está no PMODE 1 (resolução média). Mude o PMODE na linha 5 do programa para "4" (alta resolução):

```
 5 PMODE 4,1
10 PCLS
20 SCREEN 1,1
30 CIRCLE (128,96),95
40 GOTO 40
```

Agora rode-o. Isso é um círculo! (Deve ser um círculo cinza sobre um fundo preto.)

CÍRCULOS DE CORES DIFERENTES

Após decidir o raio do círculo, você pode escolher sua cor. Entretanto, no modo duas cores, você não tem muita opção, mas no modo 4 cores (PMODE 1 ou 3) você achará a opção de cor muito excitante. O seu programa ainda está assim:

```
 5 PMODE 1,1
10 PCLS
20 SCREEN 1,1
30 CIRCLE (128,96),95
40 GOTO 40
```

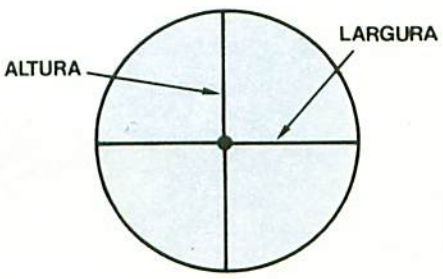
Primeiro, vamos fazer o círculo num tamanho mais flexível:

```
30 CIRCLE (128,96),30
```

Agora, para variar um pouco, mude a cor para ciano:

```
30 CIRCLE (128,96),30,6
```

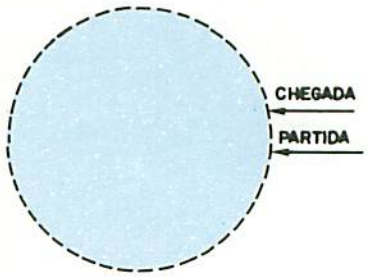
Viu como é fácil! Na verdade, você pode mudar a cor do círculo para quaisquer das cores disponíveis.



TRAÇANDO ELIPSES

Você já teve bambolê, bolinha de pingue-pongue ou pneu e tentou amassá-los com as mãos? Que forma eles adquiriram?
Você pode fazer o mesmo a um círculo se o Computador usar a opção da razão largura/altura (*hw*).

A largura do círculo será sempre a mesma, desde que você especifique o raio. A altura é determinada por *hw*. Se *hw* for maior do que 1, o círculo será "mais alto" do que "largo". Se *hw* for menor do que 1, ocorrerá o contrário. Por exemplo:




```
 5 PMODE 4,1
10 PCLS
20 SCREEN 1,1
30 CIRCLE (128,96),30,,1
40 GOTO 40
```

fornecerá um círculo correto. Porém, se você mudar *hw* para:

```
30 CIRCLE (128,96),30,,3
```

o círculo ficará mais alto do que largo. Se você mudar *hw* para:

```
30 CIRCLE (128,96),30,,.25
```

o círculo ficará mais largo do que alto. Se *hw* é igual a 0, o círculo torna-se "infinitamente" mais largo do que mais alto. Em outras palavras, ele torna-se uma linha horizontal. Quando *hw* aumenta além de 1, o círculo se aproxima de uma linha vertical.
Mude a linha 30 das seguintes maneiras e rode o programa:

```
30 CIRCLE (128,96),30,,0
```

E, depois:

```
30 CIRCLE (128,96),30,,100
```

Observe que nós omitimos o código de cor. Isto diz ao Computador para usar a cor de primeiro plano. Nós ainda temos de incluir a vírgula, entretanto.

AS VARIAÇÕES DO CÍRCULO

As últimas opções disponíveis com CIRCLE lhe permitem desenhar apenas uma parte de um círculo (um arco). Para utilizar esta opção, especifique o ponto onde deve começar o arco (qualquer número entre 0 e 1), coloque uma vírgula e indique onde é o fim (qualquer número entre 0 e 1).

O ponto inicial (0) para qualquer círculo é equivalente às 3 horas num relógio de parede. Uma vez iniciado o círculo, o desenho da sua construção ocorre no sentido horário a partir do ponto inicial. Por exemplo, se você quiser apenas um semicírculo, digamos do ponto 6 horas (.25) para o ponto 12 horas (.75), digite:

```
30 CIRCLE (128,96),30,1,1,.25,.75
```

Sempre que o ponto inicial é igual ou maior do que o ponto final, ou ainda quando ambos são omitidos, o Computador desenha um círculo completo.

PINTE COM PAINT

A função gráfica PAINT permite a você pintar qualquer figura com qualquer cor disponível. A sintaxe para PAINT é:

PAINT (x,y),c,b

x especifica uma coordenada X e é um número de 0 a 255.
y especifica uma coordenada Y e é um número de 0 a 191.
c especifica o código de cor e é uma expressão numérica de 0 a 8.
b especifica a cor de borda na qual a pintura deve parar e é um número de 0 a 8.

Os números dentro dos parênteses especificam onde deve começar a pintura. O primeiro parâmetro após o parêntese, c, especifica o código de cor da pintura. O parâmetro final, b, diz ao Computador a cor da borda na qual a pintura deve parar. Se o Computador alcança uma borda diferente daquela especificada pela cor, a pintura passará da borda. Vamos mudar o programa "Linhas" para:

```
 5 PMODE 3,1
10 PCLS
20 SCREEN 1,1
30 LINE (0,0)-(255,191),PSET
40 LINE (0,191)-(255,0),PSET
50 CIRCLE (129,96),90
60 PAINT (135,125),8,8
70 GOTO 70
```

Antes de rodar o programa, precavenha-se fazendo previsões sobre o que acontecerá. Se você olhar as linhas 30 e 40, verá as linhas concorrentes. A linha 50 gera um círculo cujo centro está no ponto onde se interseccionam as duas linhas. Até aí tudo bem, mas e sobre a PAINT na linha 60? Se você imagina que o Computador irá até a posição (135,125) da tela e pintará com laranja até a pintura atingir a borda laranja, você está absolutamente certo!

Retire a linha 30 do programa e rode-o. Veja como o Computador pinta o semicírculo após as bordas serem redefinidas.
A propósito, você observou qual PMODE e conjunto de cor você usou? PMODE 3 é um modo 4 cores e o conjunto de cor 1 lhe dá as cores cinza, ciano, roxa e alaranjada.
Permaneça no PMODE 3, mas altere o conjunto de cor (SCREEN 1,0) e rode o programa. Sem mudar outras linhas, você deve obter um círculo vermelho sobre um fundo verde. Para evitar confusão, mude também a cor na instrução PAINT:

```
60 PAINT (135,125),2,4
```

Agora quando você rodar o programa, metade do círculo será pintado de amarelo (código 2) até a borda vermelha (código 4).

Lembre-se sempre de que você só pode usar cores que estejam disponíveis no modo gráfico e conjunto de cor selecionados por PMODE e SCREEN.

Quando você especifica uma cor que não está disponível no conjunto utilizado, o Computador subtrai 4 dos códigos 5 a 8. O código zero é interpretado como 3.

COMO SE TRAÇA UM DESENHO

Até aqui nós mostramos como criar linhas, círculo e caixas. Nós ainda mostramos como enchê-los e pintá-los se necessário. Uma coisa que não fizemos, entretanto, foi mostrar como realmente "traçar" uma linha, caixa ou triângulo.
Sim, acredite ou não, existe um modo mais fácil de fazer algumas das coisas que nós mostramos a você. Este atalho é chamado DRAW, e lhe permite desenhar uma linha (ou séries delas) especificando sua direção, ângulo e cor — todos na mesma linha do programa.

DRAW linha

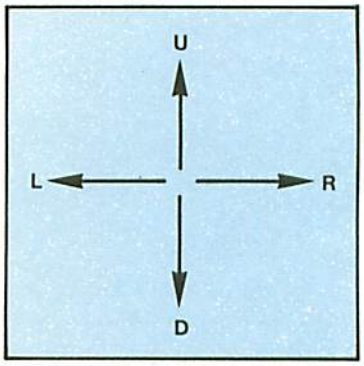
linha é uma string que pode incluir:

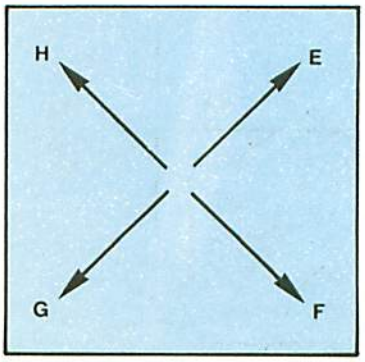
comandos de movimento
M = Move para a posição onde começa o desenho
U = Para cima
D = Para baixo
L = Para a esquerda
R = Para a direita
E = Ângulo de 45°
F = Ângulo de 135°
G = Ângulo de 225°
H = Ângulo de 315°
X = Executa uma substring e retorna

modos
C = Cor
A = Ângulo
S = Escala

opções
N = Sem atualizar a posição do traço
B = Espaço em branco (sem DRAW, apenas desloca)

Nota: *Se* linha *é uma string constante, ela deve vir entre aspas. Coloque sempre a opção B diretamente antes do comando de movimento M; caso contrário, podem aparecer linhas não desejadas.*





Com DRAW, simplesmente você tem de localizar o ponto inicial e dizer ao Computador qual a direção e quão longe a linha tem de ir. Ou, você pode omitir o ponto inicial e o Computador começará na última posição DRAW (ou no centro da tela se você ainda não tiver usado o DRAW no programa).
Vamos fazer uma tentativa usando o seu velho programa "Linhas". Retire as linhas 25 e 30 do programa e coloque a seguinte:

```
25 DRAW "BM128,96;U25;R25;D25;L25"
```

Nós separamos cada instrução de movimento com um ponto e vírgula (;), mas você não precisa fazê-lo — simplesmente torna o programa mais fácil de ler. Entretanto, você deve separar sempre as coordenadas (X,Y) com uma vírgula (,).

Pronto! Você pode imaginar por que o vértice inferior esquerdo está em (128,96)? Observe cuidadosamente os dois primeiros números entre aspas. O comando de movimento, M, posiciona a primeira ação do desenho no ponto especificado.

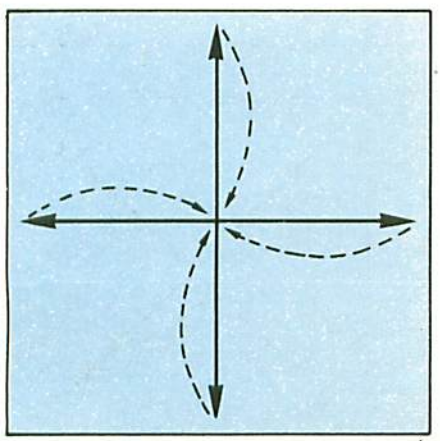
M x,y

x é a coordenada X e é uma expressão numérica de 0 a 255.
y é a coordenada Y e é uma expressão numérica de 0 a 191.

Nota: *M deve ser sempre precedido pela letra B, ou aparecerão linhas não desejadas.*

No programa acima nós instruímos o Computador para começar a ação do desenho em (128,96), desenhar (U) 25 pontos, para cima, mais 25 (R) para a direita, mais 25 (D) para baixo e finalmente 25 (L) para a esquerda.
Nota: *Se você não especificar o comprimento da linha, o Computador usará 1 como comprimento.*

VOCÊ PREFERE UM LOSANGO?



Ao invés de desenhar linhas horizontais e verticais, coloque o quadrado sobre um dos seus vértices. Para fazer isso, substitua U, R, L e D por E, F, G e H na linha 25 do programa:

```
25 DRAW "BM128,96;E25;F25;G25;H25"
```

Este desenho começa também em (128,96). Entretanto, ao invés de sair na horizontal, a primeira linha inclina 45 graus e as outras três são desenhadas nos seus ângulos deslocados.
Se você está no PMODE 0 ou 1 e usa E, F, G ou H para gerar uma linha que tem um comprimento de número ímpar e no mínimo uma coordenada (X,Y) ímpar, as linhas F e H terão um ligeiro "salto" no ponto médio; se ambas as coordenadas (X,Y) tiverem números pares, as linhas E e G terão o "salto". Isso é normal.

MOVIMENTOS ABSOLUTOS OU RELATIVOS?

Suponha que nós desenhamos um quadrado e então queremos desenhar um outro, próximo a esse. O problema é saber exatamente quão longe do primeiro deve estar o segundo quadrado, mas você não quer localizar as coordenadas (X,Y).
Há uma outra forma do comando M que lhe permite especificar o movimento "relativo" ao invés do "absoluto". Até aqui, nós desenhamos com movimento absoluto, significando que temos os pontos especificados em termos de suas coordenadas (X,Y) reais. Com movimento relativo, nós especificamos pontos em relação ao ponto *atual* (último desenho). A sintaxe do movimento relativo é:

M sinal desloc. x, desloc. y

desloc. x é um número especificando a distância para se deslocar da posição X atual. Se *desloc. x* é precedido por um sinal mais "+", a posição X é incrementada pela quantidade especificada. Se *desloc. x* é precedido por sinal de menos "–", a posição X é decrementada pela quantidade especificada. Quando se deve dizer para o Computador usar movimento relativo não absoluto, deve ser usado *um sinal de menos ou de mais*.
desloc. y é um número especificando a distância para se deslocar da posição Y atual. O sinal do número determina se a posição atual é aumentada (+) ou diminuída (–). Para deslocamentos positivos, o sinal pode ser omitido (apenas de *desloc. y*).

Movimento absoluto: "Vá para a esquina da Rua Alvorada com Casa do Ator". Movimento relativo: "Desça duas quadras e vire a direita".

Por exemplo, se você desejar criar um outro quadrado relativo à posição da caixa original em nosso programa "Linhas" (modificado), você pode acrescentar esta linha:

```
30 DRAW "BM+15,15;U25;R25;D25;L25"
```

Quando o Computador executa a linha 30, a posição DRAW atual está em (128,96) (a última posição DRAW na linha 25). Assim, o vértice inferior esquerdo do novo quadrado está localizado em (X = 128 + 15 = 143, Y = 96 + 15 = 111).
Se você muda a linha 30 para:

```
30 DRAW "BM+15,-15;U25;R25;D25;L25"
```

e roda o programa, o ponto inicial do novo quadrado será X = 128 + 15 = 143, Y = 96 – 15 = 81.

CUMPRA A ESCALA

Até agora, desenhamos linhas, quadrados e cubos. Mas, o que acontece se os quadrados se tornam muito grandes ou muito pequenos? Bem, o seu Computador tem uma função embutida que lhe permite ampliar (ou reduzir) qualquer figura gerada por DRAW. Tudo o que você tem a fazer é colocar o seguinte na string:

Sx

x é um número de 1 a 62 e indica o fator de escala em unidades de 1/4

1 = 1/4 de escala
2 = 2/4 de escala
3 = 3/4 de escala
4 = 4/4 de escala (original)
5 = 5/4 de escala (125%)
8 = 8/4 de escala (dobro)
12 = 12/4 de escala (triplo)
etc.

Nota: *Se você não especificar um número para S, o Computador usará S4 (4/4 = 1).*
Após um comando Sx, todos os movimentos relativos e absolutos serão representados de acordo até o próximo comando Sx.

Ao usar a escala de redução, se a linha resultante não é um número inteiro, o Computador arredondará o comprimento da linha para o número inteiro mais próximo. Por exemplo, "S2U25R25D25L25" resultaria num quadrado 12.5 × 12.5. O Computador desenha um quadrado de 13 × 13.

Sim, é tempo de se refinar o programa "Linhas" uma vez mais, mas vamos fazê-lo desenhar um único quadrado dessa vez. Retire a linha 30 do programa e mude a linha 25:

```
25 DRAW "S2;BM128,96;U25;R25;D25;L25"
```

Rode o programa. Você deve ter um quadrado com um vértice inferior esquerdo em (128,96) mas na metade do tamanho que você especificou. Rode o programa seguinte para ver como um pequeno quadrado pode ser reduzido e aumentado para encher toda a tela.

```
 5 PMODE 4,1
10 PCLS
20 SCREEN 1,1
25 FOR SCALE = 1 TO 62
30 S$ = "S" + STR$(SCALE)+ ";"
35 DRAW S$ + "BM10,10OU20R20D20L20"
40 NEXT SCALE
50 GOTO 50
```

Não pense que o menor quadrado é aquele especificado na linha 35. O que nós especificamos é o quarto da série (S4).

Você não usou muitas strings e funções string nesta seção. Para uma rápida revisão, volte à seção anterior. Nós voltaremos às strings no Capítulo 16.

PARÂMETRO Cx (COR)

A opção C de DRAW lhe permite especificar a cor de uma linha particular. Primeiro, vamos fazer uma atualização da listagem do programa "Linhas":

```
 5 PMODE 3,1
10 PCLS
20 SCREEN 1,1
30 DRAW "S2;BM128,96;U25;R25;D25;L25"
40 GOTO 40
```

Seu programa agora está assim:

```
 5 PMODE 3,1
10 PCLS
20 SCREEN 1,1
30 DRAW "C6;BM128,96;U25;
   R25;D25;L25"
40 GOTO 40
```

Volte para a escala normal (mude S2 para S4 ou retire S2), e dentro do primeiro conjunto de dois pontos na linha 30, coloque:

```
C6
```

Rode-o. Você obteve um quadrado ciano sobre um fundo cinza? Agora substitua o C6 (na linha 30) por C8 e rode o programa. O quadrado ficou laranja?
C deve ter o seguinte formato:

Cx
x é uma expressão numérica de 0 a 8 e especifica um dos 9 códigos de cores. É opcional, se omitido, é usada a cor de primeiro plano.

Você pode usar Cx em qualquer lugar dentro da instrução DRAW e todas as ações seguintes serão da cor que você especificou. Por exemplo, mude a linha 30 do programa para:

```
30 DRAW "C8; BM128,96;U25;R25; C6; D25;L25"
```

Se você quiser apagar uma linha, desenhe uma outra sobre a mesma usando a cor de fundo.

Rode-o. Você deve ter um quadrado de duas cores sobre a sua tela. As duas primeiras linhas devem ser laranja, mas quando a linha começa a descer (antes de D25,), a linha deve tornar-se ciano.

A INCLINAÇÃO DO A

Uma outra opção disponível é o A. Permite que você especifique o ângulo que a linha deve ser desenhada. Após a opção ter sido incluída no comando DRAW, todas as linhas subseqüentes serão desenhadas com um deslocamento angular especificado por Ax.

Ax

x é uma expressão numérica de 0 a 3 e especifica um dos seguintes ângulos:

0 = 0 graus
1 = 90 graus (sentido horário)
2 = 180 graus (sentido horário)
3 = 270 graus (sentido horário)

Nota: *Se você não especificar um ângulo, o Computador usa A0.*

Para ilustrar isso, mude a linha 30 do programa para:

```
30 DRAW "A0;BM128,96;U25"
```

Rode-o. Sua tela deve mostrar uma linha (comprimento de 25 pontos) reta. Agora mude a linha para:

```
30 DRAW "A1;BM128,96;U25"
```

Rode o programa. A linha deve indicar agora para a direita. (A diferença no comprimento da linha é porque os "pontos" no PMODE 3 são retângulos mais altos do que largos.)

COMO COLOCAR BRANCOS

Suponha que você queira desenhar uma linha em branco ou invisível. Você pode fazer isto com a opção B, que especifica que a próxima linha (e somente a próxima linha) deve ser em branco. Por exemplo, vamos dizer que você está desenhando letras do alfabeto e pronto para desenhar a letra "C". Ela nada mais é do que um quadrado com um lado em branco. Mude o seu programa para gerar um quadrado, mas deixe o lado direito em branco (gerado por D25):

```
30 DRAW "BM128,96;U25;R25 ;B ;D25;L25"
```

Então rode o programa. Veja o que nós queremos dizer. Lembre-se, apenas, que a linha imediatamente seguinte à B estará em branco.

O QUÊ?! MAIS OPÇÕES?

Ainda uma outra propriedade de DRAW é N, a opção "sem atualizar". N diz ao Computador para retornar à posição atual após desenhar a linha seguinte (linha 2). Vamos usar a opção numa linha de programa, e, então, discuti-la. Mude a linha 30 do programa para:

```
30 DRAW "M128,96; N; U25; N; R25; N; D25; N; L25;"
```

Rode-o. Deve haver 4 linhas saindo do centro da tela, cada uma em quatro direções diferentes (para cima, direita, para baixo e esquerda).
O que acontece é que você diz ao Computador para desenhar uma linha começando em (128,96), que soma 25 pontos, mas deve retornar a sua posição original antes de iniciar uma outra linha.
O Computador então desenhou uma linha de 25 pontos para a direita e retornou a sua posição original. Em seguida, ele desenhou 25 pontos para baixo e novamente voltou à posição original. Finalmente, o Computador desenhou uma linha de 25 pontos para a esquerda e, uma vez mais, retornou à posição original.

CONSTANTES STRING X VARIÁVEIS STRING

Provavelmente você pensa que nós estávamos enganando você quando dissemos que a string após a instrução DRAW pode ser uma constante ou uma variável. Se você deseja usar uma variável string, apenas identifique-a tal como na linha de programa que precede DRAW e use a variável string no lugar dos elementos entre aspas de DRAW. Por exemplo, mude seu programa para:

```
25 A$ = "BM128,96;C8;U25;R25;D25;L25"
30 DRAW A$
```

Agora rode-o. O programa deve gerar uma caixa laranja (25 × 25), cujo canto inferior esquerdo está no centro da tela. O COLOR BASIC oferece uma variação disto, entretanto. Nós a intitulamos "a opção de execução (X)". Enquanto você está executando uma rotina DRAW, a opção de execução lhe permite executar uma outra string DRAW, então retornar e completar a primeira operação. Mude a linha 30 do programa para:

```
30 DRAW "BM95,50;U25;R25; XA$; D25;L25"
```

Rode-o. O Computador deve começar uma linha iniciando em (95,50) que sobe, então vira à direita e desenha uma outra linha. Neste ponto, é executada A$ e um quadrado de 25 × 25 é desenhado, começando em (128,96). Após a execução de A$, o Computador retorna à string original e completa a sua execução.
Nota: *A posição do traço não retorna à sua posição anterior (por exemplo, o primeiro quadrado). Ela permanece onde estava após completar A$.*

ARMAZENANDO FIGURAS

Mesmo com várias mudanças feitas no programa "Linhas", você pode pensar que ainda não está qualificado para usar todos os recursos disponíveis. Entretanto, você pensará diferente quando ver o que o Computador faz com GET e PUT.
Estas funções permitem que seu Computador disponha de uma área retangular que contém uma figura qualquer, armazene-a numa matriz, então a pega de volta e a põe na tela mais tarde. Quando você simula movimento, as instruções GET e PUT podem mover um objeto mais rápido do que qualquer outro método que você já tem com seu CP 400 COLOR. As sintaxes para GET e PUT são:

O quê? As matrizes (ou arranjos) foram explicados no Capítulo 10; assim, se você está um pouco esquecido sobre o que é uma matriz volte e refresque sua memória antes de ir adiante.

GET ponto inicial, ponto final, destino, G

ponto inicial é a coordenada (x1,y1) do canto superior esquerdo de um retângulo na tela.
ponto final é a coordenada (x2,y2) do canto inferior direito do mesmo retângulo.
destino é o nome de uma matriz predefinida que armazenará o conteúdo do retângulo.
G diz ao Computador Colorido para armazenar os conteúdos do retângulo com detalhe gráfico completo. Isso é opcional para alguns
G diz ao CP 400 COLOR para armazenar os conteúdos do retângulo com detalhe gráfico completo. Isso é opcional para alguns usos.

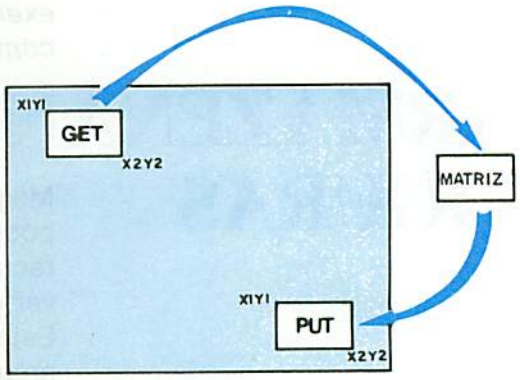
PUT ponto inicial, ponto final, fonte, ação

ponto inicial e final são explicados acima.
fonte é o nome de uma matriz predeterminada que contém os dados a serem gravados no retângulo mostrado.
ação determina como os dados são gravados no retângulo:
PSET ativa cada ponto que está na matriz fonte.
PRESET desativa cada ponto que está na matriz fonte.
AND, OR, NOT (Veja abaixo.)
ação é opcional sob algumas condições.

Nota importante: *Assegure-se de que você está no mesmo PMODE para GET e para PUT. Caso contrário, você pode não "pegar" o que você "colocou".*

O programa abaixo lhe dará uma idéia do que faz GET/PUT. Mais tarde, nós conversaremos sobre como funcionam estas funções. Introduza-o e rode.

```
 5 PCLEAR 4
10 PMODE 3,1
15 PCLS
20 SCREEN 1,1
25 DIM V(20,20)
30 CIRCLE (20,20),10
35 GET (10,10)-(30,30),V
40 PCLS
42 FOR DLAY = 1 TO 300: NEXT DLAY
45 PUT (110,110)-(130,130),V
50 FOR DLAY = 1 TO 300: NEXT DLAY
60 GOTO 60
```



Deve aparecer um círculo laranja no canto superior da tela (posição (10,10) — linha 30 do programa). Após um ou dois segundos, ele deve desaparecer; de repente reaparece em outro lugar da tela (posição (110,110) — linha 45 do programa).
O que o Computador faz...

- Cria uma matriz (na linha 25).
- Cria um círculo (na linha 30).
- Obtém o círculo que estava dentro do retângulo fonte (especificado na instrução GET) e o armazena na matriz (linha 35).
- Limpa a tela (linha 40).
- Coloca o círculo do arranjo no retângulo de destino especificado na instrução PUT (linha 45).

DESCRIÇÃO DE UM RETÂNGULO

A posição e tamanho do retângulo mostrado são determinados pelas duas coordenadas (X,Y) usadas em GET/PUT:

Posição	Canto superior esquerdo = ponto inicial = (x1, y1) Canto inferior direito = ponto final = (x2,y2)
Tamanho	Largura = x2 − x1 Comprimento = y2 − y1

COMO ESCOLHER A MATRIZ CORRETA

Observe que antes de usarmos GET e PUT, nós temos de criar um arranjo bidimensional para armazenar (linha 25). Como nós determinamos o tamanho de cada dimensão da matriz? O tamanho dela deve se ajustar ao retângulo mostrado.
A primeira dimensão da matriz = largura do retângulo.
A segunda dimensão da matriz = comprimento do retângulo.
Por exemplo, um retângulo 40 x 20 requer uma matriz 40 x 20. Como o COLOR BASIC tem um elemento zero, a seguinte instrução de dimensão cria um arranjo adequado:

```
DIM V(39,19)
```

QUAL O TAMANHO MÁXIMO DE UM RETÂNGULO?

O tamanho do retângulo é limitado pela quantidade de memória disponível para uso pelo seu programa. Cada elemento da matriz armazenada ocupa 5 posições de memória (bytes). Assim, na prática, os sistemas de 16 k de RAM limitam-se a matrizes com pouco menos de 1 400 elementos (ou seja, comprimento × largura < 1 400). Na verdade, se seu programa é muito extenso, você pode ter de usar uma matriz menor do que essa. Na verdade, o retângulo deve ser grande o suficiente também para guardar os gráficos e sua posição na tela deve ser escolhida para incluir todos os pontos que você quer obter.
Se você usa a opção GET, G, você deve usar uma das opções disponíveis com PUT (PSET, PRESET, AND, OR, NOT); caso contrário, aparecerão resíduos quando você colocar o retângulo de volta sobre a tela. Nem sempre é necessário usar as opções.

- No PMODE 0, 1 ou 3, não use as opções.
- No PMODE 4 ou 2, use as opções.

As opções PUT realizam as funções como mostrado na tabela da página seguinte.

EXPANDINDO-SE

Observe como o próximo programa usa as instruções GET e PUT para simular movimento. Para acelerar a ação, você pode aumentar o incremento STEP. Preste atenção particular ao modo como o programa usa as funções que você já conhece (LINE, CIRCLE, PAINT etc.).

```
 5 PCLEAR 4
10 DIM V(30,30)
15 PMODE 3,1
20 PCLS
25 SCREEN 1,1
30 CIRCLE (128,96),30
35 PAINT (128,95),2,4
40 PAINT (128,97),3,4
45 GET (98,81)-(128,111),V,G
50 PCLS
55 FOR I = 150 TO 1 STEP -1
60 PUT (I,81-I/5)-(I + 60,111-I/5),V,PSET
65 NEXT I
70 GOTO 70
```

O único truque real que fizemos foi mudar (atualizar) a posição do retângulo pelo valor de I.

Opção	Função
PSET	Ativa os pontos que estão colocados no retângulo original.
PRESET	Desativa os pontos que estão colocados no retângulo original.
AND	Um operador lógico. Compara os pontos armazenados no retângulo original com o retângulo de destino. Se ambos são iguais, então, o ponto da tela será ativado; senão, o ponto da tela será desativado.
OR	Um operador lógico. Compara os pontos (como acima). Se está ativado, o ponto permanece ativado.
NOT	Um operador lógico. Converte o estado de cada ponto no retângulo de destino independente dos conteúdos do arranjo PUT.

EXERCÍCIO DO CAPÍTULO

Agora você pode começar a pintar sua casa. Peque o programa do Capítulo 11 e aprimore-o.

CAPÍTULO

14

toque outra vez

Os seus olhos já estão ficando embaçados de tanto ver pontos, linhas, quadrados, triângulos, círculos e pinturas?
Nesse caso, é hora de darmos a seus olhos um descanso e passarmos a atacar seus ouvidos com a função PLAY. Ela permite não apenas reproduzir música, mas também compô-la.
Antes de mais nada, devemos observar que PLAY não é uma função gráfica. Conseqüentemente, você não terá de começar seus programas com PMODE, PCLS ou SCREEN. Portanto, vamos pôr isso tudo de lado e ouvir um pouco de música.

OUÇA COM ATENÇÃO

PLAY música

música *é uma expressão string que especifica:*
nota — *Uma letra de "A" a "G" ou um número de 1 a 12.*
oitava — *A letra O seguida por número de 1 a 5. Se omitido, é usada a oitava 2.*
comprimento de nota — *L seguido de um número de 1 a 255. Se omitido, é mantido o comprimento atual.*
ritmo — *T seguido por um número de 1 a 255. Se omitido, é usado T2.*
volume — *V seguido por um número de 1 a 31. Se omitido, é usado V15.*
comprimento da pausa — *P seguido de um número de 1 a 255.*
execução de substrings — *As substrings devem ser precedidas por um prefixo X e seguidas por um ponto e vírgula, por exemplo: XA$;*

AS NOTAS MUSICAIS

Obviamente, você não pode ter música sem notas musicais. PLAY lhe dá dois modos para especificar a nota exata que você precisa.
A primeira (e provavelmente a mais fácil) é tocar a nota que você quer, introduzindo uma das notas musicais padrão — "A, B, C, D, E, F ou G".
Os semitons, sustenido e bemol, são indicados usando-se "+" ou "#" para sustenidos e "–" para bemóis.

Por exemplo: "A" representa o Lá; "A#", Lá sustenido; e "A–", Lá bemol.
Introduza o seguinte para ver o que isto significa:

```
PLAY "A"
```

Para ouvir a mudança que um bemol ou sustenido pode proporcionar, digite:

```
PLAY "A;A#"
PLAY "A–;A;A#;A;A–"
```

Você pode fazer o mesmo com todas as 7 notas (A-G) da escala. **(Nota:** há apenas meio-tom separando B e C, E e F. Assim B# = C,B = C– e E# = F,E = F–.)

NOTAS E NÚMEROS

Um outro modo de especificar uma nota musical é usando um número entre 1 e 12, precedido pela letra N (N é opcional; se omitido, o número sozinho indicará a nota).
Os números de 1 a 12 representam cada nota da escala musical, incluindo sustenidos e bemóis. Isto é uma notação mais concisa; embora seja também mais difícil de ler se você já se acostumou à notação padrão.

Nota: *PLAY não reconhece a notação "C–" ou "B#". Use os números de 1 a 12, respectivamente, ou substitua "C" por "B#" e "C–" por "B".*
Você obterá a nota que quiser, já que B# = C e C– = B.

Rode o seguinte programa para ouvir a escala completa de 12 tons. (Daqui por diante, nós chamaremos isso de programa "Escala".)

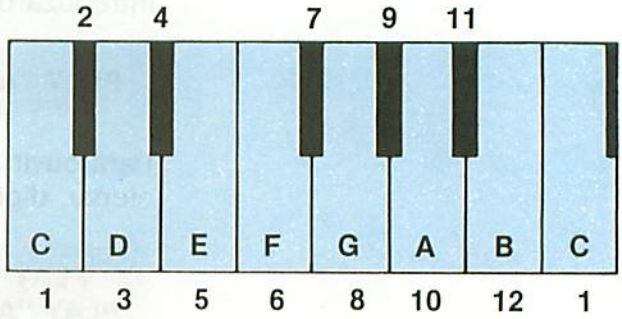
```
 5 CLS
10 FOR N = 1 TO 12 'N = NOTA
15 PRINT "NOTA #"; N
20 PLAY STR$(N)
30 NEXT N
```

Acrescente um atraso no programa para que você possa comparar os números às notas:

```
25 FOR I = 1 TO 500: NEXT I
```

A tabela a seguir coloca as notas com seus respectivos códigos.

Número	Nota	Número	Nota
1	C	9	G#/A–
2	C#/D–	10	A
3	D/E–	11	A#/B–
4	E–/D#	12	B
5	E/F–		
6	F/E#		
7	F#/G–		
8	G		



MUDANDO O TEMPO DAS NOTAS

As notas no programa "Escala" duram 1/4 do tempo padrão. Como nós não especificamos o comprimento da nota (isto é, se é inteira, meia, quarto de nota etc.), o Computador usou automaticamente quarto de nota, que é a duração inicial.
Para escolher a duração da nota, use L seguido por um número entre 1 e 255. Por exemplo: o número 1 identifica a nota inteira; 2, a meia nota; 4, o quarto de nota; 8, um oitavo; 16, 1/16 etc.
Na verdade, você pode usar qualquer número de 1 a 255. (Você já ouviu falar sobre 1/255 de nota?)
Vamos variar as durações das notas para tentar perceber, na prática, como isso funciona. Digite esta linha e aperte ENTER:

```
PLAY "L2;A;L4;A;L2;A"
```

TABELA DE COMPRIMENTO DE NOTAS

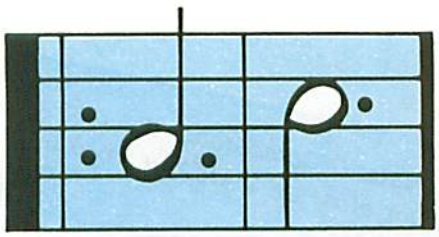
Numero L	Comprimento de nota
L1	Nota inteira
L2	Meia nota
L3	Três quartos de nota
L4	Um quarto de nota
L8	Um oitavo de nota
L16	1/16 de nota
L32	1/32 de nota
L64	1/64 de nota
.	.
.	.
.	.
L255	1/255 de nota

L2 indica meia nota; L4, um quarto de nota. Assim nós tocamos como segue: meia, quarto e meia novamente.

```
PLAY "L1; A;A#;A-" ENTER
```

Mais uma coisa sobre a opção L: não precisa ser repetida para cada nota que é tocada. PLAY usará o valor atual da nota até que você lhe peça para trocar por um novo valor. Isso explica porque não precisamos repetir L1 no último exemplo. Na verdade, a maioria das opções PLAY que aparecerá neste capítulo usará o valor "atual".

DURAÇÃO INTERMEDIÁRIA



Você sabe que há outras durações além das meia nota, um quarto de nota etc. Por exemplo, há um quarto de nota pingado, que equivale a 3/8 de nota.
Você pode tocar essa nota acrescentando um ponto (.) ou uma série de pontos (...) ao número após o L. Cada ponto que você acrescenta (e você pode acrescentar quanto quiser), aumenta o comprimento da nota de metade do seu valor normal. Por exemplo:

```
L4. = 1/4 + 1/8 = 3/8 de nota
```

Experimente isto:

```
PLAY "L4.;A;L8;C;L4.;E;L8;C;E;C;E;C;L4;A"
```

MAIS GRAVE OU MAIS AGUDO

A oitava que estamos usando (#2) soa bem, mas, é como tocar com um piano de 12 teclas. Para a boa música, precisamos muito mais que isso.
A letra O é a opção PLAY que permite a você selecionar outras oitavas. Para alterar a oitava use o O seguido de um número entre 1 e 5 (qualquer número, que não estes, resultará num erro).

Seu Computador usa automaticamente a segunda oitava se você não especificar qual oitava você quer. Vamos tentar tocar essa oitava.

```
PLAY "CDEFGAB"
```

O que houve? Correu tudo bem até a quinta nota, o Sol, que corresponde ao G. A próxima nota, o Lá, voltou ao começo da mesma escala. Mas nós queremos sair desta escala, não é? Então tente isto:

```
PLAY "CDEFG;O3;AB"
```

FALE MAIS ALTO!

Certo, você pode ajustar o volume de sua música apenas controlando o volume da TV. Mas você não precisa ficar sentado em frente à TV regulando o volume conforme a música. Especialmente, quando o Computador pode fazê-lo por você.
O seu Computador faz isso com a opção V (volume). Tudo o que você precisa usar é o V seguido por um número entre 0 e 31. Se você não especificar o valor de V, seu Computador automaticamente usa V15.
Ajuste o volume de sua TV para um nível normal e rode este pequeno programa:

Nota: O Computador usa o valor atual de V até que você o altere.

```
 5 CLS
10 PLAY "V5;C;V10;C;V20;C;V30;C;"
20 GOTO 10
```

Aperte BREAK para sair do ciclo, antes que você ganhe uma dor de cabeça daquelas.

ENFIM, UM MOMENTO DE SILÊNCIO

Pode ser que o último programa seja mais fácil de se ouvir se as notas não forem tocadas todas juntas. Vamos usar a opção de pausa (P) para alguns momentos de silêncio entre as notas e ver se fica melhor.

Para colocar uma pausa entre as notas, use P seguido de um número entre 1 e 255. Os números correspondem àqueles usados com L (comprimento de nota) com uma diferença importante — os pontos não podem ser usados com P. Para compensar, apenas digite uma série de pausas. Por exemplo, para obter uma pausa de 3/8, digite P4P8.
Mude a linha no último programa para:

```
10 PLAY "V5;C; P2; V10;C; P2; V20;C; P2; V30;C; P2"
```

Realmente uma pausa de meia nota (P2) entre todos esses Dós não melhora o som, mas já dá uma idéia de como funciona o P.

Nós deixamos espaços entre cada combinação volume/nota, de modo que você possa ler a linha sem dificuldade. Não são necessários os espaços.

ENTRANDO NO RITMO

Nesse momento, nosso programa de teste deve estar assim:

```
 5 CLS
10 PLAY "V5;C;P2; V10;C;P2; V20;C;P2; V30;C;P2"
20 GOTO 10
```

É passável, mas não agradável, principalmente e porque o ritmo (velocidade) está um pouco lento.
Você pode aumentar (ou diminuir) o ritmo com T e um número entre 1 e 255. Se você não especificar o número para T, seu Computador usará automaticamente T2. Inicie reduzindo o ritmo de seu programa:

```
10 PLAY "T1; V5;C;P2; V10;C;P2; V20;C;P2; V30;C;P2"
```

Acelere-o mudando T1 para T15. E então? Melhorou?
E se você acelerar ao máximo, com 255? Isso não demora muito, demora?

EXECUTANDO A SUBSTRING

Lembra-se como você podia executar uma substring usando a opção de execução (X) com DRAW? PLAY tem uma propriedade similar que lhe permite executar uma substring, e então voltar à string original e terminá-la.
A função de execução toma a seguinte forma:

```
XA$;
```

A$ contém uma seqüência de comandos e funções normais. X diz ao Computador para PLAY A$. O ponto e vírgula após A$ é necessário.
Arranje novamente nosso programa de demonstração de modo que ele execute uma substring:

```
 5 CLS
10 A$ = "A;A#;A-"
20 B$ = "O5;XA$;"
30 C$ = "O1;XA$;XB$;"
40 PLAY C$
```

Rode o programa e siga a execução.

Nota: *Um ponto e vírgula (;) deve seguir o sinal de dólar ($) sempre que você utiliza a função de execução. Nesse exemplo, você pode retirar todos os pontos e vírgulas, exceto aqueles seguintes ao dólar ($).*

UMA NOTA A MAIS

Não, nós não forçaremos uma nova nota, como H ou J, para você. Nós temos um jeito melhor para você usar algumas das opções que estamos descrevendo. Com O (oitava), V (volume), T (ritmo) e L (comprimento da nota), você pode usar um dos seguintes sufixos *em vez de* adicionar um número. Nós ilustraremos estas características usando um programa simples:

```
 5 CLS
10 PLAY "T2"
20 PLAY "A;A#;A-"
30 GOTO 20
```

Note que estabelecemos o valor T na linha 10. Rode o programa somente uma vez apenas para identificar o som. Agora insira T+ na linha 20.

```
20 PLAY "T+;A;A#;A-"
```

Rode-o. O sinal + automaticamente acrescenta 1 ao valor de T, cada vez que a linha 20 é executada. De uma lenta partida você pode alcançar um vôo.

Verifique o ritmo com o menos (–) rodando este programa:

```
 5 CLS
10 PLAY "T255"
20 PLAY "T–; A;A#;A–"
30 GOTO 20
```

Depois de uma rápida partida o Computador finalmente começa a diminuir o ritmo.
Não é a multiplicação mais rápida que a adição? Na linha 10, ajuste o ritmo para 2, troque o T na linha 20 para T> e deixe-o rodando.

SUFIXO	PROPÓSITO
+	Adiciona 1 ao valor atual.
–	Subtrai 1 do valor atual.
>	Multiplica o valor atual por 2.
<	Divide o valor atual por 2.

*Há uma fórmula matemática para calcular o comprimento da nota. Entretanto, talvez você não a use com muita freqüência. É assim: valor do comprimento da nota + (valor do comprimento da nota * número de pontos)/2.*

```
10 PLAY "T2"
20 PLAY "T>; A;A#;A–"
```

Você começou com T2, certo? O Computador multiplica este valor por 2 tornando-o 4, 4 × 2 para 8, 8 × 2 para 16, e assim até alcançar 255. Você pode também diminuir o ritmo rapidamente pela divisão do ritmo atual por 2 usando "<".

```
10 PLAY "T255"
20 PLAY "T<; A;A#;A–"
```

Lembre-se de que você pode fazer a mesma coisa para aumentar ou diminuir o comprimento da nota, o volume e a oitava.

UM MOMENTO DE DESCANSO

Depois de tanta informação envolvendo retas, círculos, pontos, oitavas, bemóis, semitons e outras coisas, você deve estar "meio" esgotado. Pois então, num último esforço, introduza o programa abaixo, pegue seu livro preferido (não aquele que fala de computadores, é claro!), acomode-se no melhor sofá e fique a ouvir Strauss! *Danúbio Azul*, para ser exato.

```
10 REM DANUBIO AZUL PARA MICRO E ORQUESTRA
20 A$ = "L8;DF#;O3;A;L4;A;O4;L8;AA;O3;P8;F#F#;"
   :B$ = "P8;O2;DDF#;O3;A;L4;A;L8;O4;AA;P8;GG;"
30 C$ = "P8;O2;C#C#E;O3;B;L4;B;L8;O4;BB;O3;P8;GG;"
   :D$ = "P8;O2;C#C#E;O3;B;L4;B;L8;O4;BB;O3;P8;F#F#;"
40 E$ = "P8;O2;DDF#;O3;A;L4;D;L8;O4;DD;P8;AA;":
   F$ = "P8;O2;DDF#;O3;A;L4;D;L8;O4;DD;P8;BB;"
50 G$ = "P8;O3;EEG;L16;O4;B;P16;L2;B;":H$ = "L8;O3;
   G#;O4;A;L2;F#;L8;D;O3;F#;L4;F#;L8;E;L4;O4;B;L8;A;O3
   ;D;P16;L16;D;L8;D"
200 CLS 3
210 PRINT@233,"DANUBIO AZUL";
220 PLAY "XA$;XB$;XC$;XD$;XE$;XF$;XG$;XH$;"
```

EXERCÍCIO DO CAPÍTULO

Aproveitando Strauss, vamos repetir *Danúbio Azul* várias vezes e cada vez mais rápido. Sugestão: use T+.

CAPÍTULO

15

o jogo dos números

Além das funções matemáticas descritas na Seção II deste livro você tem agora em seu repertório várias funções avançadas. Nós daremos a você um breve resumo de cada função e mostraremos como usá-la para desenvolver qualquer operação matemática de alto nível. Antes disso, entretanto, há algumas funções e definições que você deve conhecer.

QUADRADOS, CUBOS & POTÊNCIAS

Responda rápido! Qual é o quadrado de 1,5? E o cubo de 77? Se você não sabe, pergunte ao seu Computador. Quando quiser elevar um número a enésima potência, siga este formato:

NÚMERO ↑ POTÊNCIA

número especifica o número que você quer elevar à potência. Ele pode ser qualquer expressão numérica.
↑ é gerado pressionando ⬆
potência é o expoente ao qual o número será elevado. Ele pode ser qualquer expressão numérica.

No Apêndice H mostramos algumas fórmulas matemáticas (geometria plana, trigonometria e álgebra) e suas instruções BASIC equivalentes.
Vamos resolver o 77 ao cubo. Seguindo seu formato, teremos na tela o seguinte:

```
PRINT 77 ↑ 3
456533.002
OK
```

Não se preocupe com o ".002". Este é chamado um "erro de arredondamento" e é inevitável (é triste dizer), porque o Computador não é uma calculadora perfeita. Mas até aí, nenhuma máquina é.

Tente elevar 10 à 10ª potência. Sua tela exibiu:

```
1.00000001E+10
```

Já que 10 000 000 000 tem mais que 9 dígitos significativos, o Computador indica na notação E. Isso já foi explicado na Seção II. E 100 à 100ª potência? Você obteve um ?OV ERRO (estouro). Isto significa que a resposta é muito grande para o Computador manipular. Seu Computador foi projetado para manipular qualquer número entre -10^{38} a $+10^{38}$. Qualquer número fora dessa faixa resultará num erro de estouro.

RAIZ QUADRADA

SQR habilita você a encontrar a raiz quadrada de um número.

SQR (número)

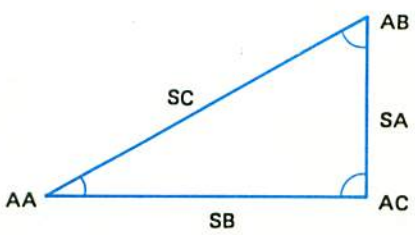
número é uma expressão numérica nunca menor que 0.

Por exemplo, se você quer a raiz quadrada de 100, digite:

```
PRINT SQR (100) ENTER
```

e você encontrará como resposta o 10.

VOCÊ PEGA TRÊS LADOS E...



É mesmo uma receita de triângulo, você tem que aprender, pois vai usá-la durante todo o tempo que discutirmos as funções trigonométricas; assim tente se familiarizar muito bem com ela.
Você verá que a trigonometria tem suas aplicações práticas. Por exemplo, imagine que o triângulo que você está trabalhando neste capítulo é o telhado de uma casa, que você está construindo. Essas funções podem ajudar você a determinar o comprimento da viga ou a inclinação do telhado. Assim, a matemática permite construir coisas e este capítulo pode acrescentar o conhecimento que você estava precisando.
Note que usamos para os ângulos o prefixo A (ângulo A = AA etc.). Para um lado oposto a um certo ângulo, usamos a letra S

como prefixo mais a letra deste ângulo (por exemplo, lado SA é oposto ao ângulo AA).
No nosso caso, o triângulo é retângulo sendo AC o ângulo reto. Em relação ao ângulo AA, na trigonometria, SB é chamado "adjacente"; o lado SA, "oposto"; e o lado SC, "hipotenusa". Usando nosso triângulo, nós podemos definir as funções de um triângulo comum da seguinte maneira:
seno de AA = SIN(AA) = oposto/hipotenusa = SA/SC
co-seno de AA = COS(AA) = adjacente/hipotenusa = SB/SC
tangente de AA = TAN(AA) = oposto/adjacente = SA/SB

GRAUS X RADIANOS

Para definir um ângulo, você pode usar 2 unidades de medida. A unidade mais comum é grau, a outra, "mais técnica", é o radiano. Seu Computador assume que todos os ângulos são medidos em radianos. Podemos converter os ângulos de radianos para graus (e vice-versa) como mostrado no quadro ao lado.

Graus para radianos:
graus/57.29577951
Radianos para graus:
radianos/57.29577951

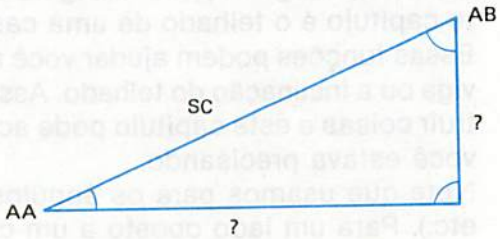
No programa exemplo deste capítulo, nós incluímos um "conversor". Este permite entrar com graus, e o Computador automaticamente converte para radianos.

O SENO (SIN)

SIN (ângulo)

ângulo é uma expressão numérica para um ângulo em radianos.

Um uso para SIN é determinar os 2 lados desconhecidos de um triângulo, se 2 ângulos e o 3.º lado são conhecidos.
Introduza e rode o programa seguinte. Você pode entrar qualquer valor que você quiser.



```
 5 CLS
10 INPUT "QUAL O ANGULO A (AA)";AA:IF AA<=0 OR
   AA>=180 THEN 100
20 INPUT "QUAL O ANGULO B (AB)";AB:IF AB<=0 OR
   AB>=180 THEN 100
30 INPUT "QUAL O LADO C (SC)";C:IF C<=0 THEN 100
40 AC=180-(AA+AB) 'VALOR DO ANGULO AC
50 IF (AA+AB+AC)<>180 THEN 100 'SOMA DOS ANGULOS
   DE UM TRIANGULO=180 GRAUS
60 AA=AA/57.29577951: AB=AB/57.29577951:
   AC=AC/57.29577951' CONVERTE GRAUS EM RADIANOS
70 SA=((SIN(AA))/(SIN(AC)))*SC:IF SA<0 THEN 100
80 SB=((SIN(AB))/(SIN(AC)))*SC:IF SB<0 THEN 100
90 PRINT" LADO A (SA) E'" SA: PRINT "LADO B(SB) E'" SB:
   GOTO 10
100 PRINT "DESCULPE; NAO E' UM TRIANGULO, TENTE
   NOVAMENTE"
110 GOTO 10
```

Quando o Computador pedir os ângulos AB e AC, você deverá entrar suas medidas em graus. Se tentar usar ângulos negativos ou ângulos maiores que 180 graus, o Computador vai para a linha 100, imprime a mensagem e retorna para a 10 requisitando novamente os ângulos AB e AC. Se você tentar fornecer um número negativo para o lado SC, o mesmo acontecerá! Como você não conhece o lado do ângulo AC, o Computador automaticamente o calcula na linha 40. Se a soma dos três ângulos não é igual a 180 graus, o Computador toma uma ação apropriada na linha 50. A linha 60 converte graus em radianos; assim o cálculo do seno pode ser feito.

FAZENDO O GRÁFICO DO SENO

Você provavelmente já viu uma onda senoidal. Elas são usadas em tensão CA e outras medidas elétricas. Rode o seguinte programa para ver uma onda senoidal:

```
 1 PMODE 1,1
 5 PCLS 3
10 SCREEN 1,1
15 COLOR 2,3
20 LINE (0,90)-(255,90),PSET
25 FOR G = 0 TO 255 STEP 5
30 X=G/57.29577951 * 2
35 LINE (G, - SIN(X)*60+90) - (G,90),PSET
40 NEXTG
45 GOTO 45
```

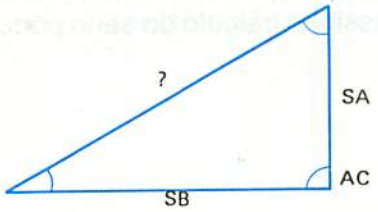
O CO-SENO

A função co-seno é relacionada com a função seno e tem a seguinte sintaxe:

COS(ângulo)

ângulo é uma expressão numérica para um ângulo em radianos.

Uma aplicação para o co-seno é determinar o comprimento de um 3.º lado de um triângulo se você já conhece dois lados e um ângulo. Para a identificação do lado e do ângulo, veja o triângulo do exemplo.



Note que o programa funciona do mesmo jeito do programa do seno, exceto na linha 50 onde aparece (↑) exponenciação e na linha 60, o SQR.

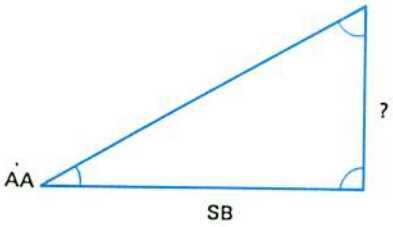
```
 5 CLS
10 INPUT "QUAL O ANGULO C (AC)" ; AC: IF AC<=0 OR
   AC>=180 THEN 100
20 AC=AC/57.29577951 'CONVERTE GRAUS EM RADIANOS
30 INPUT "QUAL O LADO A (SA)"; SA : IF SA<=0 THEN 100
40 INPUT "QUAL O LADO B(SB)" ; SB : IF SB<=0 THEN 100
50 SC=((SA↑2)+(SB↑2)) - (2*(SA*SB*COS(AC))):IF SC<0 THEN
   100
60 PRINT "LADO C (SC) E'" SQR(SC): GOTO 10
100 PRINT "DESCULPE, NAO E' UM TRIANGULO, TENTE
   NOVAMENTE"
110 GOTO 10
```

A TANGENTE

A 3ª função trigonométrica que você pode usar no seu Computador é TAN. Ela permite calcular a tangente de um ângulo.

TAN(ângulo)

ângulo é uma expressão numérica para um ângulo em radianos.



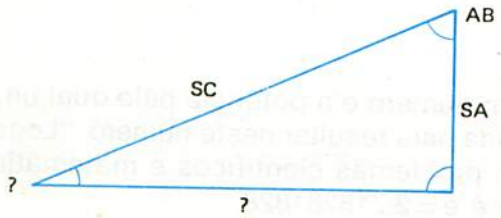
Entre outras coisas, a função tangente pode ser usada para determinar o comprimento desconhecido de um lado de um triângulo se você sabe a dimensão de outro lado e de um ângulo.

```
 5 CLS
10 INPUT "QUAL O LADO B (SB)"; SB : IF SB< =0 THEN 100
20 INPUT "QUAL O ANGULO A(AA)" AA: IF AA< =0 OR
   AA> =180 THEN 100
30 AA = AA/57.29577951 'CONVERTE GRAUS EM RADIANOS
40 SA = SB*(TAN(AA)): IF SA< =0 THEN 100
50 PRINT "LADO A (SA) E"' SA : GOTO 10
100 PRINT "DESCULPE, NAO E' UM TRIANGULO, TENTE
   NOVAMENTE"
110 GOTO 10
```

A chave para este programa, naturalmente, é a linha 40, onde a tangente do ângulo AA é multiplicada pelo comprimento do lado SB para determinar o comprimento do lado SA.

ARCO-TANGENTE (ATN)

O seguinte programa usa ATN e TAN para calcular 2 ângulos desconhecidos de um triângulo, quando 2 lados e 1 ângulo são conhecidos.



```
10 CLS
20 INPUT "QUAL O LADO A (SA)"; SA: IF SA< =0 THEN 150
30 INPUT "QUAL O LADO C (SC)"; SC: IF SC< = 0 THEN 150
40 INPUT "QUAL O ANGULO B (AB)"; AB:IF AB< =0 OR
   AB> =180 THEN 150
50 X=(180-AB) 'AA+AC=180-AB
60 X=X/57.29577951 'CONVERTE GRAUS EM RADIANOS
70 Y=((SA-SC)/(SA+SC))*TAN(X/2)
80 Z=ATN(Y)
90 AA=(X/2)+(Z)
100 AC=(X/2)-(Z)
110 AA=AA*57.29577951 'CONVERTE RADIANOS EM GRAUS
120 AC=AC*57.29577951 'CONVERTE RADIANOS EM GRAUS
130 PRINT "ANGULO A (AA) E'" AA "GRAUS"
140 PRINT "ANGULO C (AC) E'" AC "GRAUS"
150 PRINT "DESCULPE, NAO E' UM TRIANGULO, TENTE
   NOVAMENTE"
160 GOTO 20
```

Notar que as linhas 110 e 120 convertem radianos em graus.

TAN((AA – AC)/2 é igual a ((SA – SC)/(SA + SC))*TAN((AA + AC)/2).

LOGARITMOS

LOG fornece o logaritmo natural de um número. É o inverso do EXP, assim:

```
X=LOG(EXP(X))
```

LOG(número)

número é uma expressão numérica maior que 0.

O logaritmo de um número é a potência pelo qual uma dada "base" deve ser elevada para resultar neste número. "Logaritmos" são muito usados em problemas científicos e matemáticos. Na função LOG, a base é e = 2.718281828.

Para encontrar o logaritmo de um número em outra base B, use a seguinte fórmula:

$$\log_{\text{base B}}(x) = \frac{\log_e(x)}{\log_e(B)}$$

Por exemplo:

$$\log_2 32768 = \frac{\log 32768}{\log 2}$$

EXPONENCIAÇÃO

A função EXP é a exponencial natural de um número, isto é, $e^{\text{número}}$. Esta função é o inverso do LOG; por essa razão, X = EXP(LOG(X)).

EXP (número)

número é uma expressão numérica menor que 87.3365.

Rode este programa para ver como EXP funciona:

```
10 CLS
20 INPUT "ENTRE X";X
30 PRINT "EXP(X) = "; EXP(X)
40 GOTO 20
```

POR UM RESULTADO MELHOR

É impressionante como o Computador trabalha com um número com 9 dígitos significativos, especialmente quando 8 destes estão à direita do ponto decimal.

Entretanto, você pode não querer todos esses números; pode apenas querer a parte inteira do número (isto é, os números à esquerda do ponto decimal). FIX permite obter esse número, simplesmente cortando todos os que estão à direita do ponto decimal.

FIX (número)

número é uma expressão numérica.

Por exemplo:

```
PRINT FIX (2.34690)
2
OK
```

Digite este programa:

```
10 CLS
20 INPUT "UM NUMERO DO TIPO X.YZ";X
30 W = FIX(X)
40 F = ABS(X) - ABS(W)
50 PRINT "PARTE INTEIRA = "; W
60 PRINT "PARTE FRACIONARIA = "; F
70 GOTO 20
```

Agora rode-o. Ele separa a parte inteira da decimal de um número dado.

UMA FUNÇÃO ESPECIAL

O COLOR BASIC tem uma função numérica, DEF FN, diferente de todas as outras que já mencionamos. DEF FN permite que você crie sua própria função matemática e que a use da mesma maneira que as outras funções (tais como SIN, COS etc.). Uma vez tendo usado DEF FN para definir uma função, você pode colocá-la para trabalhar em seu programa, usando o prefixo FN junto ao seu nome que você designou para sua função.

Quando você usar esta característica, não se esqueça de usar a instrução DEF FN antes de executar a função que ela define.
De outra maneira, um ?UF ERRO (função indefinida) ocorrerá.

DEF FN nome (lista de variáveis) = fórmula

nome é o nome designado para a função que você criou.
lista de variáveis é a variável que representa cada variável a ser usada pela função.
fórmula define a operação relacionada com as variáveis dadas na **lista de variáveis**.
Notas: • *Os nomes das variáveis que aparecem na fórmula servem apenas para definir a fórmula; elas não afetam as variáveis de mesmo nome. Apenas um argumento é permitido na fórmula, entretanto DEF FN deve conter apenas uma variável.*
• *DEF FN pode apenas ser usado em um programa.*

Por exemplo, uma operação matemática que você usou várias vezes neste capítulo foi a conversão de graus para radianos. Não seria mais fácil se o Computador tivesse uma função "construída" para fazer isso por você?
Se você alterar o programa exemplo onde foi usado SIN, você verá como fica mais simples se criarmos DEF FN para converter graus em radianos.

```
7 DEF FNR(X) = X/57.29577951
60 AA = FNR(AA): AB = FNR(AB): AC = FNR(AC)
```

Você pode ver quantas digitações foram economizadas, introduzindo o 57.29577951 apenas uma vez! Quando FNR é chamado para uso, o Computador imediatamente insere os valores necessários e desenvolve a operação por ele definida.

EXERCÍCIO DO CAPÍTULO

Escrever um programa em que você pode escolher a função que deseja usar. Cada opção deve ter uma sub-rotina que pede os valores e exibe a resposta desejada. O nosso programa é dado no Apêndice A.
Nota: *Se você implementá-lo com as principais operações matemáticas ele se transforma em uma ótima calculadora.*

CAPÍTULO

16

a união faz a string

Nós não queremos ser repetitivos, mas voltamos a falar sobre strings. Se você quiser recordar dê uma olhada na Seção II.

STRINGS

A função STRING$ é usada para criar gráficos, tabelas, ou qualquer similar, pois cria uma seqüência de caracteres iguais.

STRING$ (comprimento, caractere)

comprimento é uma expressão numérica entre 0 e 255.
caractere é uma expressão string ou numérica que fornece um caractere para o código ASCII correspondente. Se uma string constante é usada ela deve estar entre aspas.

O número de caracteres exibidos depende do número que você especificou no comprimento. Os caracteres usados dependem do caractere ou código ASCII especificado. Veja na Seção IV uma lista completa dos códigos dos caracteres ASCII.
Você pode querer fazer o cabeçalho para uma tabela como, por exemplo, uma agenda de endereços.

```
 5 CLS
10 X$ = STRING$ (13,"*")
20 PRINT@0, X$; "AGENDA"; X$
30 Y$ = STRING$(4,"*")
40 Z$ = STRING$(9,42)
50 PRINT@32, "NOME";Y$;"ENDERECO";Z$;
   "TEL";Y$
60 GOTO 60
```

Na linha 10 nós designamos a X$ o valor STRING$(13,"*"), isto é, uma string de 13 asteriscos.
A linha 20 diz ao Computador para imprimir (iniciando a impressão na localização 0 da tela) X$, depois a palavra AGENDA seguida por X$ novamente. (Veja o quadro com as posições do PRINT@ na tela, na Seção IV).
Como X$ é igual a 13 asteriscos (*), esses caracteres são impressos antes e depois da palavra AGENDA.
Na linha 30 foi designado a Y$ a string de 4 asteriscos. Já na linha 40, nós não especificamos um asterisco, ao invés disso, foi indicado o código ASCII 42. Se você recorrer ao Apêndice F, códigos ASCII dos caracteres, verá que ele representa justamente o asterisco.

A linha 50 imprime o título da sua tabela. A partir daí você pode criar qualquer gráfico ou tabela.

POR PARTES

Se você quiser procurar uma string dentro de uma outra, use INSTR.

INSTR (posição,Sbusca,alvo)

posição especifica a posição na *Sbusca* (string de busca), onde a busca vai começar, e é uma expressão numérica entre 0 e 255. É opcional; se omitido, a busca começa automaticamente pelo primeiro caractere *Sbusca*.
Sbusca é a string onde será procurada a string alvo.
alvo é a string que você está procurando.

Nota: *Sbusca e alvo são expressões string.*

INSTR retorna um 0 se:
- *posição* é maior que o comprimento de *Sbusca*;
- *Sbusca* é nula;
- *alvo* não pode ser encontrado.

Vamos entender a função de INSTR:

```
 5 CLEAR 500
10 CLS
15 INPUT "TEXTO DE BUSCA";S$
20 INPUT "TEXTO DO ALVO";T$
25 C=0: P=1 'P=POSICAO
30 F=INSTR(P,S$,T$)
32 PRINT F
35 IF F=0 THEN 60
40 C=C+1
45 PRINT LEFT$ (S$,F-1)+STRING$(LEN(T$) ,CHR$(159))+
   RIGHT$(S$,LEN(S$)-F-LEN(T$)+1)
50 P=F+LEN(T$)
55 IF P<=LEN(S$)-LEN(T$)+1 THEN 30
60 PRINT "ENCONTROU"; C ; "OCORRENCIA(S)"
```

Vamos rodar um exemplo:

```
TEXTO DE BUSCA? O SEU COMPUTADOR E' MESMO ESPERTO
TEXTO DO ALVO? ES
O SEU COMPUTADOR E' M□□MO ESPERTO
O SEU COMPUTADOR E' MESMO □□PERTO
ENCONTROU 2 OCORRENCIA(S)
OK
```

O que acontece é que:

- Você atribuiu a S$(busca) o valor: O SEU COMPUTADOR E' MESMO ESPERTO (linha 15).
- Você atribuiu a T$(alvo) o valor: ES (linha 20).
- Você disse ao Computador para iniciar na 1.ª posição (P) de S$ e procurar por T$ (linha 30).
- Cada vez que o alvo é encontrado, a linha 35 imprime sua posição.
- Quando INSTR localiza T$, ele imprime e coloca um bloco amarelo (CHR$(159)) em T$. Depois ele sai em busca da próxima ocorrência de T$ e procede do mesmo modo para encontrá-la (linha 45-55).
- Finalmente, o Computador exibe quantas vezes encontrou T$ em S$ (linha 60).

O programa a seguir contém uma lista com nomes e endereços. Este é um modo fácil de armazenar informação. Notar como economizamos espaços não colocando espaços entre as palavras. Isto torna mais difícil para você ler, mas não para o Computador. Notar também que colocamos um asterisco antes do código postal para que o Computador não o confunda com o número da rua. Neste caso, estaremos procurando pelo nome e endereço de todos os indivíduos que moram na área de código postal 042 — conseqüentemente, *042 será o alvo da string (A$).

```
 5 CLS
10 A$ = "*042"
20 X$ = "JOAOSILVA,JUCUMA702,SANTANASP*02019:
   ELISANG,LOBO101,IPIRANSP*04212:PEDROSA,
   MACUCO20,INDIANSP*04523"
30 Y$ = "SANDRASANTOS,XINGU10,CARIOCASP*04222:
   MARCOSREIS,TANGARA130,CLEMENTSP*04019:
   PAULOSAT,CORREDEIRA433,SAUDESP*04127"
40 Z$ = "MARIABRAGA,PIUNA350,PLANALTOSP*03983:
   ROBERTODIAS,PACONE89,SUMARE*01254:RAULBISPO,
   IQUITOS111,PINHEIROSSP*05444"
```

O seu Computador pode buscar X$, acrescentando:

```
50 PRINT INSTR(X$,A$)
```

Agora rode o programa.

Sua tela exibirá:

```
61
OK
```

Isto significa que o código postal procurado se encontra na posição 61 de X$.
E Y$? Edite a linha 50 para o Computador buscar através destes endereços. Ele indicou a posição do dado que você precisa? Agora tente com Z$. Com certeza o Computador vai exibir um zero pois este é o melhor modo de dizer que não existe o que você procura nesta lista.

NOVA FRASE

Nós já falamos sobre MID$ na seção anterior. Ele retira uma parte de uma string. Agora vamos mostrar o que mais ele pode fazer. Se modificarmos sua sintaxe, além de retirar, ele pode substituir parte da string.

MID$ (string antiga, posição, comprimento) = substituto

string antiga é o nome da string que você quer alterar.
posição é uma expressão numérica especificando a posição do primeiro caractere a ser trocado.
comprimento é uma expressão numérica especificando o número de caracteres que serão substituídos. É opcional.
substituto é uma expressão de string que substitui uma parte específica de uma *string antiga*.
Nota: *Se* substituto *é menor que o comprimento, a string substituta inteira será usada. A string resultante é sempre do mesmo tamanho que a original.*

Rode o seguinte programa:

```
 5 CLS
10 A$ = "BARRA MANSA, SP"
20 MID$(A$,14) = "RJ"
30 PRINT A$
```

Na linha 10 foi atribuído a A$ o valor BARRA MANSA, SP. Depois, na linha 20, você pediu ao Computador para substituir a string antiga (A$) por RJ, iniciando na posição 14.
MID$ é duas vezes mais eficaz quando usada com INSTR. Usando as duas é possível "buscar e destruir" o texto; INSTR busca, MID$ troca ou "destrói". O programa a seguir ilustra o que dissemos:

```
 5 CLS
10 INPUT "DIA E MES (DD/MM). ";X$
20 P=INSTR(X$,"/")
30 IF P=0 THEN 10
40 MID$(X$,P,1)="-"
50 PRINT X$ "FICA MAIS FACIL LER, NAO FICA?"
```

Neste programa, INSTR busca por uma barra "/". Quando a encontra, MID$ coloca um hífen "-" no lugar.

EXERCÍCIO DO CAPÍTULO

Escreva um programa que permite saber a posição de uma string dentro de outra string e também substituir trecho dela.

CAPÍTULO

17

portas de acesso

Instruções de entrada/saída permitem enviar dados do teclado ao Computador, do Computador à TV, do Computador à impressora. Essas funções são usadas principalmente em programas para introduzir dados, sair resultados e mensagens.

AS LINHAS

LINE INPUT lembrete; variável string

lembrete é a mensagem lembrete. É opcional; se usada deve vir entre aspas.
variável string é o nome atribuído à linha que será introduzida pelo teclado.

LINE INPUT é similar a INPUT exceto:

- Quando a instrução é executada, e o Computador está esperando por uma entrada pelo teclado; não é mostrado o sinal de interrogação na tela.
- Cada instrução LINE INPUT pode atribuir um valor a uma única variável.
- Vírgulas e aspas serão aceitos como parte da string de entrada.
- Espaços em branco que vêm na frente da string não são ignorados — eles formam parte da variável string.

LINE INPUT é um modo conveniente de introduzir strings de dados sem ter que se preocupar com entradas acidentais de delimitadores (vírgulas, aspas, dois pontos etc.). Tudo é aceito. Em algumas situações você precisa introduzir vírgulas, aspas e espaços em branco antes do dado como parte dos dados e LINE INPUT atende bem tais casos. Por exemplo:

```
LINE INPUT X$
```

permite a entrada de X$ sem exibir qualquer lembrete.

```
LINE INPUT "ULTIMO NOME, PRIMEIRO NOME? ";N$
```

exibe uma mensagem lembrete e pede os dados. Os dados que vierem após a vírgula não serão ignorados como na instrução INPUT. Notar que fornecemos um sinal de interrogação e em seguida um espaço.

Rode o seguinte programa para ter uma idéia do LINE INPUT.

```
10 CLEAR 300: CLS
20 PRINT TAB(8); "INSTRUCAO LINE INPUT": PRINT
30 PRINT: PRINT "***TEXTO DE ENTRADA***"
40 '***DADA A STRING, IMPRIMI-LA***
50 A$= "" 'CANCELA A$
60 LINE INPUT "= = >"; A$
70 IF A$ = ""THEN END 'SE A STRING FOR NULA, PARE!
80 PRINT A$
90 GOTO 50
```

FORMATOS

Agora você descobriu mais coisas que pode fazer com o seu Computador e mais coisas que ele pode fazer por você. Por exemplo, você quer que o seu Computador crie uma tabela que use números, mas você não quer digitar os sinais mais e menos, repetidamente.
PRINT USING faz trabalhos deste tipo, habilitando o Computador a imprimir strings e números em um formato "personalizado". Isto pode ser especialmente útil quando estiver trabalhando com relatórios de contas, cheques, tabelas, gráficos ou qualquer outra situação que requeira um formado de impressão específico.

PRINT USING formato, item-lista

formato é uma string que diz ao Computador qual formato usar na impressão de cada um dos itens do item-lista. Consiste em "especificadores de campo" e outros caracteres.
item-lista é o dado a ser formatado.

Nota: *PRINT USING não imprime espaços em branco antes e depois do número, exceto quando indicado no formato.*

Os seguintes especificadores de campo podem ser usados como parte do *formato*:

#

Este sinal especifica a posição de cada dígito do número a ser mostrado. O número de # estabelece o tamanho do campo numérico. Se o campo numérico é maior que o número de dígitos

do valor, então a parte não usada, à esquerda do número, será preenchida com espaços e aquela, à direita do ponto decimal, será preenchida com zeros (veja a seguir). Se um campo numérico é pequeno para manter o número, este será exibido com um sinal % na frente.

```
PRINT USING "#####"; 66.2
66
PRINT USING "#"; 66.2
%66
PRINT USING "#.#"; 66.25
%66.3
```

Você pode colocar o ponto decimal em qualquer lugar do campo numérico estabelecido pelo sinal #. O Computador automaticamente arredondará qualquer dígito à direita do ponto decimal que não couber no campo.

```
PRINT USING "##.#" 58.76
58.8
PRINT USING "##.## ";10.2,5.3,66.897,.234
10.20 5.30 66.90 0.23
```

Neste exemplo, dois espaços foram colocados na string de formato depois do sinal # para separar os números quando impressos.

,

A vírgula, quando colocada em qualquer lugar entre o primeiro dígito e o ponto decimal de seu número, exibirá uma vírgula à esquerda de cada três dígitos. A vírgula estabelece uma posição adicional em seu campo numérico. Para evitar um estouro (indicado pelo sinal de porcentagem na frente) é uma boa idéia colocar uma vírgula a cada três posições no campo numérico. O estouro ocorre porque o campo não tem tamanho suficiente.

```
PRINT USING "#########,"; 12345678
12,345,678
PRINT USING "#########,"; 123456789
%123,456,789
PRINT USING "###,###,###"; 123456789
123,456,789
```

**

Quando você coloca dois asteriscos no início de um campo numérico, todas as posições não usadas, à esquerda do decimal, serão preenchidas com asteriscos. Os dois asteriscos estabelecem mais duas posições no campo numérico.

```
PRINT USING "**####"; 44.0
****44
```

$

Se seu número representar dinheiro coloque um $ na frente do campo numérico. Um $ será colocado na frente do número na saída.

```
PRINT USING "$###.##";18.6735
$ 18.67
```

$$

O sinal $$ colocado no início do campo atuará como um sinal $ flutuante. O sinal ficará justamente na frente do primeiro dígito.

```
PRINT USING "$$##.##"; 18.6735
$18.67
```

**$

Se estes três sinais são usados no início do campo, as posições vagas, à esquerda do seu número, serão preenchidas pelo sinal * e o sinal $ novamente se posicionará na primeira posição, precedendo seu número.

```
PRINT USING "**$.##"; 8.333
*$8.33
```

+

Quando um sinal de mais (+) é colocado no início ou fim do campo numérico, ele será impresso como um (+) para números positivos ou um menos (–) para números negativos.

```
PRINT USING "+**#####"; 75200
**+75200
PRINT USING "+###"; -216
-216
```

–

Quando um sinal de menos (–) é colocado no fim do campo, faz com que um sinal de menos apareça depois de todos os números negativos. Um espaço aparecerá depois dos números positivos.

```
PRINT USING "####.#-"; -8124.430
8124.4-
```

↑↑↑↑

Quatro setas para cima significam que o número é impresso na forma exponencial.

```
PRINT USING "##.####↑↑↑↑";123456
1.2346E+05
```

!

Um sinal de exclamação (!) faz com que o Computador imprima apenas o primeiro caractere da string.

```
PRINT USING "!"; "ARITMETICA"
A
```

%espaço%

Para especificar um campo de string de mais de um caractere, %espaço% é usado. O comprimento do campo será 2 mais a quantidade de espaços entre os sinais de porcentagem.

```
PRINT USING "% %"; "ABCDEFG"
ABCD
PRINT USING "% %"; "NADA DISSO"
NADA D
```

Para verificar se ficou tudo bem entendido, rode esse programa:

```
  5 CLS
 10 A$ = "**$##,###,###.##CRUZEIROS"
 20 INPUT "QUAL SEU PRENOME"; P$
 30 INPUT "QUAL SEU NOME"; N$
 40 INPUT "QUAL SEU SOBRENOME";S$
 50 INPUT "ENTRE A QUANTIDADE A PAGAR";P
 60 CLS
 70 PRINT "ORDEM DE PAGAMENTO PARA"
 80 PRINT USING "!";P$;"."N$;".";
 90 PRINT S$
100 PRINT: PRINT USING A$; P
110 GOTO 110
```

A linha 10 define um formato usando **$ para preencher os espaços da frente com asteriscos, e coloca um sinal de $ antes do primeiro número. (Este formato é as vezes usado para proteger cheques de serem alterados.) A linha 10 também ajusta o campo numérico usando o sinal #. Qualquer número que você entrar menor que o campo numérico será precedido com asterisco para preencher os espaços. Incluídos na linha 10 estão mais dois especificadores de campos, o ponto decimal e a vírgula.
O ponto decimal é impresso no campo exatamente onde você especificar. Como você disse ao Computador para ter duas casas à direita do ponto decimal (para centavos), qualquer número com mais de dois dígitos será arredondado para dois.
O sinal de exclamação (!) na linha 80 diz ao Computador para usar apenas o primeiro caractere de P$(prenome) e N$(nome). Assim as linhas 80 e 90 imprimem as iniciais de seu nome junto com o sobrenome.

CADÊ O CURSOR?

POS é uma função de entrada/saída que permite testar a posição do cursor na tela ou na impressora.

POS (número do dispositivo)

número do dispositivo é 0 (tela) ou -2 (impressora)

POS fornece um número indicando a posição do cursor na tela ou a posição do carro na impressora.

```
PRINT TAB (8) POS(0)
```

retorna o número 8 na coluna 8 da linha atual.
POS pode ser usado também para evitar a separação de sílabas no fim da linha, na tela ou na impressora. É uma aplicação importante, entretanto é necessário diminuir o comprimento da linha. Rode o seguinte programa para ver o POS funcionar.

O espaço na frente antes do "8" ocasiona que "8" apareça na coluna 9.

```
 5 CLS
10 A$ = INKEY$
20 IF A$ = ""THEN 10
30 IF POS (0)>22 THEN IF A$ = CHR$(32) THEN
   A$=CHR$(13)
40 PRINT A$;
50 GOTO 10
```

Este programa permite que você use o teclado como uma máquina de escrever (exceto que os erros não podem ser corrigidos a menos que a impressora seja primeiro desabilitada). POS estará atento ao final da linha, assim nenhuma palavra será dividida. Na linha 30, o Computador verifica se a posição "atual" do cursor é maior que a posição de coluna 22 (a tela tem 32 colunas). Se o cursor passou da coluna 22, o Computador inicia uma linha na próxima vez que você pressionar espaço (CHR$(32)). Quando o Computador decide iniciar uma nova linha, ele faz isso imprimindo um retorno de carro (CHR$(13)) — na verdade, o Computador imprime ENTER.

Nós escolhemos para teste a posição 22 do cursor, já que ela tem 10 espaços a menos que o comprimento máximo da tela, 32, dando uma boa margem para completar uma palavra muito longa.

ENTRADAS E SAÍDAS

Você já pensou no seu vídeo com um dispositivo de "saída" e no teclado como um dispositivo de "entrada"?

Com PRINT, PRINT USING, LINE INPUT e POS, você pode usar o número do dispositivo para direcionar a entrada ou a saída. Por exemplo, se você digitar:

```
PRINT#-2, USING "# # #.# # #";123.45678 ENTER
```

a tela permanecerá "silenciosa" enquanto a impressora trabalha, imprimindo:

```
123.457
```

Você pode usar qualquer um dos especificadores de campos normais com

```
PRINT#-2, USING.
```

POS(-2) fornece a posição atual da impressora (ou seja, a posição atual do carro). Rode o seguinte programa:

```
 5 CLS
10 FOR I=1 TO 10
20 PRINT#-2, "*";
30 PRINT "IMPRESSORA POS=";POS(-2)
40 NEXT I
50 PRINT#-2,""
```

A tela mostrará a posição do carro de impressão. Note que a "posição" é calculada internamente, não mecanicamente. A maioria das impressoras não imprime até que a linha 50 seja executada.
LINE INPUT# pode ser usado quase que do mesmo modo, exceto que ele permite que você leia uma "linha de dados" de um arquivo em fita cassete.
LINE INPUT# lê tudo do primeiro caractere até:

- um caractere de retorno de carro que não foi precedido por um caractere de avanço de linha;
- o fim de arquivo;
- o 249º caractere de dado.

Outros caracteres encontrados — aspas, vírgulas, espaços em branco antes dos dados, seqüência avanço de linha/retorno de carro — estão incluídos na string. Por exemplo:

```
LINE INPUT#-1,A$
```

carrega uma linha de dado do arquivo cassete em A$.
O programa seguinte usa LINE INPUT# para contar o número de linhas em qualquer programa armazenado em fita. Entretanto, o programa deve ter sido gravado em fita no formato ASCII (opção A).

```
 10 CLEAR 500
 20 LINE INPUT "NOME DO ARQUIVO DE DADOS? ";F$
 30 K=0 'K E' O CONTADOR
 40 OPEN "I",-1,F$
 50 IF EOF (-1) THEN 100
 60 LINE INPUT#-1, A$
 70 K=K+1
 80 PRINT A$
 90 GOTO 50
100 CLOSE#-1
110 PRINT "ARQUIVO CONTEM";K"LINHAS"
```

EXERCÍCIO DO CAPÍTULO

Escreva um programa para criar uma tabela, mostrando sua renda e gastos em um mês. Você deve relacionar o dinheiro que entra e sai, para o Computador calcular a renda líquida (mais ou menos).
Usar STRING$ para organizar a tabela, dando uma flexibilidade suficiente para poder usá-la mês após mês sem ter que mudar o programa inteiro.

CAPÍTULO

18

o toque que faltava

Há ainda umas poucas características que você não teve a chance de conhecer. Neste capítulo, trataremos de cada uma delas.

NÃO PRECISA

Em algum dialeto do BASIC, LET **tem de** ser usado quando você atribui um valor a uma variável (por exemplo LET X = 5). Como você sabe, o COLOR BASIC do seu CP 400 não requer LET. Mesmo assim, você pode usá-lo sem confundir o seu Computador. Uma razão para isso é assegurar a compatibilidade com outras versões do BASIC que usem LET. Dessa forma, tanto faz digitar:

```
10 LET A$ ="A#"
```

como usar:

```
10 A$ ="A#"
```

ATRÁS DELE!

TRON e TROFF são ferramentas de depuração que auxiliam você a "rastrear" a execução das instruções do programa.
TRON ativa um "rastreador" que imprime os números das linhas do programa à medida que são executadas. Os números aparecem entre colchetes. TROFF desativa este "rastreador".
TRON e TROFF são usados deste modo: TRON ENTER e TROFF ENTER. Digite este programa:

```
10 PCLS
20 PMODE 3,1
30 SCREEN 1,1
40 FOR X= 90 TO 120 STEP 10
50 LINE (0,0)-(X,155),PSET
60 NEXT X
70 GOTO 10
```

Digite TRON ENTER e rode o programa. O Computador exibirá:

```
[10][20][30][40][50][60][50][60][50][60][50][60][70]...
OK
```

Tudo isto significa que o Computador executa primeiro a linha 10, depois a 20, a 30, a 40, a 50, a 60, repete a 50 e a 60 mais 2 vezes e finalmente a 70. Não se esqueça de digitar TROFF ENTER para desativar o "rastreador".

HORA CERTA

Seu CP 400 também tem um "temporizador" embutido que você certamente usará em algum programa que requer tempo. Nós chamamos esta função de TIMER porque ela "marca o tempo" em 1/60 de segundo (aproximadamente).
No instante em que você liga o Computador o TIMER começa a contar do zero e vai até 65535. Esta operação demora quase 18 minutos. Assim que atinge 65535, ele volta para o zero e recomeça. (O contador pára durante a operação do gravador ou da impressora. O TIMER dá uma pausa durante estas interrupções.) Para saber o conteúdo do contador num certo momento, digite:

```
PRINT TIMER ENTER
```

e a tela exibirá um número de 0 a 65535.
Você também pode ajustar o TIMER com qualquer tempo digitando:

TIMER = número ENTER

onde número é uma expressão numérica entre 0 e 65535.

Para ver o TIMER (e PRINT@USING, uma nova função), rode o seguinte programa chamado "Teste de Matemática". Ele testará você com um problema de matemática. Quando você pressionar A, B, C ou D, o Computador lhe dirá se sua resposta está certa e quanto tempo você levou para responder (usando TIMER).

```
10 DIM CH(3),L$(3) 'CH(#) = ESCOLHA, L$ = FORMATO DA
   RESPOSTA
20 LI = 10:LS = 20 'LIMITE INFERIOR E LIMITE SUPERIOR
   PARA X E Y
30 NV = LS-LI + 1
40 P$ = "QUANTO E' # # # + # # #? 'FORMATO DA
   PERGUNTA
50 FOR I = 0 TO 3 'INICIALIZE CH( )
60 L$(I) = CHR$(I + 65) + ")# # #"
70 NEXT I
80 CLS
90 X = INT(RND(NV) + LI-.5) 'OBTER ENTRE LI E LS
100 Y = INT(RND(NV) + LI-.5) 'OBTER ENTRE LI E LS
110 R = INT(X + Y + .5) 'RESPOSTA CORRETA
130 FOR I = 0 TO 3 'OBTER MULT.ESCOLHAS
140 CH(I) = INT(RND(NV) + LI-.5)
150 NEXT I
160 RC = RND(4)-1 'FAZ UMA ESCOLHA CERTA
```

```
170 CH(RC)=R
180 PRINT@32, USING P$;X,Y 'EXIBE PROBLEMA
190 FOR LN=3 TO 6
200 PRINT@LN * 32+10, USING L$(LN-3);CH(LN-3)
210 NEXT LN
220 TIMER =0
230 A$=""  'LIMPAR TECLADO
240 A$=INKEY$: IF A$="" THEN 240
250 SV=TIMER 'SE TECLA E' PRESSIONADA, SALVE
    CONTEUDO DE TIMER
260 IF A$<"A" OR A$ "D" THEN 240 'TECLA INVALIDA —
    VOLTAR
265 PRINT@8 * 32+10,A$
270 K=ASC(A$)-65
280 IF CH(K)=R THEN PRINT "CERTO!": GOTO 300
290 PRINT "ERRADO! A RESPOSTA E'"; R
300 PRINT "VOCE LEVOU";SV/60; "SEGUNDOS"
310 INPUT "PRESSIONE <ENTER>PARA O PROXIMO
    PROBLEMA";EN
320 GOTO 80
```

Por tentativa, mudar os limites superior e inferior (linha 20) para *X* e *Y*. Você pode fazer com que ele desenvolva outra operação matemática, diferente da adição. Ou então pode querer que o Computador registre os tempos das respostas. Aquele que fizer o menor tempo é o vencedor. (Adicione 5 segundos para cada resposta incorreta.)

AS BASES DOS NÚMEROS

O COLOR BASIC permite o uso de constantes hexadecimais e octais.

Números hexadecimais são números representados na base 16, e são compostos por numerais de 0 a 9, e "numerais" de A a F. A constante hexadecimal deve estar na faixa 0-FFFF, correspondendo à faixa decimal 0-65535. Qualquer número precedido pelo símbolo &H é interpretado como um número hexadecimal. Por exemplo:

```
&HA010 &HFE &HD1 &HC &H4000
```

Números em octal são valores representados na base 8 e são compostos por numerais de 0 a 7. A constante octal deve estar na faixa 0-177777. É armazenada como inteiro de dois bytes, correspondendo ao decimal na faixa 0-65535. Qualquer número precedido

pelo símbolo &0 ou & é interpretado como uma constante decimal. Por exemplo:

```
&070 &054 &065 &717 &01234
```

Constantes "hex" (hexadecimal) e octal são convenientes em programas que fazem referência a posições de memória e seus conteúdos. Para maiores informações, consulte o Capítulo 19 e o Apêndice B.

CONVERSÃO

Quando trabalhar com programas em linguagem de máquina, você vai ver que é conveniente usar números em hexadecimal. Seu Computador converte facilmente um número decimal para hexadecimal com HEX$. Ele fornece uma string que representa um valor hexadecimal.

HEX$ (número)

número é uma variável ou número decimal de 0 a 65535.

EXERCÍCIO DO CAPÍTULO

Escreva um programa que permita converter valores livremente entre as 3 bases (decimal, hexa e octal).

Nota: *Os números sempre dentro da faixa decimal de 0 a 65535.*

CAPÍTULO

19

linguagem de máquina

Este capítulo descreve como chamar sub-rotinas em linguagem de máquina com um programa em BASIC, e lista certas sub-rotinas da ROM que você pode achar proveitosas.
"Linguagem de Máquina" (LM) é a linguagem usada internamente por seu Computador. Consiste em instruções diretas ao microprocessador. Sub-rotinas de linguagem de máquina são proveitosas para aplicações especiais, simplesmente porque elas podem fazer as coisas rapidamente. A escrita de tais rotinas requer familiaridade com programação de linguagem Assembly e com o conjunto de instruções do Microprocessador 6809E.
Nesta seção usaremos estes passos para usar a sub-rotina em LM, como segue:
1. Proteção de memória.
2. Armazenamento da sub-rotina LM na RAM.
3. Contar ao BASIC onde está a sub-rotina.
4. Chamar a sub-rotina.
5. Retornar ao BASIC.

Mais à frente, estaremos introduzindo um programa BASIC que desenvolve todas as 5 operações. Você pode digitar as linhas de programa em BASIC como elas são dadas, mas não tente executar o programa até ter lido toda esta seção.
Nossa sub-rotina LM será uma sub-rotina simples. Ela obtém um caractere do teclado. O caractere é fornecido como um código ASCII ao invés de uma string.
A sub-rotina tem umas poucas características não disponíveis com INKEY$ ou INPUT. Primeiro ela fornecerá qualquer código de tecla, incluindo o código para BREAK. Segundo, ela permite que você digite códigos de controle A-Z (CTRL-A até CTRL-Z). Para digitar um caractere de controle, pressione ↓, libere-o, então pressione qualquer tecla de A a Z. A faixa dos códigos de controle gerados varia de 1 a 26.
No retorno da sub-rotina, a referência à USR é "substituída" por um código de caractere.
Nós chamaremos a sub-rotina "PEGTEC". Para uma listagem desta sub-rotina, veja o fim desta seção.

Passo 1 — Proteger a memória

Com o comando CLEAR, você pode reservar uma parte de RAM para armazenar sua sub-rotina LM. O primeiro parâmetro CLEAR separa o espaço para strings e o segundo determina o endereço de proteção de memória. Por exemplo:

```
10 CLEAR 25, 16350
```

separa o espaço para strings com 25 bytes e reserva a memória acima de 4051 no final da RAM (ver mapeamento de memória). Seu programa LM pode então seguramente ser armazenado nesta área.

Passo 2 — Armazenar a sub-rotina de linguagem de máquina na RAM

Programas LM podem ser carregados da fita via CLOADM, ou colocados (POKE) na RAM. Em nosso exemplo, armazenaremos os códigos nas li-

nhas DATA; então lemos e colocamos cada código na localização correta da RAM. Os números nas linhas DATA são os códigos da sub-rotina LM listada mais adiante nesta seção.

```
20 FOR I=1 TO 28
30 READ B : POKE 16350+I, B
40 NEXT I
50 DATA 173, 159, 160, 0
60 DATA 39, 250, 129, 10, 38, 12
70 DATA 173, 159, 160, 0, 32, 250
75 DATA 129, 65, 45, 2
80 DATA 128, 64, 31, 137, 79
90 DATA 126, 180, 244
```

Passo 3 — Contar ao BASIC onde está a sub-rotina

Antes que você possa usar a sub-rotina, você tem de contar ao Computador onde ela começa. Você faz isto através da instrução DEFUSRn. Essa instrução define o endereço de entrada para uma chamada posterior. A chamada será realizada pela instrução USRn.
Aqui está a linha de programa para fazer isto:

```
100 DEF USR1=16351
```

Passo 4 — Chamar a sub-rotina

No ponto correto de seu programa, inserir uma referência à função USR:

```
110 A=USR1(0)
```

Em nosso exemplo, 0 é um "argumento auxiliar". Ele não será usado pela sub-rotina LM.
Quando este comando é encontrado, o BASIC chamará a sub-rotina LM.

Nota: *Na entrada para a sub-rotina, você pode obter o argumento USR (0 neste caso), chamando uma sub-rotina na EPROM, INTCNV, que retorna com o valor inteiro no registro D. O endereço do INTCNV é em hexadecimal.*

Passo 5 — Retornar ao BASIC

Se você não quer retornar qualquer valor ao programa em BASIC, termine a sub-rotina com uma instrução RTS. Se você quer retornar o valor de 2 bytes inteiros, carregue o inteiro no registro D na seqüência MSB-LSB; aí, então, termine a sub-rotina chamando a sub-rotina especial na ROM, GIVABF. O endereço de GIVABF é hexadecimal B4F4.
Depois de um RST, a referência USR, em seu programa BASIC, retornará ao argumento auxiliar original. Depois de uma chamada para GIVABF, a referência USR em seu programa BASIC fornecerá o valor que você carregou no registro D.

O PROGRAMA EM BASIC...

O seguinte programa coloca o código objeto na RAM e então usa a sub-rotina para obter entrada do teclado. Digite com cuidado e então rode. Cada vez que você pressiona uma tecla, o controle retorna ao BASIC com o código ASCII para aquela tecla. Tente pressionar BREAK. Você obterá o código 3 para BREAK. O programa termina quando você pressiona ENTER ou ↓ M
Para obter qualquer dos códigos de 1 a 26, pressione ↓, libere-o então pressione uma tecla de A a Z.

```
 10 CLEAR 25, 16350 ' MEMORIA DE RESERVA
 15 CLS
 20 FOR I=1 TO 28 ' ARMAZENA CADA BYTE DO CODIGO
    OBJETO
 30 READ B : POKE 16350+I, B
 40 NEXT I
 45 'AQUI ESTA' O CODIGO OBJETO
 50 DATA 173, 159, 160, 0
 60 DATA 39, 250, 129, 10, 38, 12
 70 DATA 173, 159, 160, 0, 39, 250
 75 DATA 129, 65, 45, 2
 80 DATA 128, 64, 31, 137, 79
 90 DATA 126, 180, 244
 99 'CONTAR AO BASIC ONDE ESTA' A SUB-ROTINA
100 DEF USR1 = 16351
110 A=USR1(0) 'CHAMA A SUB-ROTINA E POE O RESULTADO
    EM A
115 IF A=13 THEN END
120 PRINT "CODIGO="; A
130 GOTO 110
```

Para uma variação no programa, mude a linha 120 para:

```
120 PRINT CHR$ (A); ' EXIBE O CARACTERE
```

E O CÓDIGO FONTE

Nota: *Isto não é para ser digitado. Está aqui para aqueles que querem entender como a sub-rotina LM funciona.*

CÓDIGO OBJETO

HEXADECIMAL	Fonte	Código	Comentário
AD 9F AO 00	LOOP1	JSR[POLCAT]	; Procura uma tecla
27 FA		BEQ LOOP1	; Se nada, busca outra
81 OA		CMPA#10	; CTRL?
26 OC		BNE OUT	; Não, saída
AD 9F AO 00	LOOP2	JR POLCAT	; Sim, obter próxima tecla
27 FA		BEQ LOOP2	; Se nada, tente de novo
81 20		CMPA#65	; E A-Z?
20 02		BLT OUT	; Se <A, sai
80 40		SUBA#64	; Converte para CTRL A/Z
IF 89	OUT	TFR A,B	; Obter retorno Byte pronto
4F	OUT	CLRA	; Zera MSB
7E B4 F4		JMP GIVABF	; Retorna valor ao BASIC
	POLCAT	EQU 40960	
	GIVABF	EQU 46324	

Notas: *"Código fonte" não funciona para o Computador. O código fonte deve ser traduzido para código objeto, o qual o Computador entende. Na lista acima, o código objeto é dado na forma hexadecimal. Nós o convertemos em números decimais para nosso programa em BASIC.*

ATÉ DEZ ROTINAS POR VEZ

USRn é uma das funções de linguagem de máquina que você usará muito. É exatamente como a USR encontrada no programa exemplo deste capítulo, exceto que com USRn, você pode chamar até 10 sub-rotinas em linguagem de máquina, que foram anteriormente armazenadas e definidas por DEF USR. Você pode então continuar com a execução do seu programa BASIC.

Quando uma chamada USR é encontrada em uma instrução, o controle vai ao endereço definido na instrução DEF USRn. Este endereço especifica o ponto de entrada para sua rotina em linguagem de máquina. Terminada a sub-rotina em linguagem de máquina, o controle retorna à instrução seguinte à USRn.

USRn (argumento)

n especifica uma das chamadas USR disponíveis e é um número de 0 a 9. É opcional; se omitido, zero é assumido.
argumento é uma expressão numérica ou string.

LOCALIZAÇÃO DE MEMÓRIA PARA FUNÇÕES USR

A instrução CLEAR deve ser usada no início de um programa para reservar memória para as funções USR. Por exemplo:

```
CLEAR 50, 12000
```

Reserva a RAM iniciando no 12001.

CARREGANDO AS FUNÇÕES USR

As funções USR podem ser colocadas na memória ou carregadas da fita cassete usando CLOADM. A instrução DLOAD pode ser usada para retirar (transferir) as funções USR de outro Computador.
Uso da pilha — Uma função USR, que precisa mais de 30 bytes de pilha armazenada, deve prover sua própria área de pilha. Isto é completado salvando o indicador de pilha do BASIC na entrada para a função USR, colocando um novo indicador de pilha e rearmazenando o indicador de pilha do BASIC antes de retornar ao BASIC.

DEFININDO FUNÇÕES USR

DEF USR é usado para definir o endereço de entrada de uma função USRn.

DEF USRn = endereço

n é um dígito de 0 a 9; se omitido, 0 é usado.
endereço especifica o endereço de entrada para uma rotina de linguagem de máquina e deve estar na faixa entre 0 e 65535.

O valor desta expressão é passado para a função USR como seu argumento. (Ver "Argumentos da função USR" para uma informação mais detalhada sobre a passagem do argumento para a função USR.)

RETORNANDO AO BASIC DE UMA FUNÇÃO USR

Para retornar ao BASIC de uma função USR, um RTS ou uma seqüência de instrução equivalente deve ser executada. O indicador de pilha deve recuperar seu valor de entrada anterior ao retorno ao BASIC. Os valores dos registradores A, B, X e CC não precisam ser preservados pela função USR.
Funções USR sempre retornam um valor ao BASIC. A menos que a função USR explicitamente atribua um valor de retorno, o valor retornado é aquele do argumento passado à função USR. (Ver "Retornando valores ao BASIC" para maiores informações.)

ARGUMENTO DA FUNÇÃO USR

O argumento passado a uma função USR é o valor da expressão do argumento especificado na chamada USR. Na entrada para a função USR, o conteúdo do registrador A indica o tipo de argumento, como a seguir:

A = zero (argumento numérico)
A ≠ zero (argumento string)

Se o argumento é numérico, o registrador X contém um indicador para o Acumulador de Ponto Flutuante (FAC), que contém o argumento. É possível impor um inteiro como argumento, chamando a rotina INTCNV de BASIC da função USR (INTCNV = X'B3ED').
Se o argumento é uma string, INTCNV ocasiona um erro e o controle retorna ao BASIC. Se o argumento é um número em ponto flutuante fora da faixa −32768 a +32767, INTCNV provoca um erro de estouro, e o controle é retomado pelo BASIC. A rotina INTCNV fornece um inteiro em complemento de 2 de 16 bits em D.
Para argumentos numéricos ((A) = 0), FAC contém o expoente, FAC + 1 contém o MSB... FAC + 4 contém o LSB, e FAC + 5 contém o sinal da mantissa. O expoente é um inteiro de oito bits sinalizado com o 128 decimal adicionado a ele. Um expoente zero significa que o número é zero; neste caso, a mantissa é insignificante. A mantissa é armazenada na forma normalizada com o bit mais significativo do byte mais significativo igual a 1. Este bit pode ser então usado para indicar o sinal: 0 para a mantissa positiva, 1 para a mantissa negativa.

Para um argumento string, o registrador X aponta para uma expressão de cinco bytes. O primeiro byte da expressão é o comprimento (em caracteres) da string. O 3.º e 4.º bytes da expressão contêm o endereço do primeiro byte da string. O 2.º e 5.º bytes são reservados para o Computador e não estão disponíveis para uso.
Um indicador para uma variável BASIC pode ser passado para uma função USR usando-se a função VARPTR na expressão do argumento da função USR.
Sua sintaxe é:

VARPTR (*nome da variável*)

Por exemplo:

```
X = USR0(VARPTR(A))
```

passa um indicador para a variável A. INTCNV pode ser chamado para obter o indicador. É responsabilidade da função USR determinar o tipo de variável. Esta informação não é passada para a função USR pelo BASIC.
Para variáveis numéricas, o indicador é para um valor de ponto flutuante de cinco bytes; sendo o primeiro byte, o expoente, e os quatro bytes restantes, a mantissa. Valores de pontos flutuantes são armazenados em uma tabela de variáveis em um formato ligeiramente diferente daqueles que são armazenados no FAC. O bit mais significativo do byte mais significativo da mantissa é 1 (já que mantissas de ponto flutuante são normalizadas), e esta posição de bit é usada para armazenar o sinal da mantissa. O número é positivo se o bit de sinal for um zero; e negativo, se for um 1.
Para várias strings, o apontador é para uma expressão de cinco bytes. O primeiro byte é o comprimento da string, o 3.º e 4.º bytes são o endereço do 1.º caractere na string. O 2.º e 5.º bytes são reservados para o Computador e não estão disponíveis para uso.
É possível passar um indicador para uma matriz variável como o argumento para uma função USR. Neste modo, uma função USR pode acessar qualquer um dos elementos da matriz. Os valores dos elementos são armazenados na memória da seguinte maneira (listado de baixo para o alto da memória):

- valor do primeiro elemento da última dimensão
- valor do último elemento da última dimensão
- valor do primeiro elemento da primeira dimensão
- valor do último elemento da primeira dimensão

O comprimento de cada elemento é de 5 bytes. *Valor* é o valor que você atribuiu às variáveis dentro da string.
Outro método de passar valores para uma função USR é dar "POKE" aos valores na localização de memória reservados para uso da função USR.

RETORNANDO VALORES AO BASIC

A função USR sempre retorna no mínimo um valor ao BASIC, sendo este o valor da função. Se a rotina USR não retorna explicitamente este valor, o valor retornado é aquele do argumento passado à função USR.

Geralmente, o tipo do valor retornado é o mesmo do argumento da função USR. Neste caso, a função USR retorna seu valor no FAC como uma expressão string apontada por X, do mesmo formato que o argumento. Independente do tipo de argumento, um valor inteiro pode ser retornado carregando D com um inteiro de complemento de 2 de 16 bits, e chamando a rotina GIVABF do BASIC (GIVABF = X'B4F4') retorna ao BASIC. Valores adicionais podem ser retornados ao BASIC modificando os valores das variáveis BASIC. *Muito cuidado quando retornar valores string para o BASIC.* A seguinte advertência se aplica quando retornar uma string como o valor da função tão bem quanto quando modificar uma variável string do BASIC. O comprimento de uma string pode ser modificado trocando-se o byte de comprimento na expressão da string. Strings podem ser diminuídas desta maneira, mas uma função USR nunca deve tentar aumentar uma string. Se o comprimento da string a ser retornada ao BASIC não é conhecido quando da chamada, a string deve ser forçada ao comprimento máximo de 255 caracteres. Por exemplo:

```
A$ = USR0(STRING$(255,""))
```

passa uma string de 255 caracteres de espaços em branco para a função USR. A função USR pode então colocar uma string de até 255 caracteres na memória apontada pela expressão e diminuir a string se necessário.

O endereço inicial de uma string pode ser modificado trocando o apontador de dois bytes na expressão de string. Entretanto, o novo endereço de início geralmente deve ser aquele da localização de memória incluído na string original. Isto é, o endereço de início pode ser trocado por qualquer dos endereços OSA até OSA + OSL-1, onde OSA é o endereço de início original e OSL é o comprimento da string original. É também aceitável trocar os endereços de início entre duas strings. Isto deve ser proveitoso para uma rotina de ordenação de strings. Deve-se ter cuidado para assegurar que nenhuma das duas strings se misturem. É possível para uma expressão string apontar para um dado string que está atualmente dentro do programa BASIC. Isto pode apenas ocorrer quando uma string é definida como uma seqüência literal. Por exemplo, ambos A$ = "ABC" USR0("DEF") são expressões que apontam para o texto do programa BASIC. Se uma função USR tem como função modificar estas strings, o programa BASIC seria realmente alterado. Este problema pode ser evitado adicionando uma string nula ("") a qualquer seqüência literal que deve ser modificada por uma função USR. Por exemplo:

```
A$ = "ABC" + ""
```

fará com que a string seja copiada no espaço string, onde ela pode escapar de ser modificada por uma função USR.

Finalmente é possível adicionar valores ao BASIC armazenando-os nas localizações de memória alocadas para uso pela função USR. O programa BASIC pode então acessar estes valores usando a função PEEK.

ROTINAS DA ROM

O BASIC do CP 400 contém algumas sub-rotinas que podem ser chamadas por um programa na EPROM; algumas destas podem ser chamadas por um programa do BASIC via função USR ou USRn. Cada sub-rotina será descrita no seguinte formato:

NOME — Endereço de entrada
Operação desenvolvida
Condição de entrada
Condição de saída

Nota: *NOME é apenas para referência. Ele não é reconhecido pelo Computador. O endereço de entrada é dado na forma hexadecimal; você deve usar um salto indireto para este endereço. Condições de Entrada e Saída são dadas para programas de linguagem de máquina.*

BLKIN = [A006]

Lê um bloco do gravador cassete.

CONDIÇÕES DE ENTRADA
O gravador cassete deve estar ligado, ter bit de sincronismo (veja CSRDON). CBMFAD contém o endereço do buffer.
CONDIÇÕES DE SAÍDA
BLRTYP, que está localizado em 7C, contém o tipo de bloco:
0 = início do arquivo
1 = dados
FF = fim de arquivo
BLRLEN, localizado em 7D, contém o número de bytes de dados no bloco (0-255).

Z* = 1, A = CSRERR = 0 (se não há erros)

*Z — *é um flag (indicador) no registro de Código de Condição (CC).*

Z = 0, A = CSRERR = 1 (se ocorre um erro de verificação)
Z = 0, A = CSRERR = 2 (se ocorre um erro de memória)
(nota: CSRERR = 81)
A menos que um erro de memória ocorra, X = CBUFAD + BLKLEN. Se houver um erro de memória, X aponta para além do endereço ruim. As interrupções são mascaradas. U e Y são guardados e todos os outros modificados.

BLKOUT = [A008]

Envia um bloco para a fita cassete.

CONDIÇÕES DE ENTRADA
A fita deverá estar em velocidade normal e uma seqüência de 55 em hexadecimal deverá ter sido escrita se este 1.º bloco é escrito depois do motor ligado.
CBUFAD, localizado em 7E, contém o endereço do buffer.
BLKTYP, localizado em 7C, contém o tipo de blocos.
BLRLEN, localizado em 7D, contém o número de bytes de dados.

CONDIÇÕES DE SAÍDA
Interrupções são mascaradas; X = CBUFAD + BLRLEN. Todos os registros são modificados.

WRTLDR = [A00C]

Liga o gravador e escreve uma seqüência inicial.

CONDIÇÕES DE ENTRADA
Não há.

CONDIÇÕES DE SAÍDA
Não há.

CHROUT = [A002]

Envia um caractere para o Dispositivo CHROUT. Envia um caractere para o dispositivo especificado pelo conteúdo do GF (DEVNUM).
DEVNUM = 2 (impressora)
DEVNUM = 0 (tela)

CONDIÇÕES DE ENTRADA
Na entrada, o caractere que vai sair está em A.

CONDIÇÕES DE SAÍDA
Todos os registros exceto CC não se alteram.

CSRDON = [A004]

Inicia cassete (CSRDON); liga o gravador e obtém bit de sincronismo para leitura.

CONDIÇÕES DE ENTRADA
Não há.

CONDIÇÕES DE SAÍDA
FIRQ e IRQ são mascarados. U e Y, não se alteram. Os demais, modificados.

JOYIN = [A00A]

Amostra os potenciômetros dos joysticks. JDYIN mede todos os 4 potenciômetros de joystick e armazena seus valores no POTVAL até POTVAL + 3.

Joystick esquerdo
sobe/desce 15A
direita/esquerda 15B

Joystick direito
sobe/desce 15C
direita/esquerda 15D
Para sobe/desce, o valor mínimo = sobe
Para direita/esquerda, o valor mínimo = esquerda

CONDIÇÕES DE ENTRADA
Não há.

CONDIÇÕES DE SAÍDA
Y é preservado. Todos os outros são modificados.

POLCAT = [A000]

Ver no teclado se há um caractere pressionado.

CONDIÇÕES DE ENTRADA
Não há.

CONDIÇÕES DE SAÍDA
Z = 1, A = 0 (se nenhuma tecla está pressionada)
Z = 0, A = código da tecla (se a tecla está pressionada)
B e X são preservados. Todos os outros são modificados.

VARIÁVEIS DA IMPRESSORA SERIAL DO CP 400

VARIÁVEL	END.HEXA	END.DEC	INICIAL HEXA	VALOR DEC
LPTBTD Baud				
MSB	0095	149	00	0
LSB	0096	150	57	87
LPTLND Atraso de linha				
MSB	0097	151	00	0
LSB	0098	152	01	1
LPTCFW Comprimento do campo de vírgula	0099	153	10	16
LPTLCF Último campo de vírgula	009A	154	70	112
LPTWID Comprimento da impressora de linha	009B	155	84	132
LPTPOS	009C	156	00	00

O software do seu Computador usa as seguintes condições iniciais:
1. Baud é 600 baud.
2. Comprimento da impressora é 132 colunas.
3. Impressora gera uma saída ocupada quando não está pronta.
4. A impressora executará automaticamente um retorno de carro na coluna 132.

A interface RS 232 usa um conector DIN de 4 pinos. Um diagrama dos pinos é mostrado no Capítulo 1.
O pino 4 é a saída do Computador para a impressora. O pino 3 é o terra. O pino 1 não é usado para a impressora. O pino 2 deve ser conectado à saída ocupada (ou linha de estado) da impressora. Se sua impressora não fornece uma indicação de estado, então esta linha deve estar conectada a uma tensão positiva maior que 3 volts. Isto diz ao Computador que a impressora está pronta o tempo todo. Em adição, a variável de atraso de linha deve ser ajustada ao valor apropriado. A seguinte lis-

ta de valores alternados para variáveis de impressora de linha é fornecida para ajudar na interface de impressoras não-padrão.

Faixa de velocidade	Valor Dec. (msb,lsb)	Valor hexadecimal
120 baud	458(1 e 202)	01CA
300 baud	180	00BE
600 baud	87	0057
1200 baud	41	0029
2400 baud	18	0012
Atraso de linha	**Valor Dec. (msb,lsb)**	**Valor hexadecimal**
.288 seg	64 e 0	4000
.576 seg	128 e 0	8000
1.15 seg	255 e 255	FFFF
Comprimento da linha	**Valor Dec.**	**Valor hexadecimal**
16 carac/linha	16	10
32 carac/linha	32	20
64 carac/linha	64	40
255 carac/linha	255	FF

A última variável do campo de vírgula deve ser ajustada ao valor do comprimento — o comprimento do campo de vírgula. (O comprimento do campo de vírgula normalmente permanecerá em 16.)
Na versão 1.0 do COLOR BASIC o formato de saída para a impressora é um bit de partida, sete bits de dados (primeiro LSB) e dois bits de parada sem paridade.

MAPA DE MEMÓRIA DO CP 400

End. decimal	Conteúdo	End. hexadecimal
0-1023	Uso do sistema	0-3FF
255	RAM página direta	0FF
1023	RAM página ampliada	3FF
1025-1535	Memória da tela de texto Memória da tela de gráfico	400-5FF
1536-3071	Página 1	600-BFF
3072-4607	Página 2	C00-11FF
4608-6143	Página 3	1200-17FF
6144-7679	Página 4	1800-1DFF
7680-9215	Página 5	1E00-23FF
9216-2559	Página 6	2400-9FF
2560-12387	Página 7	2A00-2FFF
12288-13823	Página 8	3000-35FF
13824-16383	Programa e variável armazenada	3600-3FFF
32768-49151	COLOR BASIC	8000-BFFF
49152-65279	Memória de cartucho	C000-FEFF
65280-65535	Entrada/saída	FF00-FFFF

CAPÍTULO

20

exemplos de programas

BANDEIRA

REQUISITOS

16 k

PROGRAMA

```
 5 REM BANDEIRA
10 CLEAR 200
15 CLS
20 A$ = CHR$(159):A$ = A$ + A$
25 PRINT@160,STRING$(192,207);
30 FOR P = 225 TO 285 STEP 4
35 PRINT@P,A$;
40 NEXT P
45 PRINT@71,"RESERVA DE MERCADO";
50 PRINT@395,"DEFESA DOS";
55 PRINT@424, "VALORES NACIONAIS";
60 GOTO 60
```

INVENTÁRIO — LISTA DE COMPRAS

REQUISITOS

16 k

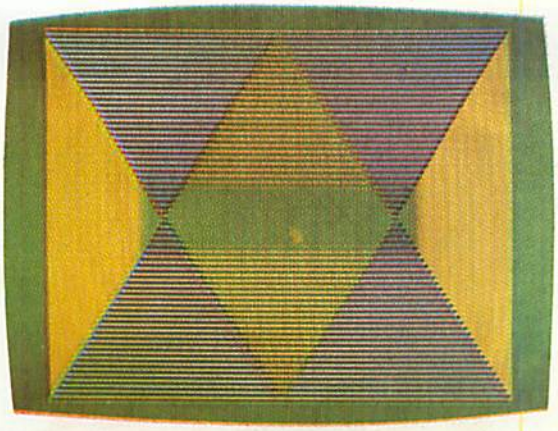
GRAVADOR

PROGRAMA

```
 5 CLEAR 2000: DIM S$(100)
10 REM INVENTARIO/LISTA DE COMPRAS
20 CLS
30 PRINT@71; "VOCE QUER--"
40 PRINT@134, "(1) ENTRAR ITENS"
50 PRINT@166, "(2) SUBSTITUIR ITENS"
60 PRINT@198, "(3) ADICIONAR A LISTA"
70 PRINT@230, "(4) ANULAR ITENS"
80 PRINT@262, "(5) IMPRIMIR TODOS ITENS"
90 PRINT@294, "(6) SALVAR ITENS NA FITA"
100 PRINT@326, "(7) CARREGAR ITENS DA FITA"
110 PRINT@395, "(1-7)";
120 INPUT M
130 IF M<0 OR M>7 THEN 10
140 ON M GOSUB 1000, 2000, 1020, 3000, 4000, 5000,
    6000
150 GOTO 10
900 REM
1000 REM ENTRAR/ADICIONAR ITENS
1010 Y=1
1020 CLS: PRINT@8, "ENTRAR/ADICIONAR ITENS"
1030 PRINT@34, "PRESSIONAR ENTER QUANDO
     TERMINAR"
1040 PRINT: PRINT "ITEM" Y;
1045 INPUT S$(Y)
1050 IF S$(Y) = " " THEN RETURN
1060 Y = Y+1
1070 GOTO 1040
1900 REM
2000 REM SUBSTITUIR ITENS
2005 N=0
2010 CLS: PRINT@9, "SUBSTITUIR ITENS"
2020 PRINT@34, "PRESSIONE ENTER QUANDO
     TERMINAR"
2030 PRINT: INPUT "N. DO ITEM PARA SUBSTITUIR"; N
2040 IF N=0 THEN RETURN
2050 INPUT "ITEM SUBSTITUTO"; S$(N)
2060 GOTO 2000
2900 REM
3000 REM ANULAR ITENS
3005 N=0
3010 CLS: PRINT@9, "ANULAR ITENS"
3020 PRINT@34, "PRESSIONE ENTER QUANDO
     TERMINAR"
3030 PRINT: INPUT "ITEM PARA ANULAR"; N
3035 IF N>Y-1 THEN 3030
3040 IF N=0 THEN RETURN
3050 FOR X = N TO Y-2
3060 S$(X) = S$(X+1)
3070 NEXT X
3080 S$(X) = " "
3090 Y = Y-1
3100 GOTO 3000
3900 REM
4000 REM IMPRIMIR ITENS
```

```
4010 FOR X = 1 TO Y-1 STEP 15
4020 FOR Z = X TO X+14
4030 PRINT Z; S$(Z)
4040 NEXT Z
4050 INPUT "PRESSIONE ENTER PARA CONTINUAR";
     C$
4060 NEXT X
4070 RETURN
4900 REM
5000 REM SALVAR ITENS NA FITA
5010 CLS: PRINT@135, "SALVAR ITENS NA FITA"
5020 PRINT@234, "POSICAO DA FITA"
5030 PRINT@294, "PRESSIONE PLAY E RECORD"
5040 PRINT@388, "PRESSIONE ENTER QUANDO
     PRONTO"
5050 INPUT R$
5060 OPEN "0",#-1, "LISTA"
5070 FOR X = 1 TO Y-1
5080 PRINT#-1, S$(X)
5090 NEXT X
5100 CLOSE#-1: RETURN
5900 REM
6000 REM CARREGAR ITENS NA FITA
6010 CLS: PRINT@136, "CARREGAR ITENS NA FITA"
6020 PRINT@235, "RETROCEDER FITA"
6030 PRINT@300, "PRESSIONE PLAY"
6040 PRINT@388, "PRESSIONE ENTER QUANDO
     PRONTO"
6050 INPUT R$
6060 OPEN "I",#-1, "LISTA"
6070 Y=1
6080 IF EOF(-1) THEN 6120
6090 INPUT#-1, S$(Y)
6095 PRINT S$(Y)
6100 Y = Y+1
6110 GOTO 6000
6120 CLOSE#-1: RETURN
```

EFEITOS ESPECIAIS



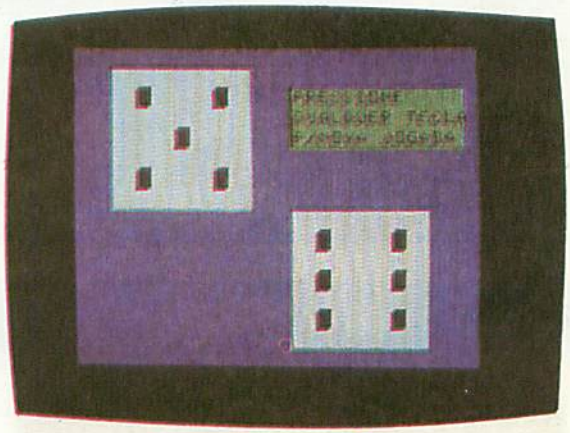
REQUISITOS

16 k

PROGRAMA

```
 2 'EFEITOS ESPECIAIS
 5 PMODE 3,1
10 PCLS3
15 SCREEN 1,0
20 FOR I = 3 TO 7
25 FOR J = 2 TO 6
30 FOR S = 0 TO 3
35 FOR R = 0 TO 3
40 COLOR R,S
45 A = 0:B=255:C=0:D=191
50 LINE (A,C)-(B,D),PSET,B
55 A=A+J:B=B-J:C=C+I:D=D-I
60 IF A<255 AND C<191 THEN 50
65 NEXT R
70 NEXT S
75 NEXT J,I
80 GOTO 30
```

DADOS ELETRÔNICOS



REQUISITOS

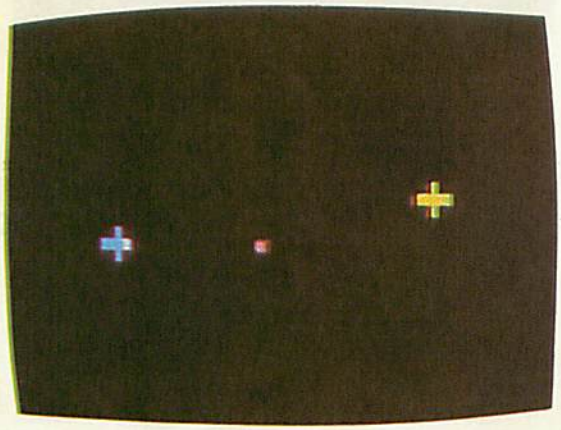
16 k

PROGRAMA

```
 4 CLEAR 2000
 5 CLS (3)
 6 DIM DB$(6)
 8 DIM DF(21), P(6), D$(6)
10 REM FACES DO DADO
20 FOR X=1 TO 21
30 READ DF(X)
40 NEXT X
50 DATA 39
60 DATA 14, 64
70 DATA 14, 39, 64
80 DATA 14, 20, 58, 64
90 DATA 14, 20, 39, 58, 64
100 DATA 14, 20, 36, 42, 58, 64
105 FOR X=1 TO 7
110 REM
120 REM LUGAR NA MATRIZ DF
130 FOR X=1 TO 6
```

```
140 READ P(X)
150 NEXT X
160 DATA 1, 2, 4, 7, 11, 16
165 REM
170 REM
175 FOR X=1 TO 6
180 M=P(X)
185 FOR Y=1 TO 7
190 FOR Z=1 TO 11
192 IF (Y-1)*11+Z<>DF(M) THEN 200
194 D$(X) = D$(X) + CHR$(128)
196 M = M+1
197 IF M=22 THEN M=0
198 IF M=X THEN M=0
199 GOTO 230
200 D$(X)=D$(X) + CHR$(143+64)
230 NEXT Z
240 FOR Z=0 TO 31-11
250 D$(X)=D$(X) + CHR$(143+32)
260 NEXT Z
270 NEXT Y, X
480 REM
490 REM JOGAR DADO
500 FOR T=1 TO 10
510 A=RND(6): B=RND (6)
520 PRINT 35, D$(A);
530 PRINT 273, D$(B);
540 NEXT T
545 PRINT 81, "PRESSIONE          ";
550 PRINT 113, "QUALQUER TECLA";
560 PRINT 145, "P/ NOVA JOGADA ";
570 K$=INKEY$: IF K$=" " THEN 570
580 GOTO 500
```

ARMAS ESPACIAIS



REQUISITOS

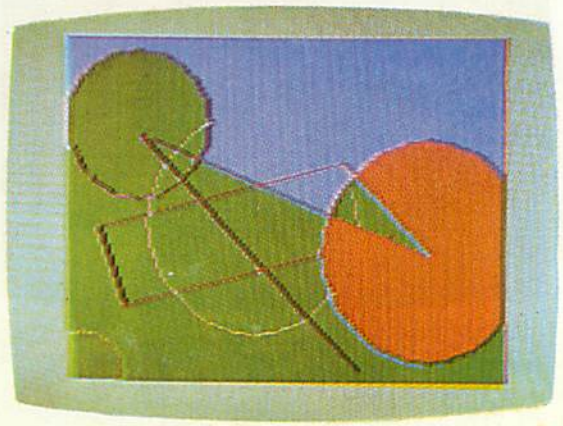
16 k

joysticks

PROGRAMA

```
10 CLEAR
20 FOR Y=0 TO 1
30 C=(Y+1)*16
40 S$(Y) = CHR$(131+C)+CHR$(139+C)+
   CHR$(130+C)
50 S2$(Y) = CHR$(128+C)+CHR$(136+C)
60 NEXT Y
100 FOR Y=0 TO 1
105 C = JOYSTK(0)
110 A(Y) = JOYSTK(0+Y*2)
120 B(Y) = JOYSTK(1+(Y*2))
130 IF A (Y)>59 THEN A(Y) = 59
140 B(Y) = INT (B(Y)/4)*4
150 L(Y) = B(Y) *8 + INT(A(Y)/2)
160 IF L(Y)> = 480 THEN L(Y) = L(Y)-32
170 NEXT Y
180 CLS(0)
190 FOR Y=0 TO 1
200 PRINT@L(Y), S$(Y)
210 PRINT@L(Y)+32, S2$(Y);
220 NEXT Y
500 P = PEEK(65280)
510 IF P = 125 OR P = 253 THEN GOSUB 1000
530 GOTO 100
800 REM
900 REM ROTINA ATIRA FOGO
1000 V1 = INT(B(1)/2)+1
1010 H1 = A(1) +2
1020 IF A(1)<>A(0) THEN 1100
1030 FOR H = H1 + 3 TO 63
1040 IF POINT (H,V1) = 2 THEN SOUND 100,2
1050 SET (H,V1,4)
1060 IF H<= H1 +4 THEN 1080
1070 RESET (H - 2, V1)
1080 NEXT H
1090 RETURN
1100 FOR H= H1 TO 4 STEP - 1
1110 IF H =H1 THEN 1160
1120 IF POINT (H-4,V1)=2 THEN SOUND 100,2
1130 SET (H-4, V1,4)
1140 IF H =H1 - 2 THEN 1160
1150 RESET (H-2,V1)
1160 NEXT H
1170 RETURN
```

GRÁFICOS ALEATÓRIOS



REQUISITOS

16 k

PROGRAMA

```
  1 '***GRAFICOS ALEATORIOS***
  2 '
 10 PMODE 3,1
 15 PCLS
 20 SCREEN 1,1
 25 F =RND(4):B=RND(8): IF B=F OR (B-4=F)
    THEN 25
 30 COLOR F, B:PCLS B: FOR L = 0 TO 5
 35 LINE -(RND(255),RND(191)),PSET
 40 CIRCLE (RND(255),RND(191)),RND(100),RND(4)/2
 50 NEXT: FOR P=0 TO 10
 55 PAINT (RND(255),RND(191)),RND (4),F
 60 NEXT
 70 I$=INKEY$:IF I$= " " THEN 70
 80 GOTO 10
```

VENTILADOR

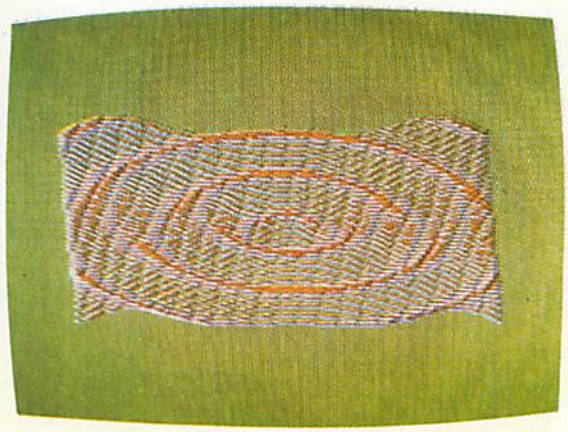
REQUISITOS

16 k

PROGRAMA

```
  1 '***VENTILADOR***
  2 '
  5 PCLEAR 8
 50 GOTO 600
 60 LINE ((255-X), (191-Y))-(X,Y),PSET
 70 J = J+1:IF J>A THEN J=0:A=RND(50)
 80 RETURN
600 REM VENTILADOR
601 FOR I = 1 TO 5 STEP 4
602 PMODE 3,I
603 PCLS
604 SCREEN 1,0
605 A=25:X=0: Y=0: J=0
610 FOR X=0 TO 254
612 COLOR X/32+1,5
615 GOSUB 60: NEXT X
620 FOR Y=0 TO 190
623 COLOR Y/24+1,5
625 GOSUB 60: NEXT Y
630 FOR X = 255 TO 1 STEP -1
633 COLOR X/32+1,5
635 GOSUB 60:NEXT X
640 FOR Y = 191 TO 1 STEP -1
643 COLOR Y/24+1,5
645 GOSUB 60: NEXT Y
650 NEXT I
660 FOR I = 1 TO 5 STEP 4
670 PMODE 3,I
680 SCREEN 1,0
690 FOR T = 1 TO 30: NEXT T
700 NEXT I
710 GOTO 660
```

RELEVO TRIDIMENSIONAL



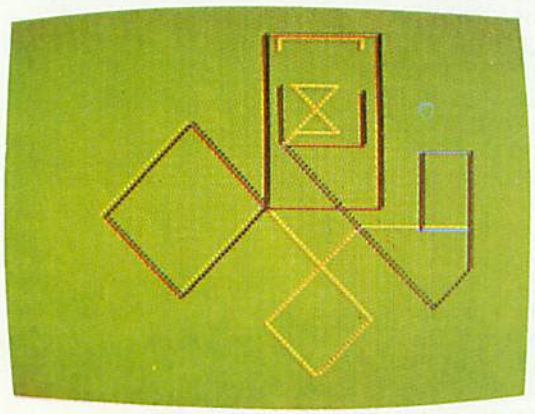
REQUISITOS

16 k

PROGRAMA

```
  5 REM RELEVO TRIDIMENSIONAL
 10 PMODE 3,1
 20 PCLS
 30 SCREEN 1,1
 40 FOR A = 0 TO 10 STEP .2
 50 FOR X = 0 TO 10 STEP .1
 60 Y = -10*COS(3*SQR((X-5)↑2+(A-5)↑2))/2+50
 70 PSET (X+24+A,Y+A*10,8)
 80 NEXT X
 90 NEXT A
 99 GOTO 99
```

DESENHANDO



REQUISITOS

16 k

PROGRAMA

```
1 '***DESENHANDO***
2 '
3 CLS
5 PRINT@128,STRING$(32,"*"):PRINT@288,
  STRING$(32, "*")
10 PRINT@200, "DESENHANDO"
15 FOR X=1 TO 600: NEXT X
20 CLS
25 PRINT@96, "PRESSIONE<↑>PARA SUBIR,<SETA
   PARA BAIXO>
   <RETROCESSO> PARA A ESQUERDA, <TAB>
   PARA A DIREITA,
   PARA DESCER, <A> SUDOESTE, <S> SUDESTE,
   <W> NOROESTE, <Q> NORDESTE"
30 PRINT@288,"PRESSIONE <1> PARA LINHA
   INVISIVEL, <2>, <3>, OU <4>, PARA LINHAS
   VISIVEIS DE CORES DIFERENTES, PRESSIONE
   </> PARA MUDAR O CONJUNTO DE CORES"
35 PRINT@448,"PRESSIONE <ESPAÇO> PARA
   PAUSAR
40 FOR X=1 TO 4800: NEXT X
45 CC=4: TG=0
50 PMODE 3,1
55 PCLS
60 SCREEN 1,TG
70 X=128:Y=96:X1=0:Y1=0
80 U$="↑":D$=CHR$(10):W$=CHR$(8):E$=CHR$(9)
90 NW$="Q": NE$="W": SW$="A": SE$="S"
100 C1$="1":C2$="2":C3$="3": C4$="4"
110 A$=INKEY$
120 IF A$=U$ THEN YI=-1:XI=0: GOTO 240
130 IF A$=D$ THEN YI=1:XI=0: GOTO 240
140 IF A$=W$ THEN XI=-1:YI=0: GOTO 240
150 IF A$=E$ THEN XI=1:YI=0: GOTO 240
160 IF A$=NE$ THEN XI=1:YI=-1: GOTO 240
170 IF A$=NW$ THEN XI=-1:YI=-1: GOTO 240
180 IF A$=SE$ THEN XI=1:YI=1: GOTO 240
190 IF A$=SW$ THEN XI=-1: YI=1: GOTO 240
200 IF C1$<=A$ AND A$<=C4$ THEN CC=
    ASC(A$)-48: GOTO 240
210 IF A$="/" THEN TG=(NOT TG AND 1) OR
    (TG AND NOT 1): GOTO 240
220 SCREEN 1, TG
230 IF A$=" " THEN XI=0: YI=0
240 X=X+XI:Y=Y+YI:IF X<0 THEN X=0
250 IF X>255 THEN X=255
260 IF Y<0 THEN Y=0
270 IF Y>191 THEN Y=191
275 IF CC=1 THEN PSET(X,Y,3)
280 PSET (X,Y,CC)
290 GOTO 110
```

SEÇÃO

IV

APÊNDICES

RESPOSTAS DOS EXERCÍCIOS A

CAPÍTULO 3

```
10 FOR X= 255 TO 1 STEP - 10
20 PRINT "TOM" X
30 SOUND X,1
40 NEXT X
50 FOR X=1 TO 255 STEP 5
60 PRINT "TOM" X
70 SOUND X,1
80 NEXT X
```

CAPÍTULO 4

```
 10 CLS
 20 T=0
 30 N=RND(255)
 40 D= RND(8)
 50 CLS(D)
 60 SOUND N,D
 70 T=T+1
 80 IF T= 15 THEN 100
 90 GOTO 30
100 CLS 0
110 FOR V= 0 TO 31
120 FOR H = 0 TO 63
130 IF V = 10 THEN SET(H,10,1)
140 IF V= 15 THEN SET(H,15,2)
150 IF V= 20 THEN SET(H,20,3)
160 NEXT H,V
170 END
```

CAPÍTULO 5

```
10 FOR C = 0 TO 8
20 CLS(C)
30 FOR S = 0 TO 59
40 CLS(C)
50 PRINT S
60 FOR T = 1 TO 445
70 NEXT T
80 NEXT S
90 NEXT C
```

CAPÍTULO 8

```
 10 CLS 0
 20 H=32
 30 V=14
 40 SET(H,V,8)
 50 A$= INKEY$
 60 IF A$ = CHR$(8) THEN 110
 70 IF A$ = CHR$(9) THEN 140
 80 IF A$ = CHR$(94) THEN 170
 90 IF A$ = CHR$(10) THEN 200
100 GOTO 50
110 H=H-1
120 IF H<0 THEN H=0
130 GOTO 40
140 H= H+1
150 IF H>63 THEN H=63
160 GOTO 40
170 V= V-1
180 IF V<0 THEN V=0
190 GOTO 40
200 V = V+1
210 IF V>31 THEN V = 31
220 GOTO 40
```

CAPÍTULO 11

```
  5 PMODE 1,1
 10 COLOR 5,8
 15 PCLS 8
 20 SCREEN 1,1
 25 LINE (4,188) - (252,4),PSET,B 'MOLDURA
 30 LINE (99,139) - (159,101),PSET,B 'PAREDE
 35 LINE - (143,79),PSET 'TELHADO
 40 LINE - (84,79),PSET 'TELHADO
 45 LINE - (68,101),PSET 'TELHADO
 50 LINE - (99,139),PSET,B 'PAREDE
 55 LINE (84,79) - (99,101),PSET 'TELHADO
 60 LINE (76,139) - (91,120),PSET,B 'PORTA
 65 LINE (116,123) - (139,112),PSET,B 'JANELA
 70 LINE (112,79) - (119,72),PSET,B 'CHAMINE
 73 LINE (4,50) - (36,72),PSET 'MONTANHA
 75 LINE - (76,46),PSET
 80 LINE - (108,57),PSET
 85 LINE - (152,39),PSET
 90 LINE - (200,72),PSET
```

CAPÍTULO 11 (cont.)

```
 95 LINE – (252,52),PSET
100 LINE (4,139) – (252,139),PSET 'CERCA
105 LINE (4,130) – (68,130),PSET
110 LINE (159,130) – (252,130),PSET
115 FOR X=20 TO 250 STEP 20
120 IF X>68 AND X<159 GOTO 130
125 LINE (X,125) – (X,139),PSET
130 NEXT X
135 GOTO 135
```

CAPÍTULO 14

Mude a linha 220:

```
220 PLAY "T+;XA$;XB$;XC$;XD$;XE$;XF$;XG$;
    XH$"
```

E acrescente as linhas:

```
215 PLAY "T1;"
230 GOTO 220
```

CAPÍTULO 15

```
 10 CLS
 20 PRINT@98, "QUAL FUNCAO VOCE QUER
    USAR?"
 30 PRINT@135, "<1> POTENCIACAO"
 40 PRINT@167, "<2> RAIZ QUADRADA"
 50 PRINT@199, "<3> SENO"
 60 PRINT@231, "<4> CO-SENO"
 70 PRINT@263, "<5> TANGENTE"
 80 PRINT@295, "<6> LOGARITMO"
 90 PRINT@327, "<7> EXPONENCIACAO"
100 PRINT@385, "DIGITE O NO. CORRESPON
    DENTE";
110 INPUT ""; N
120 ON N GOTO 200,300,400,500,600,700,800
130 GOTO 10
200 CLS
210 PRINT@9, "POTENCIACAO"
220 INPUT "QUAL O NUMERO"; X
230 INPUT "QUAL A POTENCIA"; Y
240 PRINT X "ELEVADO A "Y"="(X)↑Y
250 GOTO 900
300 CLS
```

CAPÍTULO 15 (cont.)

```
310 PRINT@9 "RAIZ QUADRADA"
320 INPUT "QUAL O NUMERO"; X
330 PRINT "A RAIZ QUADRADA DE
    "X"="SQR(X)
340 GOTO 900
400 CLS
410 PRINT@13, "SENO"
420 INPUT "QUAL O ANGULO"; X
430 PRINT "O SENO DE" X "="
    SIN(X/57.29577951)
440 GOTO 900
500 CLS
510 PRINT@11, "CO-SENO"
520 INPUT "QUAL O ANGULO"; X
530 PRINT "O CO-SENO DE" X "="
    COS(X/57.29577951)
540 GOTO 900
600 CLS
610 PRINT@11, "TANGENTE"
620 INPUT "QUAL O ANGULO"; X
630 PRINT "A TANGENTE DE
    "X"="TAN(X/57.29577951)
640 GOTO 900
700 CLS
710 PRINT@9, "LOGARITMO"
720 INPUT "QUAL O NUMERO"; X
730 PRINT "O LOGARITMO DE "X"="LOG(X)
740 GOTO 900
800 CLS
810 PRINT@9, "EXPONENCIACAO"
820 INPUT "QUAL O NUMERO"; X
830 PRINT "A EXPONENCIAL DE "X"="EXP(X)
840 GOTO 900
900 PRINT@388, "<1>MESMA FUNCAO"
910 PRINT@420, "<2>OUTRA FUNCAO"
920 PRINT@452, "<3>TERMINAR"
930 PRINT@352, "PRESSIONE:";
940 INPUT "";A$
950 IF A$= "1" THEN 120
960 IF A$= "2" THEN 10
970 END
```

CAPÍTULO 16

```
10 CLS
20 INPUT "UMA STRING";A$
30 PRINT "PRESSIONE <P> P/ POSICAO DA
   LETRA <T> P/ TROCA";
40 C$= INKEY$
50 INPUT C$
```

CAPÍTULO 16 (cont.)

```
 60 IF C$ = "P" THEN 90
 70 IF C$ = "T" THEN 230
 80 GOTO 30
 90 CLS
100 PRINT A$
110 INPUT "ALVO"; B$
120 P = 1
130 F = INSTR(P,A$,B$)
140 IF F<1 THEN PRINT F: GOTO 180
150 PRINT F
160 P = F + LEN(B$)
170 IF P< = LEN(A$) - LEN(B$) + 1 THEN 130
180 PRINT "PRESSIONE <1> PARA VOLTAR
    AO PROGRAMA <2> PARA TERMINAR";
190 D$ = INKEY$
200 INPUT D$
210 IF D$ = "1" THEN 10
220 END
230 CLS
240 PRINT A$
250 INPUT "SUBSTITUIR POR"; F$
260 INPUT "QUAL POSICAO"; P
265 IF P > LEN(A$) THEN 260
270 MID$(A$,P) = F$
280 PRINT A$
290 GOTO 180
```

CAPÍTULO 17

```
 5 CLS
10 INPUT "RENDA"; R
20 INPUT "DESPESAS"; D
30 N = R-D
40 A$ = "$$####.##"
50 B$ = "$$####.##"
60 C$ = "$$####.##"
70 CLS: PRINT@33, "MOVIMENTO MENSAL"
80 PRINT@96, STRING$(32,"-")
90 PRINT@160,"RENDA"
100 PRINT@256,"DESPESAS"
110 PRINT@352, "TOTAL (+) OU (-)"
120 PRINT@180, USING A$;R
130 PRINT@276, USING B$;D
140 PRINT@371, USING C$;N
150 GOTO 150
```

CAPÍTULO 18

```
10 CLS
20 INPUT "SE O VALOR DECIMAL DE UM
   NUMERO E'"; DEC
30 PRINT "O SEU VALOR HEXADECIMAL E'"
   HEX$(DEC)
```

B INFORMAÇÕES TÉCNICAS

Se você tiver algum problema na operação de seu Computador CP 400, verifique a tabela a seguir, que descreve alguns defeitos e sua respectiva correção. Se, entretanto, você não conseguir detectar o problema, entre em contato com a Prológica.

TABELA DE DEFEITOS E PROVIDÊNCIAS

DEFEITO	CORREÇÃO
O OK (ou mensagem equivalente se você estiver usando um cartucho) não aparece quando se liga o Computador.	1. Não há energia elétrica. Verifique a ligação do cabo de alimentação. 2. Seqüência incorreta para ligar. Todos os periféricos devem ser ligados antes do Computador. 3. Dispositivo periférico (por exemplo, a impressora) não está conectado corretamente. Verifique as conexões. 4. A tela de seu televisor precisa de ajuste. Verifique o contraste, brilho ou ajuste fino da sintonia. 5. Verifique se o cabo da antena do Computador está corretamente ligado no seu aparelho de TV.
Recepção pobre ou imagem imprecisa.	1. Verifique se a sua TV está ajustada no canal apropriado (3 ou 4 — aquele que tiver a melhor imagem). 2. Verifique as conexões da antena para certificar-se de que foram feitas com segurança e corretamente. 3. Seu aparelho de TV precisa de ajuste. Verifique os controles de contraste, brilho e ajuste fino.
Seu programa em fita não carrega.	1. Conexão imprópria do gravador. Verifique as instruções de conexão do gravador, no Capítulo 7. 2. O volume do gravador está muito alto ou baixo. Verifique o controle de volume do gravador. 3. Informação na fita pode ter sido alterada devido a descarga elétrica, campo magnético, ou fita deteriorada. Tente carregar a outra cópia.
O Computador "trava" durante a operação normal, requerendo RESET ou Liga/Desliga.	1. Flutuações na tensão CA. Veja "Energia elétrica", adiante. 2. Conector instalado incorretamente ou com defeito. Verifique todos os cabos de conexão para ver se estão seguramente ligados e se não estão gastos ou partidos. 3. Programação. Cheque o programa.
"Fantasmas" ou recepção de TV e Computador misturada.	Tente usar outro canal de TV (3 ou 4). Verifique se a antena interna está desligada.

ENERGIA ELÉTRICA

Computadores são sensíveis a flutuações no fornecimento de energia elétrica. Isto raramente é um problema, a menos que você esteja operando próximo a uma máquina elétrica. A potência fornecida pode também ser instável se algum eletrodoméstico ou equipamento de escritório nas vizinhanças tem um interruptor defeituoso que "faísque" quando ligado ou desligado.
Seu Computador está equipado com um filtro CA. Ele elimina os efeitos de flutuações da tensão e faiscamento. Entretanto, se as flutuações são severas, você precisa levar em consideração os seguintes detalhes:

- Instalar filtros especiais ou de isolação nos dispositivos que causam problemas.
- Consertar ou substituir qualquer interruptor defeituoso, mesmo os de lâmpadas ou eletrodomésticos.
- Instalar uma linha de alimentação separada para o Computador.
- Instalar um filtro de linha especial próprio para Computadores e outros equipamentos eletrônicos sensíveis.

Problemas na linha de alimentação são raros e algumas vezes podem ser evitados pela escolha certa do local da instalação. Quanto mais complexa e séria a aplicação, maior a importância que você deve dar para a escolha de fonte de alimentação ideal para seu Computador.

ESPECIFICAÇÕES TÉCNICAS

FORNECIMENTO DE TENSÃO CA

Tensão	105 – 130 V_{CA}, 60 Hz
Corrente	0.18 Amps RMS

MICROPROCESSADOR

Tipo	6809E
Clock	0.895 MHz

INTERFACE SERIAL

Sinal padrão RS-232C		Pino
CD	Detecção da portadora (linha de entrada de estado)	1
RD	Recepção dos dados	2
TERRA	Tensão de referência zero	3
TD	Transmissão de dados da saída	4

LOCALIZAÇÃO DOS PINOS DA RS-232C

SOFTWARE DA IMPRESSORA

600 baud
1 bit de partida (zero lógico)
7 bits de dados (menos significativo antes)
2 bits de parada (um lógico)
Sem paridade
Largura da impressora 132 colunas
Retorno de carro automático ao final da linha

INTERFACE PARA CASSETE

Nível de entrada para reprodução do gravador 1 a 5 volts pico a pico em uma impedância mínima de 220 ohms

INTERFACE PARA CASSETE

Nível de saída do Computador para o gravador 800 mV pico a pico em 1 Kohms
Capacidade de chaveamento Liga/Desliga Remoto 0.5 A máximo em 6 V_{CC}

LOCALIZAÇÃO DOS PINOS DA TOMADA DO GRAVADOR

1. Controle remoto
2. Entrada da tomada do fone de ouvido do gravador
3. Sinal terra
4. Saída para a tomada AUX ou MIC do gravador
5. Controle remoto

LOCALIZAÇÃO DOS PINOS DA TOMADA DO CONTROLADOR DE JOYSTICK

1. Entrada do comparador (direita-esquerda)
2. Entrada do comparador (p/cima-p/baixo)
3. Terra
4. Botão de "disparo", nível alto quando aberto, baixo quando fechado
5. V_{CC}, +5V limitado por corrente

C MENSAGENS DE ERRO

MENSAGENS DE ERRO

Abreviação	Explicação
/0	Divisão por zero. É impossível dividir por zero, também nos Computadores.
AO	Tentativa de abrir um arquivo que já foi aberto. Se você pressionar as teclas RESET durante a operação do gravador, você obterá esta mensagem. Desligue o Computador e ligue novamente.
BS	Índice válido. Os índices de uma matriz estão fora da faixa. Use DIM para dimensionar a matriz. Por exemplo, se você tiver A(12) em seu programa sem uma linha DIM precedente que dimensiona a matriz A para 12 ou mais elementos, você obterá este erro.
CN	Não pode continuar. Se você usar o comando CONT e o programa já tiver acabado ou se estiver em outra situação que não permite continuar, você obterá este erro.
DD	Tentativa de redimensionar uma matriz. Uma matriz pode ser dimensionada uma vez apenas. Por exemplo, você não pode ter DIM A(12) e DIM A(50) no mesmo programa.
DN	Erro de número de dispositivo. Apenas três dispositivos podem ser usados com OPEN, CLOSE, PRINT, ou INPUT; 0, −1, ou −2. Se você usar outro número, obterá este erro.
DS	Instrução direta. Há uma instrução direta no arquivo de dados, sem número de linha. Isto pode ser causado se você tentar carregar um arquivo de dados em fita.
FC	Chamada de função ilegal. Isto acontece quando você usa um parâmetro (número ou variável) inválido ou que está fora da faixa. Por exemplo, PLAY":" causará este erro.
FD	Dados de arquivo inválido. Este erro ocorre quando você imprime dados em um arquivo, ou introduz dados de um arquivo, usando o tipo errado de variável para o dado correspondente. Por exemplo, INPUT#-1,A, quando o dado no arquivo é uma string, ocasiona este erro.
FM	Modo de arquivo inválido. Este erro ocorre quando você tenta introduzir dados de um arquivo aberto apenas para saída (O), ou imprimir dados de um arquivo aberto para entrada (I).
ID	Instrução direta ilegal. Por exemplo, você pode usar INPUT apenas como uma linha de um programa, não como uma linha de comando.
IE	Entrada depois do final de arquivo. Usar EOF para verificar quando alcançou o final do arquivo. Quando alcançar, feche-o (CLOSE).
IO	Erro de entrada/saída. Freqüentemente causado por tentar entrar um programa ou dados de arquivo em uma fita ruim.

MENSAGENS DE ERRO

Abreviação	Explicação
LS	String muito longa. Uma string pode conter apenas 255 caracteres.
NF	NEXT sem FOR. NEXT foi usado sem a instrução FOR. Este erro também ocorre quando você tiver linhas NEXT invertidas em um ciclo alojado.
NO	Arquivo não aberto. Você não pode introduzir ou enviar dados de ou para um arquivo a menos que ele tenha sido aberto (OPEN).
OD	Faltam dados. Um READ foi executado com insuficiência de dados. Uma instrução DATA deve ter sido excluída do programa.
OM	Falta memória. Toda memória disponível foi usada ou reservada.
OS	Falta espaço para strings. Não há espaço suficiente na memória para sua operação com strings. Você deve limpar mais espaços (CLEAR).
OV	Estouro. O número é muito grande para o Computador manipular (ABS(X) > 1E38).
RG	RETURN sem GOSUB. Uma linha RETURN foi encontrada sem uma linha GOSUB correspondente,
SN	Erro de sintaxe. Este erro resulta de um comando mal digitado, com pontuação incorreta, parênteses incorretos, ou um caractere ilegal. Redigite a linha de programa ou o comando.
ST	Fórmula de string muito complexa. Foi usada uma operação com string muito complexa para se manipular. Divida-a em operações menores.
TM	Tipo errado. Isto ocorre quando você tenta atribuir dados numéricos a uma variável string (A$ = 3) ou dados string a uma variável numérica (A = "DADO").
UL	Linha indefinida. Você tem um GOTO, GOSUB, ou outro desvio no programa, pedindo para o Computador ir a um número de linha que não existe.

D CORES E SONS

Estes são os códigos para cada uma das nove cores disponíveis no seu CP 400.

CÓDIGO	COR
0	Preto
1	Verde
2	Amarelo
3	Azul
4	Vermelho
5	Cinza
6	Ciano
7	Roxo
8	Laranja

Nota: As cores podem variar suas tonalidades em função do seu aparelho de TV. O código zero (preto) corresponde, na verdade, à ausência de cor. Os códigos dos caracteres gráficos para o modo de texto estão representados na figura a seguir. Como se vê, existem 16 caracteres distintos, numerados de 0 a 15. A cor de fundo é sempre o preto, podendo-se variar a cor de primeiro plano, segundo a fórmula:

```
Código = 112 + 16 * cor + caractere
```

Por exemplo, se você quiser o caractere gráfico 7, na cor vermelha, faça:

```
CÓDIGO = 112 + 16 * 4 + 7
? CHR$(CÓDIGO);
```

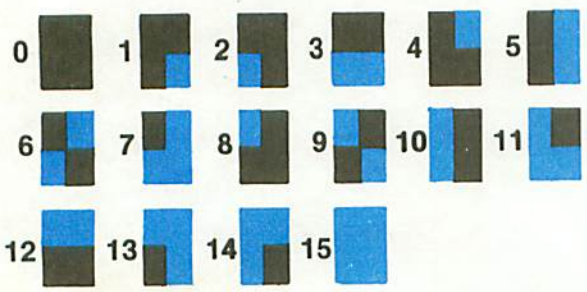


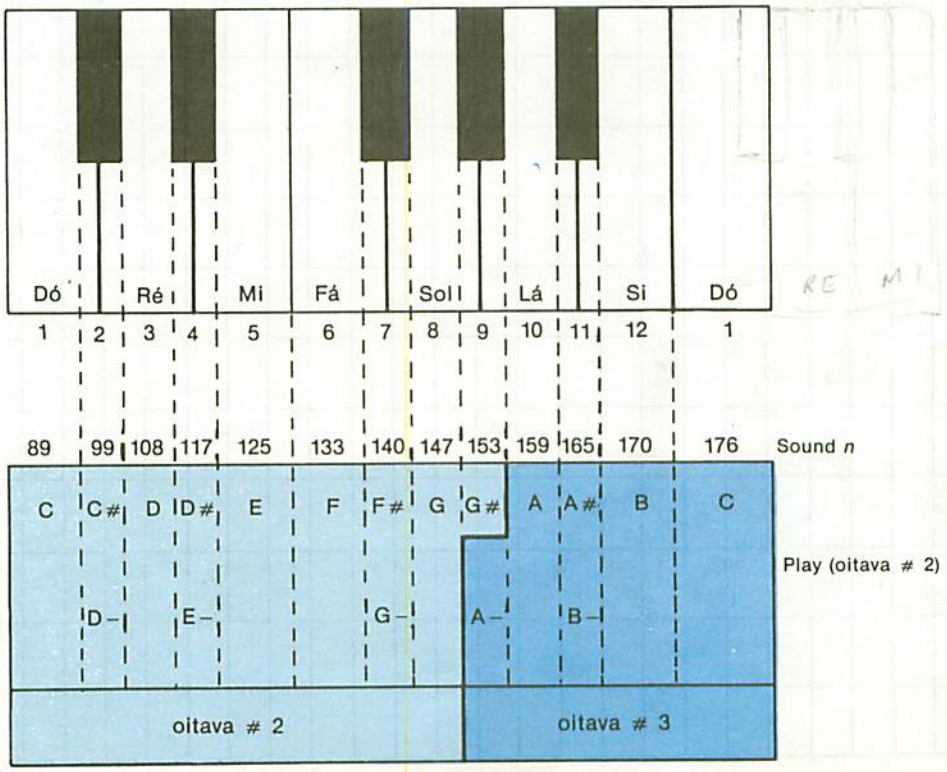
TABELA DOS CONJUNTOS DE CORES

PMODE n	SCREEN 1,c	DUAS CORES	QUATRO CORES
0	0	preto/verde	
	1	preto/cinza	
1	0		verde/amarelo/azul/vermelho
	1		cinza/ciano/roxo/laranja
2	0	preto/verde	
	1	preto/cinza	
3	0		verde/amarelo/azul/vermelho
	1		cinza/ciano/roxo/laranja
4	0	preto/verde	
	1	preto/cinza	

NOTAS MUSICAIS

A instrução PLAY não reconhece as notações B# e C-. Use os números 1 e 12, respectivamente, ou substitua B# por C e C- por B. Se você tentar usar essas notações ocorrerá um FC ERRO.

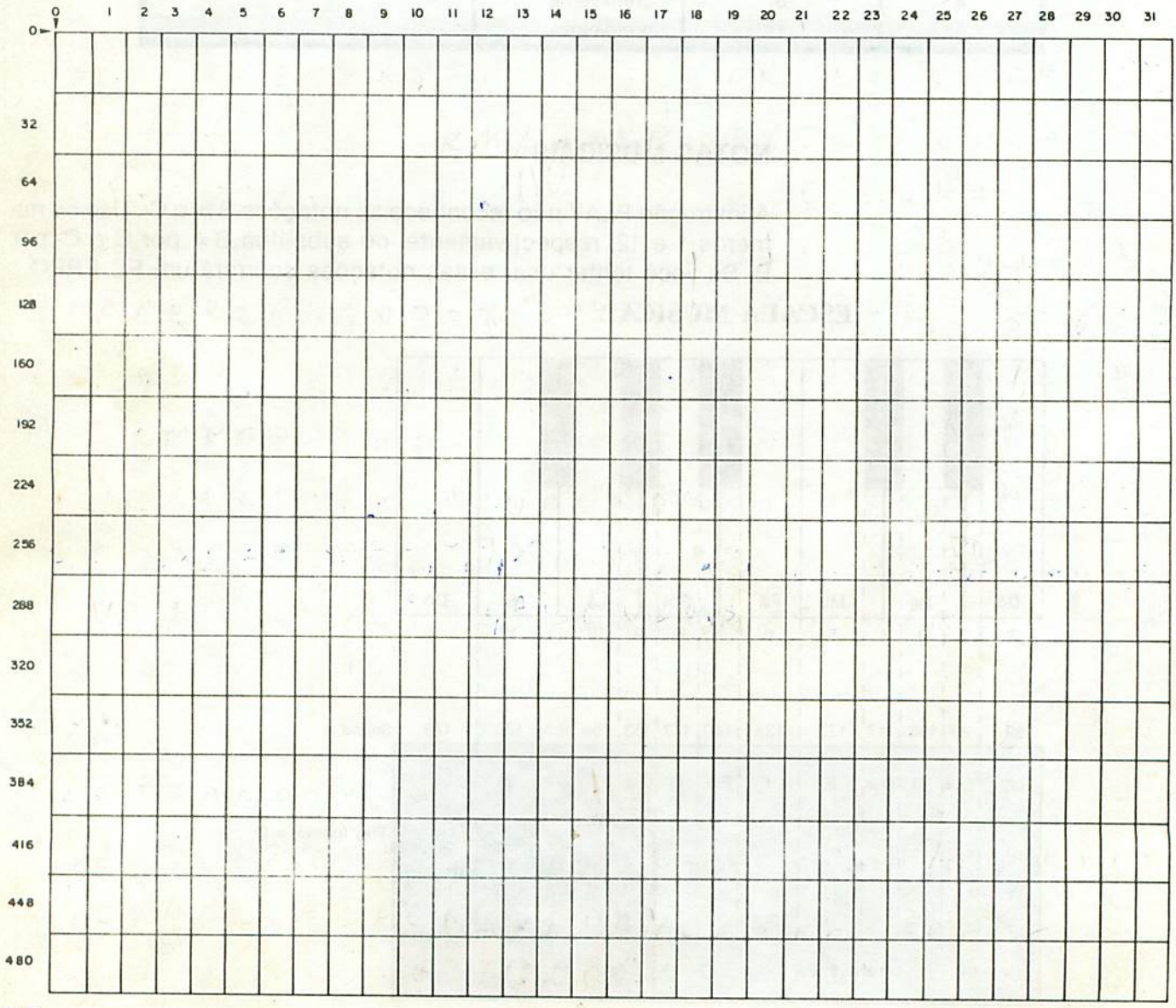
ESCALA MUSICAL



E POSIÇÕES DOS MODOS GRÁFICOS

A distribuição de mensagens na tela pode ser realizada com relativa simplicidade utilizando-se a instrução PRINT@. A figura a seguir representa a tela e numera as diferentes posições possíveis.

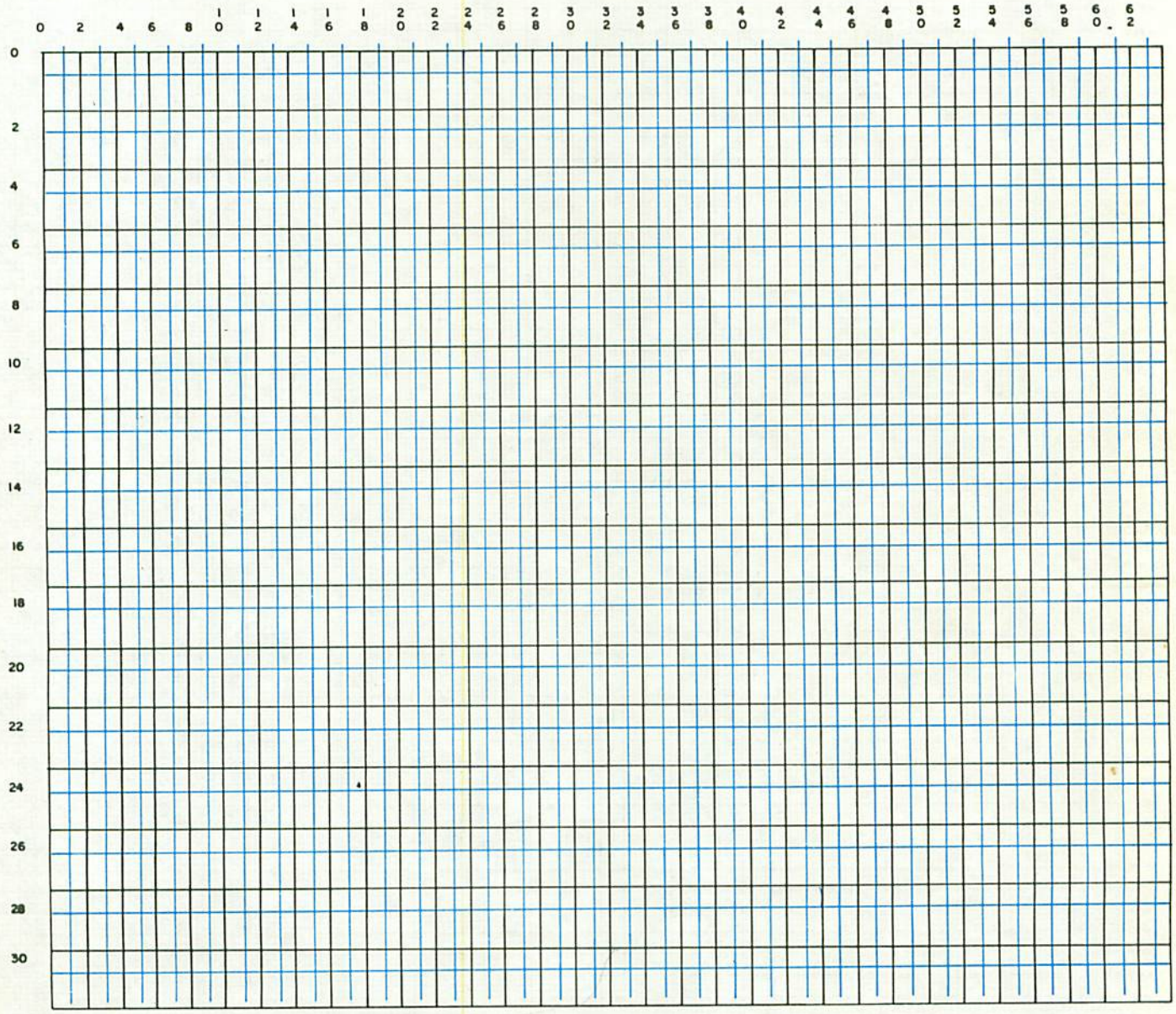
POSIÇÕES DO PRINT@ NA TELA



Ainda na tela de textos, utilizando-se a função CHR$ ou as instruções SET e RESET, pode-se misturar desenhos elementares com texto. Esse recurso permite uma resolução (detalhamento) baixa, mas que pode ser útil em alguns casos.

POSIÇÕES GRÁFICAS NA TELA

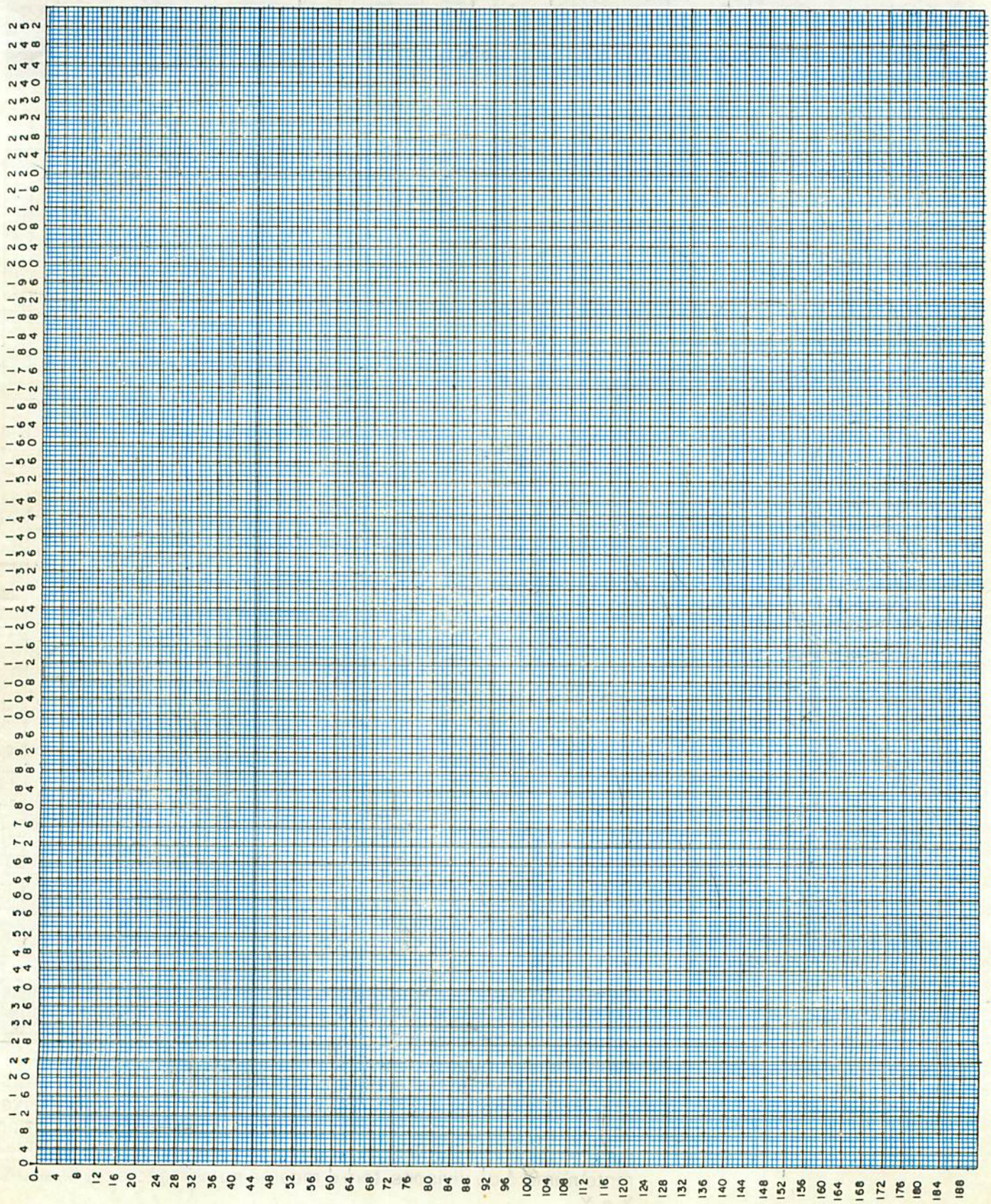
(blocos gráficos na tela de texto)



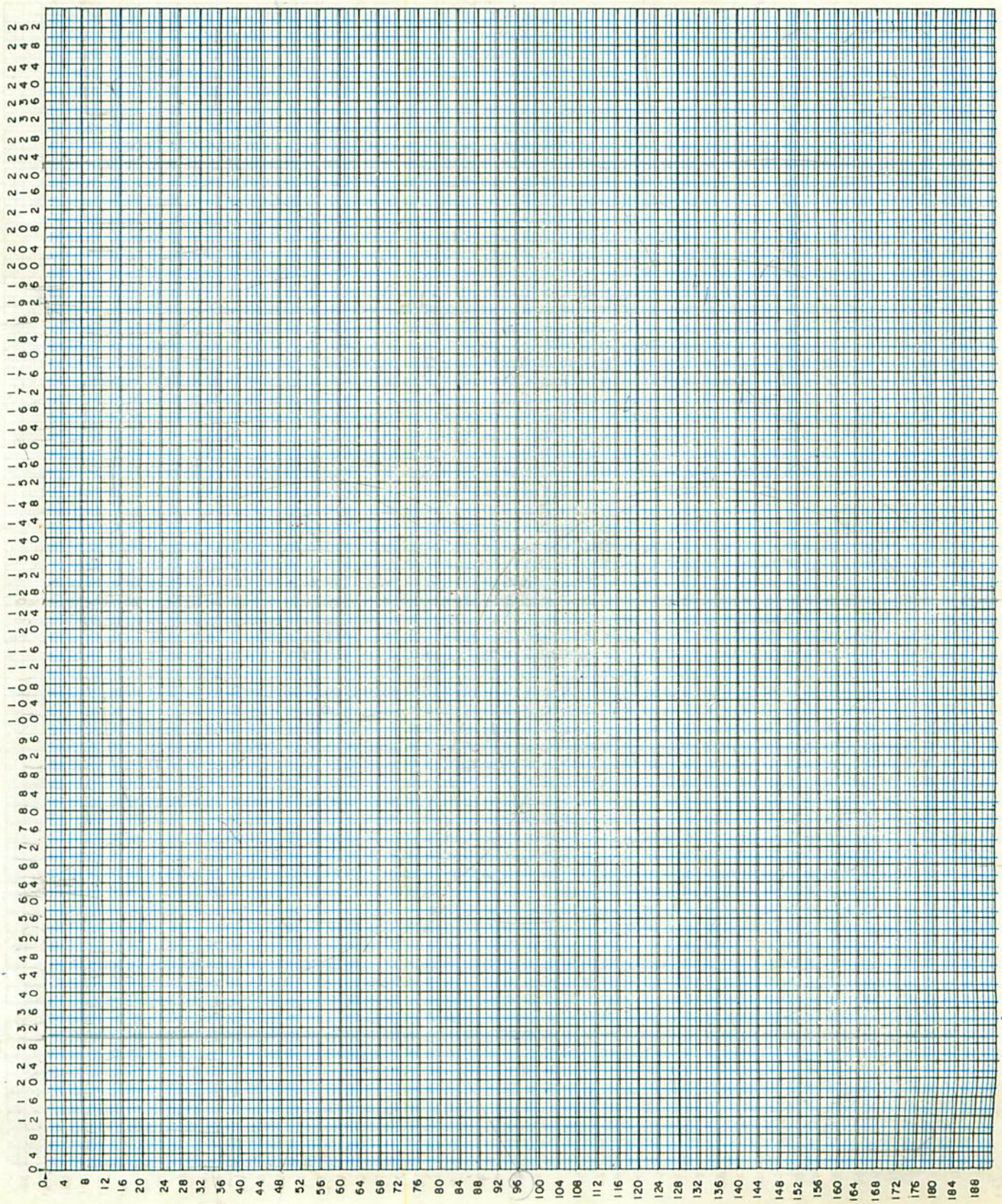
Os recursos estritamente gráficos do CP 400 permitem a escolha entre 5 modos gráficos distintos. Esses modos combinam 3 diferentes níveis de resolução com duas diferentes combinações de cores: duas e quatro cores.

Apesar da variedade de modos possíveis, todas as funções gráficas operam com seus parâmetros sempre dentro da mesma faixa de valores, que corresponde à resolução mais alta (modo 4). Por isso, os 3 quadros que apresentaremos a seguir, apesar do diferente número de divisões, têm a mesma numeração para os eixos X e Y.

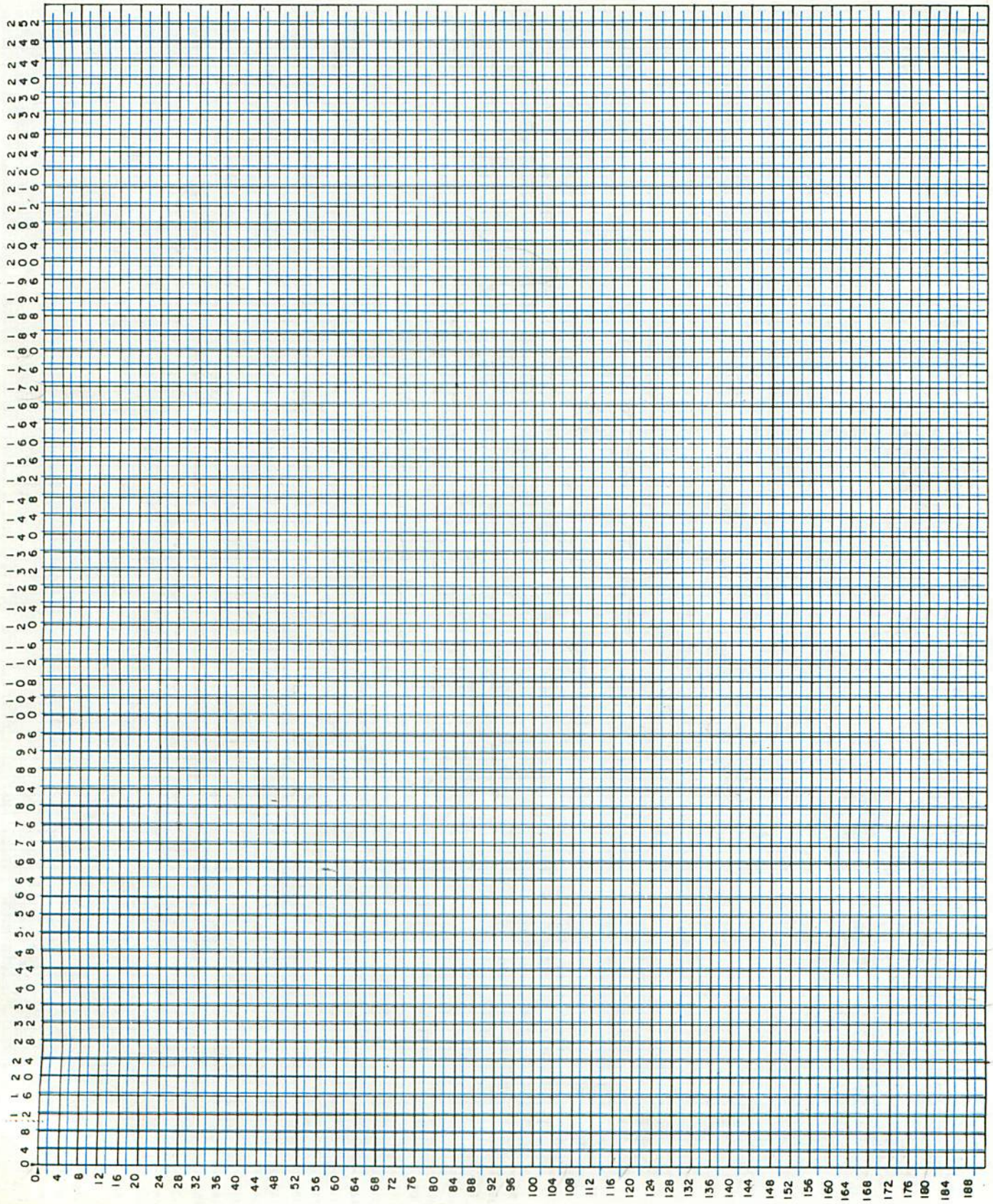
POSIÇÕES GRÁFICAS NA TELA (MODO 4)



POSIÇÕES GRÁFICAS NA TELA (MODOS 3 E 2)



POSIÇÕES GRÁFICAS NA TELA (MODOS 1 E 0)



CÓDIGO ASCII DOS CARACTERES

F

Nota: *Para opção maiúscula/minúscula pressionar as teclas* SHIFT *0 em combinação com os caracteres de A a Z. Os códigos sem* SHIFT *então aparecerão.*

TECLA	HEXADECIMAL		DECIMAL	
	SEM SHIFT	COM SHIFT	SEM SHIFT	COM SHIFT
BREAK	03	03	03	03
CLEAR	0C	5C	12	92
ENTER	0D	0D	13	13
ESPAÇO	20	—	32	32
!	—	21	33	—
"	—	22	34	—
#	—	23	35	—
$	—	24	36	—
%	—	25	37	—
&	—	26	38	—
'	—	27	39	—
(	—	28	40	—
)	—	29	41	—
*	—	2A	42	—
+	—	2B	43	—
,	2C	—	44	—
-	2D	—	45	—
.	2E	—	46	—
/	2F	—	47	—
0	30	—	48	18
1	31	—	49	—
2	32	—	50	—
3	33	—	51	—
4	34	—	52	—
5	35	—	53	—
6	36	—	54	—
7	37	—	55	—
8	38	—	56	—
9	39	—	57	—
:	3A	—	58	—
;	3B	—	59	—
<	3C	—	60	—
=	3D	—	61	—

TECLA	HEXADECIMAL		DECIMAL	
	SEM SHIFT	COM SHIFT	SEM SHIFT	COM SHIFT
>	3E	—	62	—
?	3F	—	63	—
@	40	13	64	19
A	61	41	97	65
B	62	42	98	66
C	63	43	99	67
D	64	44	100	68
E	65	45	101	69
F	66	46	102	70
G	67	47	103	71
H	68	48	104	72
I	69	49	105	73
J	6A	4A	106	74
K	6B	4B	107	75
L	6C	4C	108	76
M	6D	4D	109	77
N	6E	4E	110	78
O	6F	4F	111	79
P	70	50	112	80
Q	71	51	113	81
R	72	52	114	82
S	73	53	115	83
T	74	54	116	84
U	75	55	117	85
V	76	56	118	86
W	77	57	119	87
X	78	58	120	88
Y	79	59	121	89
Z	7A	5A	122	90
↑	5E	5F	94	95
↓	OA	5B	10	91
←	O8	15	8	21
→	O9	5D	9	93

G PALAVRAS E CARACTERES ESPECIAIS

PALAVRAS RESERVADAS

'	CLS	FOR	MID$	PPOINT	SQR
*	COLOR	GET	MOTOR	PRESET	STEP
+	CONT	GOSUB	NEW	PRINT	STOP
-	COS	GOTO	NEXT	PSET	STR$
REM	CSAVE	HEX$	NOT	PUT	STRING$
<	DATA	IF	OFF	READ	SUB
=	DEF	INKEY$	ON	RENUM	TAB
>	DEL	INPUT	OPEN	RESET	TAN
ABS	DIM	INSTR	OR	RESTORE	THEN
AND	DLOAD	INT	PAINT	RETURN	TIMER
ASC	DRAW	JOYSTK	PCLEAR	RIGHT$	TO
ATN	EDIT	LEFT$	PCLS	RND	TROFF
AUDIO	ELSE	LEN	PCOPY	RUN	TRON
CHR$	END	LET	PEEK	SCREEN	USING
CIRCLE	EOF	LINE	PLAY	SET	USR
CLEAR	EXEC	LIST	PMODE	SGN	VAL
CLOAD	EXP	LLIST	POINT	SIN	VARPTR
CLOADM	FIX	LOG	POKE	SKIPF	↑
CLOSE	FN	MEM	POS	SOUND	

SÍMBOLOS DO BASIC

SÍMBOLO	EXPLICAÇÃO
""	Indica que dados entre as aspas são constantes strings.
:	Separa instruções de programa na mesma linha.
( )	Diz ao Computador para desenvolver primeiro as operações que estiverem dentro dos parênteses.

OPERADORES DO BASIC

OPERADORES	CARACTERÍSTICA
+	Adição
+	Combina strings
+	(PLAY)Sustenido
−	Subtração
−	Bemol(PLAY)
−	Imprime sinal de menos depois do número negativo (PRINT USING)
*	Multiplicação
/	Divisão
=	Igual
>	Maior que
<	Menor que
> = ou = >	Maior ou igual a
< = ou = <	Menor ou igual a
<> ou ><	Diferente
↑	Potenciação — eleva um número a uma potência especificada.
AND	Operação AND (Multiplicação Lógica)
OR	Operação OR (Soma Lógica)
NOT	Operação NOT (Inversão Lógica)

CARACTERES DO TECLADO

CARACTERE	CARACTERÍSTICA
←	Retrocede o cursor.
ENTER	Diz ao Computador que o programa ou a linha de comando chegou ao fim (retorno de carro).
BREAK	Pára a execução do programa.
SHIFT @	Suspende a execução do programa. (Pressione qualquer tecla para continuar)
SHIFT 0	Introduz modo maiúsculo/minúsculo.

H UMA FÓRMULA NA MÃO VALE MAIS QUE DUAS NO LIVRO...

TÓPICO	FÓRMULAS	INSTRUÇÃO EM BASIC
Soma dos ângulos de um triângulo	$180° = A + B + C$	TTL = AA + AB + AC
Resolver pela área Dados lado a, ângulos B e C	$A = 180 - (B + C)$ $\text{Área} = \frac{a^2 \text{sen } B \cdot \text{sen } C}{2 \text{ sen } A}$	AA = 180 – (AB + AC) [então converte AA, AB e AC em radianos] AREA = SA↑2*SIN(AB)*SIN(AC)/(2*SIN(AA))
Dado lados a, b e c	$s = \frac{1}{2}(a + b + c)$ $\text{Área} = \sqrt{s(s-a)(s-b)(s-c)}$	S = (SA + SB + SC)/2 AREA = SQR(S*(S – SA)*(S – SB)*(S – SC))
Lei dos Senos	$\frac{a}{b} = \frac{\text{sen } A}{\text{sen } B}$ ou $a = \frac{\text{sen } A}{\text{sen } B} \cdot b$	SA = (SIN(AA)/SIN(AB))*SB
Lei dos Co-senos	$a^2 = b^2 + c^2 - 2bc \cdot \cos A$ ou $a = \sqrt{b^2 + c^2 - 2bc \cdot \cos A}$	SA = SQR(SB↑2 – SC↑2 – 2*SB*SC*COS(AA))
Lei das Tangentes	$\frac{a-c}{a+c} = \frac{\tan \frac{1}{2}(A-C)}{\tan \frac{1}{2}(A+C)}$ ou $\tan \frac{1}{2}(A-C) = \frac{a-c}{a+c} \cdot \tan \frac{1}{2}(A+C)$	Y = TAN((AA – AC)/2) Y = (SA – SC)/(SA + SC)*TAN((AA + AC)/2)
Dados 3 lados, resolver por um ângulo	$s = \frac{1}{2}(a + b + c)$ $r = \sqrt{\frac{(s-a)(s-b)(s-c)}{s}}$ $A = 2 \arctan\left(\frac{r}{s-a}\right)$	S = (SA + SB + SC)/2 R = SQR((S – SA)*(S – SB)*(S – SC)/S) AA = 2*ATN(R/(S – SA))
Equação Quadrática	$ax^2 + bx + c = 0$ $x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$	A*X↑2 + B*X + C = 0 Z = B↑2 – 4*A*C X1 = (–B + SQR(Z))/(2*A) 'SE Z> = 0 X2 = (–B – SQR(Z))/(2*A) 'SE Z> = 0
Equação Algébrica	$(a^x)^y = a^{xy}$ $a^{-x} = \frac{1}{a^x}$ $\log x^y = y \cdot \log x$ $\log xy = \log x + \log y$ $\log \frac{x}{y} = \log x - \log y$	Z = (A↑X)↑Y ou Z = A↑(X*Y) Z = A↑(–X) ou Z = 1/(A↑X) Z = LOG(X↑Y) ou Z = Y*LOG(X) Z = LOG(X*Y) ou Z = LOG(X) + LOG(Y) Z = LOG(X/Y) ou Z = LOG(X) – LOG(Y)

FUNÇÕES DERIVADAS

FUNÇÃO	FUNÇÃO ESCRITA EM BASIC X É EM RADIANOS
Secante	SEC(X) = 1/COS(X)
Co-secante	CSC(X) = 1/SIN(X)
Co-tangente	COT(X) = 1/TAN(X)
Seno Inverso	ARCSIN(X) = ATN(X/SQR(−X*X+1))
Co-seno Inverso	ARCCOS(X) = −ATN(X/SQR(−X*X+1))+1.5708
Secante Inversa	ARCSEC(X) = ATN(SQR(X*X−1))+(SGN(X)−1)*1.5708
Co-secante Inversa	ARCCSC(X) = ATN(1/SQR(X*X−1))+(SGN(X)−1)*1.5708
Co-tangente Inversa	ARCCOT(X) = −ATN(X)+1.5708
Seno Hiperbólico	SINH(X) = (EXP(X)−EXP(−X))/2
Co-seno Hiperbólico	COSH(X) = (EXP(X)+EXP(−X))/2
Tangente Hiperbólica	TANH(X) = −EXP(−X)/(EXP(X)+EXP(−X))*2+1
Secante Hiperbólica	SECH(X) = 2/(EXP(X)+EXP(−X))
Co-secante Hiperbólica	CSCH(X) = 2/(EXP(X)−EXP(−X))
Co-tangente Hiperbólica	COTH(X) = EXP(−X)/(EXP(X)−EXP(−X))*2+1
Seno Hiperbólico Inverso	ARGSINH(X) = LOG(X+SQR(X*X+1))
Co-seno Hiperbólico Inverso	ARGCOSH(X) = LOG(X+SQR(X*X−1))
Tangente Hiperbólica Inversa	ARGTANH(X) = LOG((1+X)/(1−X))/2
Secante Hiperbólica Inversa	ARGSECH(X) = LOG((SQR(−X*X+1)+1/X)
Co-secante Hiperbólica Inversa	ARGCSCH(X) = LOG((SGN(X)*SQR(X*X+1)+1)/X)
Co-tangente Hiperbólica Inversa	ARGCOTH(X) = LOG((X+1)/(X−1))/2

FAIXAS DE ENTRADAS VÁLIDAS

Seno inverso	−1<X<1
Co-seno inverso	−1<X<1
Secante inversa	X<−1 ou X>1
Co-secante inversa	X<−1 ou X>1
Co-seno hiperbólico inverso	X>1
Tangente hiperbólica inversa	X*X<1
Secante hiperbólica inversa	0<X<1
Co-secante hiperbólica inversa	X<>0
Co-tangente hiperbólica inversa	X*X>1

Certos valores especiais são matematicamente indefinidos, mas nossas funções podem fornecer valores inválidos; como por exemplo:
TAN e SEC de 90 e 270 graus e COT e CSC de 0 e 180 graus.
Se pedirmos ao Computador o valor de TAN(90/57.29577951) (o ângulo sempre em radiano), ele retornará um dado valor, sendo que a tangente de 90° é matematicamente indefinida.
Outros valores que não estão disponíveis por estas funções são:

```
ARCSIN (−1) = −PI/2
ARCSIN (1) = PI/2
ARCCOS (−1) = PI
ARCCOS (1) = 0
ARCSEC (−1) = −PI
ARCSEC (1) = 0
ARCCSC (−1) = −PI/2
ARCCSC (1) = PI/2
```

Notar que a informação acima pode não estar completa.

I GLOSSÁRIO DO COLOR BASIC

PALAVRA	CARACTERÍSTICA	EXEMPLO
ABS	Calcula o valor absoluto.	Y = ABS(X)
ASC	Dá o código ASCII do primeiro caractere da string especificada.	A = ASC(T$)
ATN	Dá o arcotangente em radianos.	Y = ATN(X/3)
AUDIO	Conecta ou desconecta saída do gravador ao alto-falante da TV.	AUDIO ON AUDIO OFF
CHR$	Retorna caractere para código ASCII, controle, ou código gráfico.	? CHR$(191)
CIRCLE	Desenha um círculo com centro no ponto (X,Y), raio *r*, cor especificada *c* e razão altura/largura (hw) entre 0 e 4. O círculo pode começar e terminar em pontos especificados (0-1).	CIRCLE (128,96),50,4,1,.5,.75
CLEAR	Reserva e armazena *n* bytes de espaço string.	CLEAR CLEAR 500 CLEAR 100,14000
CLOAD	Carrega arquivo de programa especificado do gravador. Se o nome do programa não é especificado, o primeiro encontrado é carregado (o nome deve ter até oito caracteres/espaços).	CLOAD CLOAD"PROGRAMA"
CLOADM	Carrega programa em linguagem de máquina do gravador. Um endereço de desvio para adicionar ao endereço carregado pode ser especificado.	CLOADM"PROG" CLOADM CLOADM"PROG",1000
CLOSE	Fechar arquivos ou dispositivos.	CLOSE CLOSE-2
CLS	Limpa display com a cor especificada. Se a cor não for especificada, é usado o verde.	CLS CLS(3)
COLOR	Define as cores de fundo e primeiro plano.	COLOR 1,3
CONT	Continua a execução do programa depois de ter usado BREAK ou ter usado a instrução STOP.	CONT
COS	Fornece o co-seno do ângulo medido em radianos.	Y = COS(7)
CSAVE	Salva programas no gravador (o nome do programa deve ter até oito caracteres/espaços).	CSAVE"PROGRAMA" CSAVE"PROGRAMA",A
CSAVEM	Grava um arquivo em linguagem de máquina.	CSAVE X,4E,6F,5F
DATA	Armazena dados em seu programa. Usar READ para atribuir esses dados a variáveis.	DATA 5,3, PERAS DATA PAPEL,CANETA
DEF FN	Define função numérica.	DEF FN (X) = X*3
DEFUSR	Define o ponto de entrada para a função USR.	DEFUSR5 = 45643
DEL	Permite a anulação das linhas do programa. DEL- Anula o programa inteiro. DEL *n* Anula a linha *n*. DEL *n*- Anula todas as linhas a partir de *n*. DEL-*n* Anula todas as linhas até *n*. DEL *n*-*n* Anula todas as linhas entre *n* e *n*.	DEL- DEL 30 DEL 30- DEL-30 DEL30-70
DIM	Dimensiona uma ou mais matrizes.	DIM R(65) ,W(40) DIM AR$(8,25)
DLOAD	Carrega um programa BASIC numa velocidade específica. 0 = 300 baud 1 = 1200 baud	DLOAD X,1
DRAW	Desenha uma linha com: ponto de partida, comprimento e cor específicos.	DRAW "BM100,100;S10;V25;BR25;ND25;XA$;"

PALAVRA	CARACTERÍSTICA	EXEMPLO
	Também desenha em escala, linhas em branco, linhas não atualizadas e executa substrings. Se o ponto de partida não foi especificado, assume-se (128,96) ou a última posição de DRAW é usada.	
EDIT	Permite a edição da linha de programa. *n*C Troca *n* número de caracteres. *n*D Anula *n* número de caracteres. I Permite inserção de novos caracteres. H Anula o resto da linha e permite inserção. L Lista a linha atual e continua editando. *n*Sc Busca pela n-ésima ocorrência do caractere *c*. X Amplia linha. SHIFT ↑ Escapa do subcomando. *n* ESPAÇO Move o cursor *n* espaços à direita. *n* ← Move o cursor *n* espaços à esquerda.	EDIT 25
END	Termina o programa.	END
EOF	Fornece zero se houver mais dados ou −1 se não houver mais dados no arquivo especificado. Para arquivos em fita, f = −1; para arquivos de teclado, f = 0.	
EXEC	Transfere controle para programas em linguagem de máquina no endereço especificado. Se o endereço é omitido, o controle é transferido para o endereço ajustado no CLOADM.	EXEC EXEC 32453
EXP	Retorna o exponencial natural do número ($e^{número}$).	Y = EXP(7)
FIX	Retorna a parte inteira de um número.	Y = FIX(7.6)
FOR...TO STEP/ NEXT	Cria um ciclo no programa que o Computador deve repetir do primeiro ao último número especificado. STEP é usado para incrementar o número cada vez que um ciclo é executado. Se STEP for omitido, é usado o 1.	FOR X = 2 TO 5.NEXT X FOR A = 1 TO 10 STEP5:NEXT A FOR M = 30 TO 10 STEP −5:NEXT M
GET	Lê o conteúdo gráfico de um retângulo em uma matriz para uso futuro por PUT.	GET (5,20) - (3,8),G
GOSUB	Envia o Computador ao início de uma sub-rotina com o número de linha especificado.	GOSUB 500
GOTO	Envia o Computador a um número de linha especificado.	GOTO 300
HEX$	Calcula o valor hexadecimal.	PRINT HEX$(50) Y$ = HEX$(X/20)
IF/THEN	Testa a relação. Se for verdadeira, o Computador executa a instrução seguinte a THEN.	IF A = 5 THEN 30
INKEY$	Verifica no teclado qual tecla foi pressionada.	A$ = INKEY$
INPUT	O Computador pára e espera dados do teclado.	INPUT X$ INPUT "NOME"; N$
INPUT #-1	Entra dados do gravador.	INPUT #-1,A
INSTR	Busca pela primeira ocorrência da string Y$ na string X$ e retorna à posição em que ela ocorreu.	PRINT INSTR(5,X$,Y$)
INT	Converte um número em inteiro.	X = INT(5.4)
JOYSTK	Informa a coordenada horizontal ou vertical do joystick direito ou esquerdo. 0 = horizontal esquerdo 1 = vertical esquerdo 2 = horizontal direito 3 = vertical direito	M = JOYSTK(0) H = JOYSTK(K)
LEFT$	Retorna a parte esquerda da string a partir de uma posição específica.	T$ = LEFT$(S$,5)
LEN	Retorna o número de caracteres de uma string.	X = LEN(SEN$)
LET	Atribui valores a variáveis.	LET A$ = "ANO B"
LIST	Lista linhas especificadas ou programas inteiros na tela.	LIST LIST 20-55 LIST 20 LIST-20 LIST20-

PALAVRA	CARACTERÍSTICA	EXEMPLO
LLIST	Lista linhas de programas ou programas inteiros na impressora.	LLIST LLIST 20-55 LLIST 20 LLIST-20 LLIST 20-
LINE	Desenha uma linha de um ponto inicial a um ponto final. Se o ponto de início é omitido, (128,96) ou o último ponto final é usado. PSET seleciona a cor do primeiro plano, e PRESET seleciona a cor de fundo. A opção *,B* desenha uma caixa usando como extremidades os pontos final e inicial. E *,BF* pintará a caixa com a cor selecionada em PSET.	LINE (5,3) - (6,6) ,PSET LINE -(191,191),PSET,BF
LINE INPUT	Entra linhas de texto pelo teclado.	LINE INPUT "RESPOSTA";X$
LOG	Retorna o logaritmo natural.	Y = LOG(543)
MEM	Fornece a quantidade de memória livre.	PRINT MEM
MID$	Fornece uma substring de outra string. Se a opção comprimento é omitida, a string inteira à direita da posição é retornada.	F$ = MID$(A$,3) PRINT MID$ (A$,3,2)
MID$	Substitui uma parte de uma string por outra string.	MID$(A$,14,2) = "K$"
MOTOR	Liga ou desliga o gravador.	MOTOR ON MOTOR OFF
NEW	Limpa memória.	NEW
ON...GOSUB	Envia para várias sub-rotinas.	ON Y GOSUB 50,100
ON...GOTO	Envia para várias linhas.	ON Y GOTO 190,200
OPEN	Abre arquivos (f) na tela ou teclado (0), fita (−1), impressora (−2) para entrar (I) ou (O).	OPEN "I",0,"ARQUIVO" OPEN "0", −1,"DADOS"
PAINT	Pinta gráficos na tela começando pelo ponto (X,Y) com a cor especificada. Pára na borda da cor especificada.	PAINT(10,30),4,2
PCLEAR	Reserva *n* números de páginas de memória gráfica.	PCLEAR 8
PCLS	Limpa a tela com uma cor especificada. Se o código de cor é omitido, a cor de fundo é usada.	PCLS 3
PCOPY	Copia gráficos de uma página fonte a uma página destino.	PCOPY 5 TO 6
PEEK	Retorna o conteúdo da localização de memória que você especificou.	A = PEEK(32076)
PLAY	Soa uma nota especificada (A-G ou 1-12) oitava (O), volume (V), comprimento da nota (L), ritmo (T), pausa (P), e permite execução de substrings. Também sustenido (# ou +) ou bemol (−).	PLAY "L1;A#;P8;V10;T3;L2;B-;9;XA$;"
PMODE	Seleciona resolução e página de memória para iniciar.	PMODE 4, 3
POINT	Testa se o ponto gráfico especificado está aceso ou apagado, X(horizontal) = 0-63; Y(vertical) = 0-31. O valor retornado é -1, se o ponto está no modo caractere; 0, se está apagado; ou o código de cor, se está aceso.	IF POINT (X,Y) THE PRINT "ACESO" ELSE PRINT "APAGADO"
POKE	Coloca um valor em uma localização de memória específica.	POKE 18572,255
POS	Fornece a posição atual do cursor.	PRINT TAB(8) POS(0)
PPOINT	Testa se o ponto gráfico especificado está aceso ou apagado e fornece o código da cor do ponto especificado.	PPOINT (13,15)
PRESET	Coloca a cor de fundo.	PRESET (5,6)
PRINT	Imprime mensagem especificada na tela de TV.	PRINT "OLA"
PRINT#-1	Grava dados na fita.	PRINT = -1,A
PRINT#-2	Imprime um item ou lista de itens na impressora.	PRINT = -2,CAP$
PRINT TAB	Move o cursor para uma posição de coluna especifica e imprime.	PRINT TAB(5) "NOME"
PRINT USING	Imprime números no formato especificado. #Formata números. .Ponto decimal. ,Coloca vírgula à direita de cada três números. **Preenche espaços da frente com asteriscos.	PRINT USING "####";12;3 PRINT USING "##.#";18.3 PRINT USING "####,";12.0 PRINT USING "**##;##";12.3

PALAVRA	CARACTERÍSTICA	EXEMPLO
PRINT USING (cont.)	$Coloca $ na frente. $$Sinal de dinheiro flutuante. **$Sinal de dinheiro flutuante. +Na primeira posição, ocasiona sinal a ser impresso. Na última posição, ocasiona sinal a ser impresso atrás do número. ↑↑↑↑Formato exponencial. -Coloca o sinal depois de números negativos. !Retorna o primeiro caractere da string. %espaço%Campo da string; o comprimento do campo é o número de espaços mais 2.	PRINT USING "$##.##";12.3 PRINT USING "$$##.#";12.345 PRINT USING "**$#.##";1.234 PRINT USING "+###.#";-123 PRINT USING "##.#";123 PRINT USING "##.#-";-123.4 PRINT USING "!";"AZUL" PRINT USING "% %";"ANIL"
PRINT	Imprime mensagem específica numa localização indicada na tela de texto.	PRINT@246,"OLA" PRINT@246, A$
PSET	Coloca um ponto de uma cor especificada.	PSET(X,Y,C)
PUT	Armazena gráficos de uma matriz na tela. (O tamanho da matriz retangular deve ser igual ao tamanho retangular de GET.)	PUT (3,2)-(5,6),V,PSET
READ	Lê o próximo item na linha DATA e atribui a ele variáveis especificadas.	READ A$ READ C,B
REM ou '	Permite a inserção de comentários na linha de programa. Tudo que vier depois de REM é ignorado pelo Computador.	REM ISTO E' IGNORADO 10 PRINT X:REM ISTO E'IGNORADO
RENUM	Permite renumerar linhas de programas.	RENUM 1000,5,100
RESET	Retira um ponto de uma localização especificada.	RESET (14,15)
RESTORE	Coloca o indicador do Computador de volta ao primeiro item nas linhas DATA.	RESTORE
RETURN	Termina uma sub-rotina e envia o Computador a uma instrução imediatamente após GOSUB.	RETURN
RIGHT$	Retorna a parte direita da string a partir da posição indicada.	ZIP$ = RIGHT$(AD$,5)
RND	Dá um número aleatório entre 1 e um número especificado que deve ser maior que 1.	A = RND(X)
RUN	Executa um programa do início ou a partir de uma linha especificada.	RUN
SCREEN	Seleciona gráficos ou tela de texto e o conjunto de cor.	SCREEN 1,0
SET	Coloca um ponto na tela de texto com uma cor especificada.	SET(14,13,3)
SGN	Retorna o sinal de uma expressão numérica: −1 se o argumento é negativo, 0 se o argumento é zero,. +1 se o argumento é positivo.	X = SGN(A*B)
SKIPF	Pula para o fim do próximo programa da fita, ou para o fim do programa especificado.	SKIPF"PROGRAMA"
SIN	Fornece o seno de um ângulo medido em radianos.	Y = SIN(X)
SOUND	Soa um tom durante um tempo especificado.	SOUND 128,3
STOP	Pára a execução de um programa.	STOP
STRING$	Fornece uma string de caracteres (de comprimento específico) indicados pelo código ASCII ou pelo primeiro caractere da string.	? STRING$(5,"%") ? STRING$(5,91)
STR$	Converte uma expressão numérica em uma string.	S$ = STR$(X)
SQR	Fornece a raiz quadrada de um número.	Y = SQR(5 + 8)
TAN	Fornece a tangente de um ângulo medido em radianos.	Y = TAN(45 + 7)
TIMER	Fornece o conteúdo do temporizador ou permite seu ajuste.	? TIMER TIMER = 0
TROFF	Desativa o traçador de programa.	TROFF
TRON	Ativa o traçador de programa.	TRON
USR*n*	Chama uma sub-rotina em linguagem de máquina do usuário.	Z = USR(Y)
VAL	Converte uma string em um número.	A = VAL(B$)
VARPTR	Retorna o apontador de endereço para uma variável especificada.	Y = USR(VARPTR(X))

ÍNDICE REMISSIVO

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

Fotolitos e Impressão

GRÁFICA EDITORA HAMBURG

Avenida Bogaert, 64
Vila das Mercês São Paulo
Fone: 914-0233
CEP 04298