MX-1600

64K COLOR COMPUTER

MANUAL DE OPERAÇÃO

MX-1600

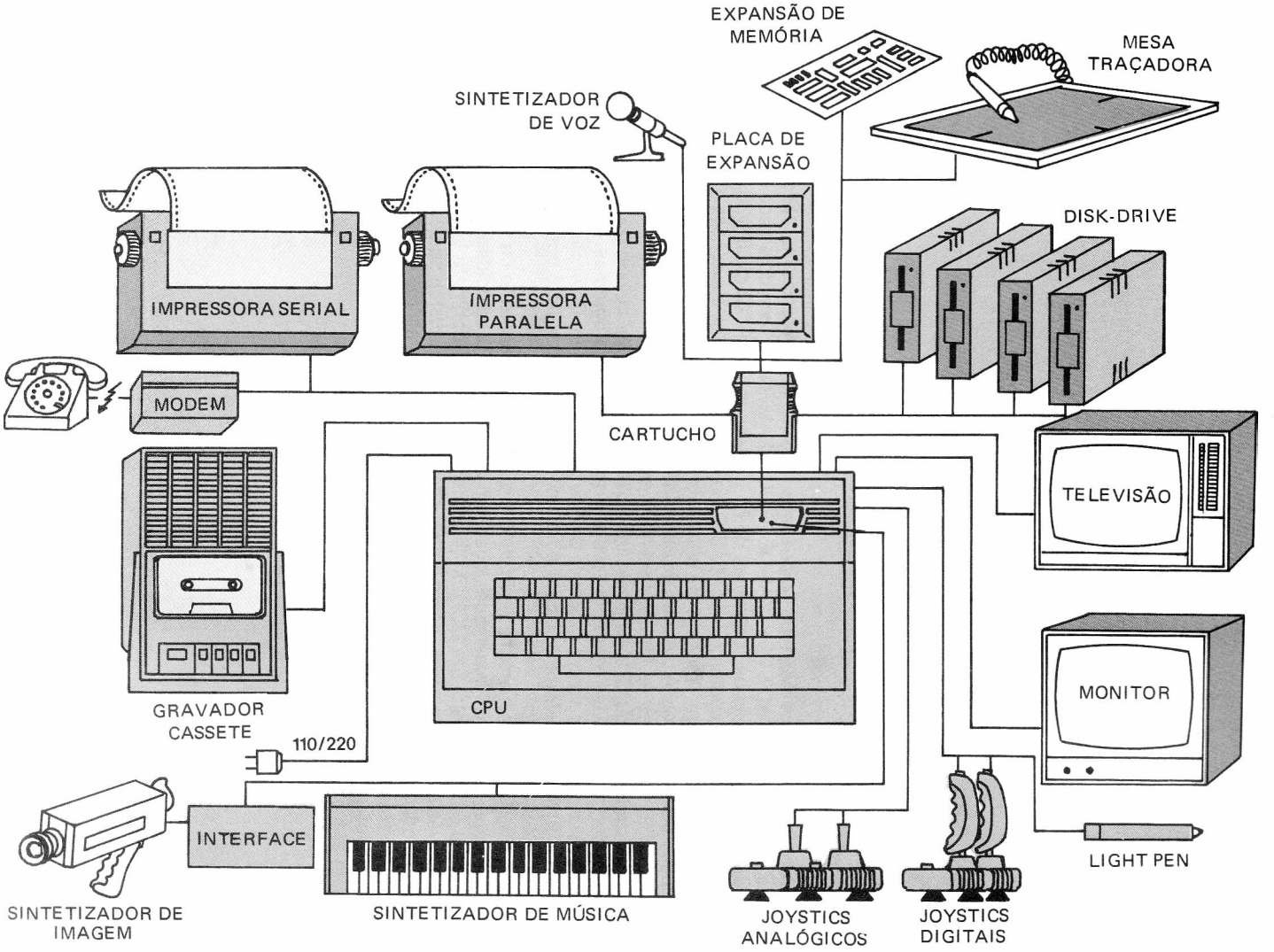
64K COLOR COMPUTER

MANUAL DE OPERAÇÃO

INTRODUÇÃO

Caro Usuário,

O seu Computador Pessoal MX-1600 Color é uma ferramenta poderosíssima. Na sua configuração básica a CPU (UNIDADE CENTRAL DE PROCESSAMENTO), vem dotada de uma memória RAM para armazenamento de programas de 64 K Bytes e uma ROM de 16 K Bytes contendo o sistema operacional e o interpretador BASIC, dos mais completos e extensos até hoje desenvolvidos. É capaz de gerar imagens de alta resolução gráfica com 256 por 192 pixels e com capacidade de até 9 cores. A geração de sons é também das mais complexas, permitindo praticamente sintetizar todas as formas de ondas audíveis.
Para ser utilizado, além da CPU é necessário apenas um televisor colorido ou monocromático e um gravador comum, caso o usuário desejar armazenar seus programas. Isto só já lhe permite desenvolver ou utilizar todo e qualquer programa de linguagem Color-BASIC (Beginners All-Purpose — Symbolic Instruction Code) e linguagem de máquina (ML) que esteja compatível com seu MX-1600 Color.
Na sua configuração mais ampla, o MX-1600 Color permite a conexão de até 4 Disk-Drive de 5 1/4" (de dupla densidade dupla-face), duas impressoras de alta velocidade (uma com interface serial e a outra paralela) um MODEM (Modulador-Demodulador) para intercâmbio de informação através da linha telefônica com outros computadores ou banco de dados (por exemplo, projeto Cirandão da Embratel), dois joysticks analógicos ou digitais (do tipo utilizado em videogames), Monitor de TV de alta resolução, cartuchos com jogos ou programas aplicativos; em resumo, todos os recursos de um sistema verdadeiramente profissional.
Leia com atenção este Manual antes de ligar seu Computador.



CONFIGURAÇÃO DO SISTEMA MX-1600

INDICE

CAPÍTULO 1

Apresentação

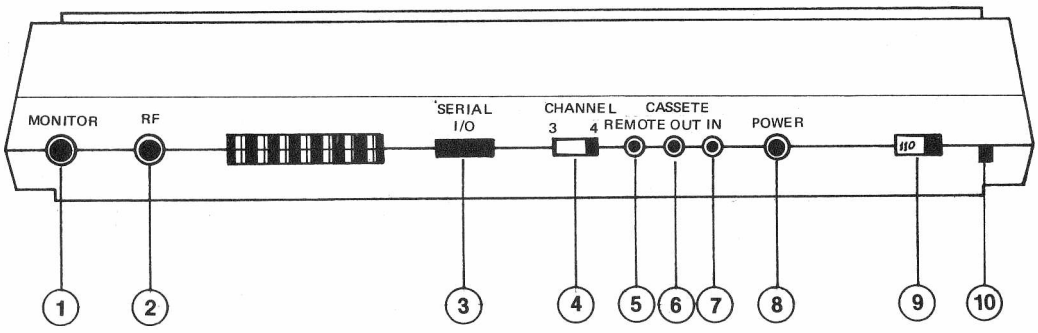
APRESENTAÇÃO

Ao retirar o seu MX-1600 Color da sua embalagem, encontrará, além deste Manual de Operação:

— *Unidade Central de Processamento,* dotada de teclado profissional de 53 teclas, saída para TV padrão PAL-M ou monitor monocromático, e saída para gravador Cassete;
— Duas fitas Cassete contendo aproximadamente 100 programas diferentes;
— Um cabo de RF para ligação do computador à televisão;
— Um cabo composto de três vias para ligação do gravador Cassete ao computador;
— Um cartucho demonstrativo contendo um programa diagnóstico e jogos.

LIGAÇÃO À REDE ELÉTRICA E À TELEVISÃO

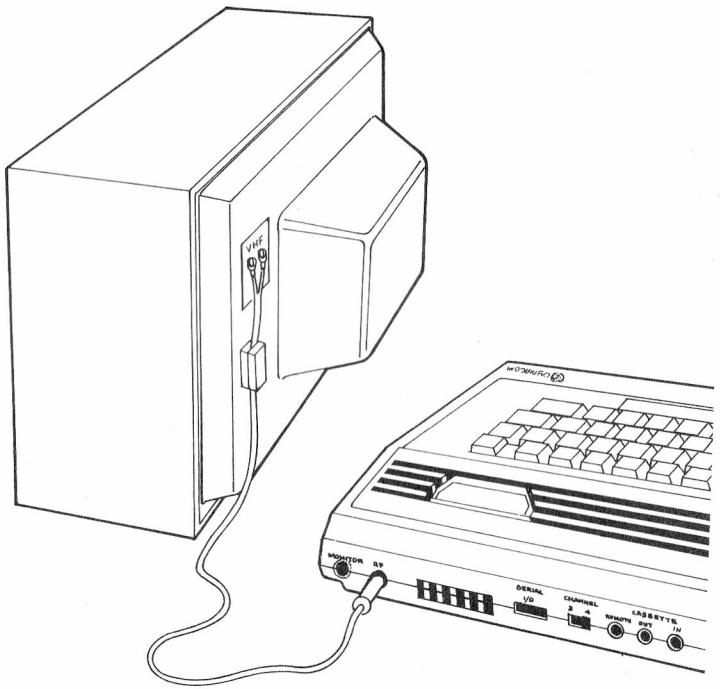
Posicione seu computador MX-1600 perto de um televisor colorido ou branco e preto (neste último caso, não poderá gerar as cores que seu computador dispõe), e perto de uma tomada elétrica. O computador vem ajustada de fábrica para a tensão de 220 volts. Se a voltagem na sua região for de 110 volts, altera, com o auxílio de uma chave de fenda o seletor de voltagem para 110, na parte traseira de seu computador.



① *SAIDA MONITOR* — permite a ligação de monitor de alta resolução, monocromático.
② *SAIDA RF* — permite ligar o MX-1600 a um televisor colorido convencional, através do cabo de RF.
③ *SAIDA/ENTRADA SERIAL* — esta tomada permite ligar periféricos como impressora ou MODEM dotados de entrada serial padrão RS-232C.
④ *CHAVE SELETORA DE CANAL* — permite selecionar entre canal 3 ou 4, para a melhor imagem.
⑤, ⑥ e ⑦ *SAIDA/ENTRADA DO GRAVADOR* — permite ligar qualquer gravador cassete através do cabo do gravador.
⑧ *CHAVE POWER* — esta chave liga e desliga o computador da rede elétrica.
⑨ *CHAVE SELETORA DE TENSÃO* — permite selecionar entre voltagem de 110 volts ou 220 volts, dependendo da tensão da rede elétrica de sua região.
⑩ *CABO DE FORÇA* — deve ser ligado firmemente a uma tomada elétrica.

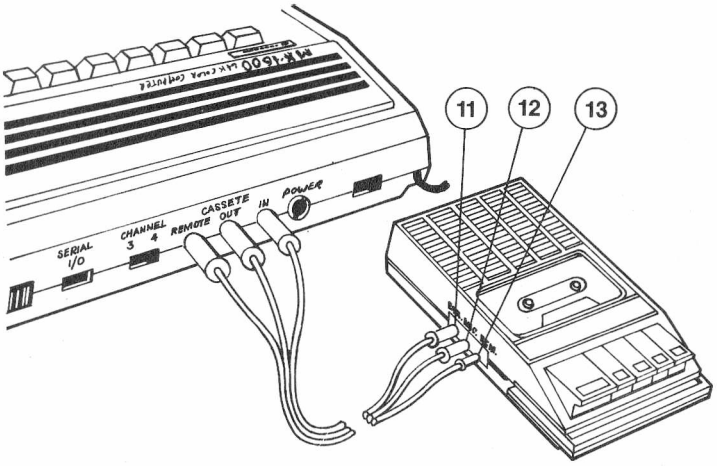
NÃO LIGUE AINDA O COMPUTADOR À REDE ELÉTRICA

Localize a saída de RF na parte traseira de seu computador e introduza nela o cabo de RF. Observe que ao lado desta saída encontra-se outra parecida denominada MONITOR. Esta poderá ser usada para ligar um monitor de alta resolução, porém, monocromático. (O cabo para tal ligação é vendido separadamente).
Na outra extremidade do cabo de RF existe um casador de impedância — com terminais para ligação nos terminais da antena de sua televisão. Portanto, substitua a antena ligada ao seu televisor pelo cabo de RF conforme ilustrações abaixo.



LIGAÇÃO DO GRAVADOR

O gravador não é imprescindível para a operação do seu computador MX-1600 Color. É necessário apenas como meio de armazenamento magnético para seu programa ou para carregar programa e aplicativos já existentes em fita Cassete.
O gravador em princípio poderá ser qualquer tipo daquele utilizado para som, todavia, os cabos de conexões fornecidos junto com seu computador, são para os gravadores portáteis. Siga as ilustrações abaixo para a ligação do gravador ao computador.
Observe que o cabo do gravador é composto por três fios separados sendo um para a gravação dos programas, um para a reprodução (carregamento) do programa de fita para seu computador e o terceiro, para o controle do motor de seu gravador. O gravador poderá operar com pilhas ou na rede elétrica, sendo ainda recomendado a utilização de gravador com contador numérico para facilitar a localização futura de programas.



⑪ *EARPHONE (EAR)*
⑫ *MICROPHONE (MIC)*
⑬ *REMOTE (REM)*

INTRODUZINDO UM CARTUCHO

Através do conector do cartucho, o MX-1600 permite a ligação além de cartuchos de jogos ou aplicativos, outros periféricos, como Disk-Drive, interfaces, programas com linguagem diferente do BASIC como por exemplo o OS-9, o FLEX, CP/M, COBOL, FORTRAN, etc. A entrada do cartucho é dotada de uma porta com fechamento a mola. Para introduzir um cartucho basta pressioná-lo no sentido vertical até o encaixe final no seu conector. Oberve que pelo formato do cartucho só há uma maneira certa de encaixe.

LIGANDO O COMPUTADOR

Após verificar a tensão da rede elétrica e efetuada a ligação ao televisor, podemos proceder e ligar o computador para os primeiros ajustes da televisão:

— Ligue seu televisor selecionando o canal 3 ou o canal 4, dependendo do qual esteja desocupado na sua região.
— Selecione no painel traseiro do seu computador entre o canal 3 ou 4 para corresponder ao canal selecionado na sua TV.
— Mantenha o volume de sua televisão no nível baixo.
— Ligue seu computador.
— Sintonize sua televisão para a melhor imagem, inclusive no que tange o brilho, o contraste e a saturação da cor.

Deverá aparecer uma área quadrada verde com os dizeres:

COLOR BASIC V (V.R)
DYNACOM 1985
OK

(V — significa versão e R — a revisão do seu computador)

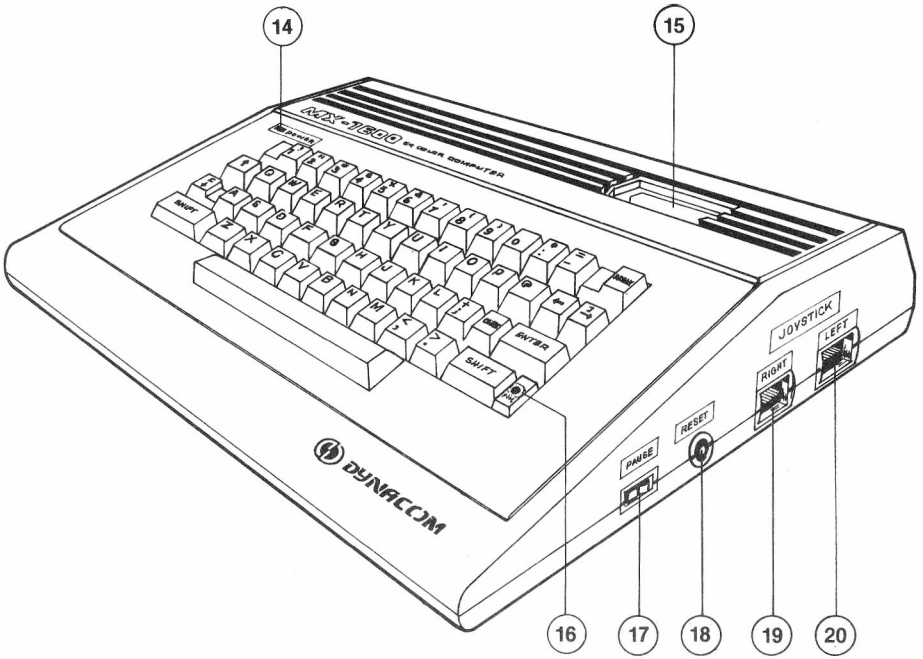
Se você estiver usando um cartucho a imagem será gerada pelo cartucho de acordo com o programa gravado nele.
Se não aparecer qualquer imagem desligue o computador e repita o procedimento. Se o defeito persistir procure verificar a causa provável na tabela abaixo:

TABELA DE DEFEITOS E CAUSAS PROVÁVEIS

DEFEITO	CAUSA PROVÁVEL
O OK (ou mensagem equivalente se você estiver usando um cartucho) não aparece quando se liga o Computador	— Chave PAUSE na posição ON — Não há energia elétrica. Verifique a ligação do cabo de alimentação. — Seqüência incorreta para ligar. — Periférico (por exemplo, a impressora) não está conectado corretamente. Verifique as conexões. — Sintonize novamente o canal. — A tela de seu televisor precisa de ajuste. Verifique o contraste, brilho ou ajuste fino da sintonia. — Verifique se o cabo da antena do Computador está corretamente ligado no seu aparelho de TV.
Recepção pobre ou imagem imprecisa	—Verifique se a sua TV está ajustada no canal apropriado (3 ou 4 — aquele que tiver a melhor imagem). — Verifique as conexões da antena para certificar-se de que foram feitas com segurança e corretamente. — Seu aparelho de TV precisa de ajuste. Verifique os controles de contraste, brilho e ajuste fino.
Seu programa em fita não carrega	— Conexão imprópria do gravador. Verifique as instruções de conexão do gravador, no Capítulo 7. — O volume do gravador está muito alto ou baixo. Verifique o controle de volume do gravador. — Informação na fita pode ter sido alterada devido a descarga elétrica, campo magnético, ou fita deteriorada. Tente carregar a outra cópia.
"Fantasmas" ou recepção de TV e Computador misturada	— Tente usar outro canal de TV (3 ou 4). Verifique se a antena interna está ligada.
O Computador "trava" durante a operação normal, requerendo Reset ou liga desliga	— Flutuações na tensão CA. Veja Energia elétrica, adiante. — Conector instalado incorretamente ou com defeito. Verifique todos os cabos de conexão para ver se estão seguramente ligados e se não estão gastos. — Programação. Cheque o programa. — Chave PAUSE na posição ON.

JOYSTICKS

O seu MX-1600 Color pode operar com dois tipos de joysticks, utilizando os mesmos conectores situados na lateral direita do aparelho, conforme ilustração abaixo:



⑭ *LED POWER* — indica, quando acesso, que o Computador está ligado à rede elétrica.

⑮ *CONECTOR DE EXPANSÃO* — permite o encaixe de cartucho com programa ou expansão com periféricos (Disk-Drive, Memória adicional, etc).

⑯ *LED PAUSE* — indica, quando acesso que a chave PAUSE está ligada.

⑰ *CHAVE PAUSE* — permite interromper qualquer programa por tempo indeterminado. Não deve ser acionada quando em operação de gravação ou reprodução (carregamento de programas).

⑱ *TECLA RESET* — utilizada para reinício de programa em cartucho ou na RAM.

⑲ e ⑳ *ENTRADAS PARA JOYSTICKS* — permitem ligar dois joysticks analógicos ou, dois joysticks digitais, sendo um para o jogador da direita e o outro, da esquerda.

Os joysticks analógicos que permitem movimentação de 360° são recomendados para determinados jogos com movimentação de cursor que necessitam posicionamento preciso e permanente na tela. (exemplo jogo Missile Command). Os joysticks digitais são os mesmos utilizados em videogames e são recomendados para jogos com ação rápida para uma das suas 8 posições possíveis (exemplo jogo PAC-MAN).
O encaixe dos cabos dos joystick é simples e só permite encaixe de uma só maneira.
Observe que o MX-1600 permite a conexão de dois joysticks simultaneamente, sendo um para jogador da direita e o outro para o da esquerda.

TECLAS ESPECIAIS

A chave Pause — chave de pausa situa-se fora da área do teclado propositalmente, por te aplicação muito específica. Sua função é interromper a execução de qualquer programa ou jogo, por tempo indeterminado, sem todavia alterar o programa, dado ou a imagem da tela que fica "congelada" durante a pausa. O teclado e qualquer outra chave deixa também de operar durante a operação de pausa.

A chave de Pausa é de duas posições (ligada/desligada) sendo indicado por um Led seu estado operacional "ligado".
O usuário deve sempre garantir que a chave pausa esteja desligada antes de ligar o computador ou iniciar qualquer programa.

A TECLA RESET

A função da tecla RESET é reiniciar um programa em cartucho ou um programa armazenado na RAM. A tecla RESET se situa na lateral direita do aparelho e poderá ser utilizada com a mesma função da chave liga/desliga, porém sem alterar o *programa* que esteja armazenando na memória do computador. Todavia, dados e variáveis são perdidos ao acionar a tecla RESET.

A TECLA SHIFT

Existem duas teclas SHIFT situadas nas extremidades baixas do teclado. Estas teclas tem a mesma função daquelas de máquina de escrever que selecionam entre maiúsculas e minúsculas ou entre caracteres situados na parte superior das teclas. Por exemplo para digitar um "!" pressione SHIFT e em seguida 1 sem liberar a tecla SHIFT.

O BASIC não reconhece nenhum comando digitado em letras minúsculas. Portanto tudo que você digitar vai ser em maiúsculas. Mas, se você precisa de letras minúsculas para um relatório a ser impresso numa impressora, por exemplo, você pode utilizá-las. Para tanto, basta digitar SHIFT 0, e tudo o que você digitar será interpretado internamente e armazenado com 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 de maiúsculas, digite novamente SHIFT 0.

A tecla SHIFT além das funções já descritas desempenha outras, quando utilizada em conjunto com certas teclas como mostra o quadro a seguir:

SHIFT 0 Muda o modo de introdução (só maiúsculas/maiúsculas e minúculas).

SHIFT → Introduz um colchete direito (])

SHIFT ↓ Introduz o 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;

A TECLA CLEAR

A tecla CLEAR limpa a tela e posiciona o cursor no canto superior esquerdo. Também fica cancelada a atual linha digitada.

A TECLA ENTER

Determina para o computador a conclusão de uma linha de instrução. O computador não interpreta nenhum comando até que essa tecla seja pressionada.

A TECLA BREAK

Interrompe o programa em andamento e posiciona o computador para aceitar novos comandos pelo teclado.

A TECLA ←
Cancela o último caracter digitado, posicionando o cursor na posição anterior. Esta tecla pode ser utilizada repetidamente para apagar uma palavra ou uma linha inteira.

A TECLA ↑
Sinal de potenciação. Quando utilizado em conjunto com o número permite elevar este número a uma potência específica.

AJUSTE DE COR
Mesmo sem experiência anterior em programação, digite o programa abaixo seguindo a risca esta ordem:

NEW ENTER

5 CLS 8 ENTER

10 FOR X= 0 TO 63 ENTER

20 FOR Y= 0 DO 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 introduziu certo, digite RUN e **ENTER**.
Este programa gera oito barras coloridas e verticais com a seguintes ordens:
Verde, Amarelo, Azul, Vermelho, Cinza, Cyan (Azul esverdeado), magenta e laranja.
Utilize novamente os ajustes da sua televisão para obter uma imagem mais próxima dessas cores. Para interromper o programa pressione a tecla **BREAK**.

AJUSTE DE SOM

Coloque o volume da TV em posição normal de audição e digite este programa seguindo a risca sua ordem:

NEW ENTER

10 FOR X = 1 TO 255 ENTER

20 SOUND X,1 ENTER

30 NEXT X ENTER

Verifique novamente o volume e digite:

RUN ENTER

CAPÍTULO 2

Primeiros Passos de programação

Este capítulo ensinará você os primeiros passos de programação. Ligue seu computador. No fim da mensagem aparecerá:

OK

OK é o "pronto" do Computador. Ele está dizendo: "OK, eu estou pronto". Ele espera pacientemente por um comando seu. Você é quem manda. Você pode dizer a ele qualquer coisa que quiser.
Dê a ele seu primeiro comando. Digite exatamente o seguinte:

PRINT "OI, EU SOU SEU COMPUTADOR PESSOAL"

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

Quando você chegar ao fim da linha na tela, continue digitando. O resto da mensagem aparecerá na próxima linha.

Agora verifique sua linha. Você colocou as aspas onde nós indicamos? Se você cometeu algum erro, não há problema. Simplesmente pressione a tecla ← e o último caracter que você digitou desaparecerá. Pressione novamente e o penúltimo também desaparecerá. E assim por diante. Se for tudo bem em sua tela deve aparecer:

OK
PRINT "OI, EU SOU SEU COMPUTADOR PESSOAL"
OI, EU SOU SEU COMPUTADOR PESSOAL
OK

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

PRINT "5"

Pressione **ENTER**... O Computador novamente obedece e na tela aparece:

5

Tente outra:

PRINT "5+5" ENTER

O Computador obedece:

5+5

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

PRINT 5+5 ENTER

Então o Computador imprime a resposta:

10

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

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

Você chegou a alguma conclusão?

> ""O Computador considera as aspas como um jornalista. Se a instrução 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. Instrução entre aspas é chamada de STRING.

FAZENDO CONTAS

Qualquer problema de matemática é fácil para seu Computador. Vamos fazer uma divisão longa. Digite:

PRINT "1234 DIVIDIDO POR 6.5 E' " 1234/6.5 ENTER

Deixe-o fazer também um problema de multiplicação:

PRINT 1985*32 ENTER

Observe 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 "16*2 = " 16*2 ENTER
PRINT 18*18 "E' O QUADRADO DE 18" ENTER
PRINT 22.5/11.18 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.

DENTRO DAS NORMAS

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

PPRINT "TUDO BEM?" ENTER

O Computador responde:

? SN ERRO

SN ERRO indica um erro de "sintaxe". Este é o modo do Computador dizer: "O comando PPRINT não existe no meu vocabulário e 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, ou na ordem certa.
O Computador apresentará 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 6/0 ENTER

O Computador responde:

?/0 ERRO

Que significa: "Não posso 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 causa esta mensagem.

CORES, PLEASE!

Até agora tudo que você viu seu Computador fazer foi imprimir em fundo 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.

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** e assim que você obtiver um OK em sua tela. Digite:

CLS (7) ENTER

Agora você deverá ter a cor magenta em sua tela com uma faixa verde no topo. Experimente mais algumas cores. Use qualquer número de 0 a 8. Seu Computador tem 9 cores. Cada cor tem um código numérico. Se digitar um número maior que 8 terá a mensagem de erro.

?FC ERRO

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 voltando a cor original verde.

PARA OS AMANTES DA 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. Experimente!

SOUND 128,1 ENTER

E o Computador reproduzirá o tom durante aproximadamente 6/100 de um segundo. Experimente com 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, se não, receberá a mensagem de erro: ?FC ERRO.

RELEMBRANDO O "SHIFT"

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 "OI" 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.

O COMPUTADOR TEM MEMÓRIA DE ELEFANTE

Uma das coisas que faz seu Computador poderoso é sua capacidade em se lembrar de tudo que você pede a ele. Para fazer o Computador lembrar do número 15 por exemplo, diga a ele que:

X = 15 ENTER

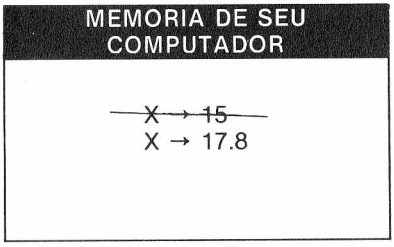
Agora digite qualquer coisa que você quiser para desviar a atenção dele, depois pressione ENTER. Para ver se o Computador lembra o que X significa, digite:

PRINT X ENTER

Ele se confundiu? Ou se esqueceu?

Faça então o seguinte:

X = 17.8 ENTER



Se você já conhece BASIC, atenção.
O MX-1600 não requer o comando LET.

Agora se você introduzir PRINT X, ele imprimirá o número 17.8. Veja no quadro acima 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 = 25 ENTER
B = 20 ENTER
AZ = 15 ENTER

Agora digite:

PRINT X, Y, B, AZ

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

Por exemplo:

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

Para o Computador o cifrão identifica uma string.

Veja se seu Computador está confuso. Digite:

PRINT X$, Y$, B$, AZ$ ENTER

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

NUMEROS	STRINGS
X → 17.8	X$ → "TENTE
Y → 25	Y$ → "LEMBRAR"
B → 20	B$ → "ISTO A SEU"
AZ → 15	AZ$ → "COMPUTADOR"

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

PRINT AZ ENTER

para ver se AZ ainda corresponde a 15.

Tente fazer 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 NÃO ABRE MÃO DE SUAS NORMAS

Você acha que o Computador aceitará estas linhas:

X = "8" ENTER
Y = "GUARDE ESTES DADOS" ENTER

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

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

Veja no quadro as normas que você ignorou.

NORMAS SOBRE STRINGS

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

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

X$ = "12" ENTER
Y$ = "GUARDE ESTES DADOS" ENTER

E o Computador as aceita.

Você acha que o Computador aceita a linha a seguir?

X$ = 12 ENTER

Estas são as normas que seu comando ignorou:

NORMAS SOBRE NUMEROS

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

Digite deste modo, assim o Computador aceita.

X = 12 ENTER
Y = 22 ENTER

Assim tudo foi guardado na memória de seu Computador.

MEMORIA DE SEU COMPUTADOR

Números	Strings
X → 12	X$ → "12"
Y → 22	Y$ → "GUARDE ESTES DADOS"

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

PRINT X*2 ENTER

O Computador imprime o produto de X vezes 2.

O Computador ainda lembra que x = 12

Tente esta linha:

PRINT Y/X

O Computador imprime o resultado de Y dividido por X.

Digite:

PRINT X$*2 ENTER

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

NORMAS SOBRE VARIAVEIS

Você pode usar dois caracteres quaisquer de A - Z para uma variável. O primeiro caracter 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 caracter $ depois dela. De outro modo ela será considerada um Número.

CAPÍTULO 3

O Computador às suas ordens

Digite:

NEW ENTER

para limpar qualquer coisa que estiver na memória do Computador. Agora digite esta linha (Não esqueça o número 10 — é muito importante.):

10 PRINT "OI, 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:

20 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 "OI, 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:

OI, EU SOU SEU COMPUTADOR
QUAL E' O SEU NOME?

Responda a questão do Computador e pressione ENTER... O que? 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 a linha novamente. O Computador só considera a última linha 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 "OI," A$ ENTER

Agora liste o programa. Digite:

LIST ENTER

Seu programa deve estar assim:

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

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

Rode o programa novamente usando outros nomes:

```
OI, EU SOU SEU COMPUTADOR
QUAL E' O SEU NOME?
? 011-227-5144
OI, 011.227.5144
OK
OI, EU SOU SEU COMPUTADOR
QUAL E' O SEU NOME?
? AGORA ENTENDI
OI, AGORA ENTENDI
OK
```

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

MEMORIA DO SEU COMPUTADOR

~~A$ JAIME~~
~~A$ 011-227-5144~~
A$ AGORA ENTENDI

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 "OI, EU SOU SEU COMPUTADOR"
20 PRINT "QUAL E' SEU NOME?"
30 INPUT A$
40 PRINT "OI" 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 nos chamamos de ciclo. (Outros preferem o termo *loop*, que quer dizer a mesma coisa). A única maneira de parar com esse ciclo vicioso é usando a tecla **BREAK**.

ILUMINANDO O 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.***

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

Digite 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 oi 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 PONTUACAO DA INSTRUCAO 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.*

DANDO UM TOQUE FINAL AO PROGRAMA

Os programadores profissionais talvez achem que utilizar **BREAK** não seja a melhor forma de interromper a execução do programa. Por que então não fazer com que o Computador educadamente, pergunte se queremos que ele pare? Mude as linhas 40 e 50 do programa acima para a seguinte forma.

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

Veja o que estas linhas fazem.
A linha 50 simplesmente imprime uma pergunta.
A linha 60 manda o Computador para 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 com outras instruções.

FAZENDO CONTAS

Neste capítulo vamos criar alguns efeitos sonoros. Mas vamos antes ensiná-lo a contar. Digite isto:

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

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

```
10 FOR X = 16 TO 21
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 = 16
X = 17
X = 18
X = 19
X = 20
X = 21
```

Adicione esta outra linha entre FOR e NEXT.

15 PRINT "...CONTANDO..."

Rode-a. Seu Computador executa qualquer linha que você inserir entre FOR e NEXT, para cada contagem. Escreva um programa para o Computador imprimir o seu nome 14 vezes.

Dica: O programa deve contar até 14

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

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 14 20 PRINT "CARLA" 30 NEXT X	10 FOR X = 1 TO 10 20 PRINT "9*"X"="9*X 30 NEXT X

CONTAR EM DOIS

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

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

Digite **RUN**... Você viu o que o STEP 2 fez? Ele fez o Computador contar de 2 em 2. A linha 10 diz ao Computador que:

- O primeiro X é 2
- O último X é 10

... e STEP 2...

- Todos os X entre 2 e 10 de 2 em 2, isto é, 2, 4, 6, 8 e 10 (STEP 2 diz ao Computador para somar 2 para obter o próximo valor de X.)

Experimente mais algumas linhas FOR... STEP para você ver exatamente como funciona; altera as seguintes linhas no lugar da linha 10.

```
10 FOR X = 5 TO 80 STEP 10
10 FOR X = 10 TO 2 STEP —2
```

CONTANDO COM MÚSICA

Agora que você ensinou o Computador a contar, você pode adicionar à contagem acompanhamento tonal.
Apague seu programa anterior e digite este:

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

Este programa fez o Computador contar de 1 até 255 (em passos unitários). Para cada valor, ele executa a ordem das linhas 20 e 30.
- Linha 20 — imprime a contagem atual de X.
- Linha 30 — dá o tom correspondente a X.

Exemplo:

A primeira vez que o Computador passa pelo FOR, na linha 10, fez o 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.
- etc.

O que acontece se você alterar a 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.*

Você tentou? Agora usando STEP, altere a linha 10 para que o Computador gere os tons seguintes:
(1) De baixo para cima, a intervalos de 10 notas.
(2) De cima para baixo, a intervalos de 10 notas.
(3) Do meio para cima, a intervalos de 5 notas.

Confira a resposta:

```
10 FOR X = 1 TO 255 STEP 10
10 FOR X = 255 TO 1 STEP—10
10 FOR X = 128 TO 255 STEP 5
```

Agora veja se você pode escrever um programa que faça o Computador gerar os tons:
(1) De cima para baixo, e depois
(2) De baixo para cima.

NOS EMBALOS DA MÚSICA

Agora você já pode mostrar ao seu Computador como fazer coisas importantes: dizer as horas e cantar (bom, *cantar* desafinado, é claro!)
Digite este exercício simples:

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

Rode o programa. Seja paciente e espere alguns segundos. Para ser exato seu Computador gasta 2 segundos para contar até 920.
Linhas 10 e 20 fornecem uma "pausa" de tempo.
Fazendo o Computador contar até 920, ele mantém o Computador ocupado por 2 segundos. Como você pode ver, isto parece um cronômetro. Apague o programa e digite este outro programa:

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

Rode o programa introduzindo os segundos que você quer contar.
Vamos adicionar algumas linhas ao nosso programa para que no fim soe um alarme. Aqui está o programa.

```
10 PRINT "QUANTOS SEGUNDOS"
20 INPUT S
30 FOR Z = 1 TO 460*S
40 NEXT Z
50 PRINT S "SEGUNDOS PASSARAM"
60 FOR T = 100 TO 180 STEP 2
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
```

Note que a instrução GOTO que colocamos no fim do programa mantém a mensagem e o alarme até que o programador pressione, impaciente a tecla **BREAK** *ou* **SHIFT** **@** para pará-lo.

MARCANDO HORA

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

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

Digite RUN... Aparecerá em sua tela:

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

Chama-se um contador dentro de outro contador ou *loop* dentro de outro *loop* (o que você preferir). Programadores chamam isto de *"loop* 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 Y seguido do de X.

Sempre quando se coloca um *loop* dentro de outro, deve-se fechar o *loop* interno antes de fechar o *loop* externo.

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

Com esta norma que você aprendeu, o Computador pode fazer muito mais coisas. Digite este programa:

```
10 FOR S = 0 TO 59
20 PRINT S
30 SOUND 180, 2
40 FOR T = 1 TO 390
50 NEXT T
60 NEXT S
70 PRINT "PASSOU UM MINUTO"
```

Rode o programa... Eis 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 exato para passar 1 segundo

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

Agora adicione esta linha para limpar a tela:

15 CLS

Rode de novo o programa. Desta vez ele:
1 — Conta os segundos de 0 a 59 (linhas 10 e 60).

E para cada segundo:
a) Limpa a tela (linha 15)
b) Imprime o sgundo (linha 20)
c) Gera um tom (linha 30)
d) Pára um tempo exato para passar um segundo (linhas 40 e 50)

2 — Quando termina a contagem, imprime a mensagem "passou um minuto" (linha 70)

Com esse conhecimento adquirido vamos construir um relógio verdadeiro:

Digite este programa:

```
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 180,2
70 FOR T = 1 TO 375
80 NEXT T
90 NEXT S
100 NEXT M
110 NEXT H
```

Isto é, em resumo o que o Computador faz neste programa:
I — Conta as horas de 0 a 23 (linha 10)

Para cada hora nova que conta:
A. Conta os minutos de 0 a 59 (linha 20)
Para cada minuto novo que conta:
(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 exato para passar um segundo (linhas 70 e 80)
(2) Quando termina de contar os 60 segundos, retorna à linha 20 para o próximo minuto (linha 100).
B. Quando termina de contar todos os 59 minutos, retorna à linha 10 para a próxima hora (linha 110).

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

Se adicionarmos a linha 120 GO-TO 10 o relógio funcionará sem parar.

SÓ FALTA CANTAR

Finalmente vamor ensinar o Computador a cantar. Consulte o Apêndice D onde temos uma tabela de tons musicais, que mostra o número do tom do Computador para cada nota do teclado musical. Por exemplo, o tom n.º 89 do Computador corresponde ao "Dó".

Digite:

```
5 FOR X = 1 TO 2
10 SOUND 125,8
20 SOUND 108,8
30 SOUND 89,8
40 NEXT X
```

Rode o programa. Estas são as primeiras três notas da nossa "obra" musical. Não lhe falta uma pausa?
Acrescente estas linhas:

```
35 FOR Y = 1 TO 230
36 NEXT Y
```

Rode o programa novamente. Parece melhor?

Vamos acrescentar mais uma parte.
Digite:

```
50 FOR X = 1 TO 2
60 SOUND 147,8
70 SOUND 133,4
80 SOUND 133,4
90 SOUND 125,8
100 FOR Y = 1 TO 230
110 NEXT Y
120 NEXT X
```

Rode o programa. Gostou da música? Pode terminá-la ou escreva outra.

CAPÍTULO 4

Decisões Decisões...

Vamos ver uma situação fácil onde o Computador deve tomar decisões:
Se (IF) você digitar AZUL... *então* (THEN) a tela deve ficar azul, ou *Se* (IF) você digitar VERMELHA... *então* (THEN) a tela deve ficar vermelha. Fácil, não é?
Digite este programa (não se esqueça de usar NEW para limpar a memória):

```
10 PRINT "VOCE QUER A TELA AZUL OU VERMELHA?"
20 INPUT C$
30 IF C$ = "AZUL" THEN 100
40 IF C$ = "VERMELHA" THEN 200
100 CLS(3)
110 END
200 CLS (4)
```

C$ = AZUL

C$ = VERMELHA

Rode o programa algumas vezes, tentando várias 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 vermelha (200 CLS). A linha 110 se encarrega disso. Quando o Computador executa a linha 110, ele não executa mais nada. É o fim do programa para ele. Por outro lado, se você digita vermelha, a linha 40 manda o Computador para a linha 200. Essa linha deixa a tela vermelha, e não é preciso nenhuma instrução END porque não tem mais programa depois da linha 200.
E se eu não digitar nem vermelha 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 vermelho. O que aconteceu? A tela ficou azul, certo? Você sabe por quê?

> *Se a condição não for cumprida (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 simplesmente para a próxima linha.*

Você pode melhorar o programa e fazer com que o Computador peça uma nova instrução no caso de você não digitar AZUL ou VERMELHA. Acrescenta estas duas linhas e veja o que acontece:

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

Agora vamos ver se você pode fazer mais uma alteração no programa.
No lugar do programa terminar quando a tela ficar azul ou vermelha, mude-o para que repita a pergunta sobre a cor que você quer.

> *Dica: Você precisa mudar a linha 110.*

Seu programa ficou assim?

```
10 PRINT "VOCE QUER A TELA AZUL OU VERMELHA?"
20 INPUT C$
30 IF C$ = "AZUL" THEN 100
40 IF C$ = "VERMELHA" THEN 200
50 PRINT "VOCE DEVE DIGITAR AZUL OU VERMELHA"
60 GOTO 20
100 CLS(3)
110 GOTO 10
200 CLS(4)
210 GOTO 10
```

Tente acompanhar o programa pela listagem.

Observe que nas instruções IF/THEM só ocorrerá o desvio se a condição for cumprida. Já nas instruções GOTO, sempre ocorrerá o desvio indicado.

NORMAS SOBRE IF/THEN E GOTO

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

NÚMEROS ALEATÓRIOS

Existe uma função em BASIC, chamada RND, què 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 escolhe e imprime um número qualquer entre 1 e 10.

Rode novamente, e outro número será mostrado, e assim por diante. Observe a seqüência em que os números aparecem. Você notará que **não** há nenhuma regra para os números escolhidos. Eles são aparentemente casuais, ou aleatórios, como preferimos chamá-los. Altere a linha 10 do programa e acrescente a linha 20 a seguir:

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

Rode o programa e pressione **SHIFT** **@** ou **BREAK** quando quiser pará-lo. Observe 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, é só mudar a linha 10 para:

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

Vamos agora ver alguns jogos simples. Utilize sua imaginação para incrementá-los ou inventar seus próprios.

ROLETA RUSSA

Neste jogo de "Roleta Russa" nosso revólver tem 10 balas. Você tem que adivinhar qual é a bala fatal.

```
10 PRINT "ESCOLHE UMA DAS BALAS (1-10)"
20 INPUT X
30 IF X = RND(10) THEN 100
40 SOUND 200,1
50 PRINT "CLICK"
60 GOTO 10
100 PRINT "BANG — VOCÊ ESTÁ MORTO"
```

Rode o programa e veja o que acontece. Vamos melhorar a rotina da linha 100. Digite:

```
100 FOR T = 1 TO 50 STEP 5
110 PRINT ".....BANG"
120 SOUND T,1
130 NEXT T
140 CLS
150 PRINT @ 230, "DESCULPE, VOCE MORREU"
160 PRINT @ 390, "A PROXIMA VITIMA, POR FAVOR"
```

Atenção para as linhas 150 e 160. Ambas usam PRINT @. Vamos ver o que elas fazem. No apêndice D há uma tela quadriculada que representa 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. DESCULPE, VOCE MORREU começa na posição 230 (224+6). A PROXIMA VITIMA, POR FAVOR começa na posição 390 (384+6). Usando essas posições na linha PRINT @ nós dizemos ao Computador onde **exatamente** queremos as mensagens.

Utilize a instrução PRINT @ para planejar o aspecto final do seu impresso na tela

JOGO DE DADOS

Para o próximo jogo você terá, ántes de começar, ensinar o Computador a jogar dados. Para jogar par de dados, o Computador deve gerar dois números aleatórios de 1 a 6. Digite então o seguinte 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 @ 390, "VOCE CONSEGUIU UM" R
```

```
80 PRINT @ 454, "VOCE QUER JOGAR DE NOVO?"
90 INPUT R$
100 IF R$ = "SIM" THEN 10
```

Rode o programa e veja o que acontece. A linha 10 limpa a tela. Na linha 20 o Computador escolhe um número aleatório para o 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 apresentada na tela.
Na linha 90 você informa se continua jogando ou não. Se você digitar um SIM, o Computador roda os dados novamente, caso contrário, o programa termina.

PINTE SUA TELA

Já que você aprendeu a utilizar as cores, vamos agora dar mais "vida" aos nossos programas. Já está na hora de começarmos a usar um poderoso recurso de seu Computador: os **gráficos coloridos.** Digite este programa:

```
10 CLS(0)
20 SET(0,0,3)
30 SET (31,14,4)
40 SET (63,31,5)
45 GOTO 45
```

Não se preocupe com a linha 45. Isso será explicado mais adiante. 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 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 apresentada uma tela quadriculada chamada "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 valores diferentes para SET e PRINT @. Por isso, existem duas telas quadriculadas no apêndice. Para este caso utilize o chamado de Posições Gráficas na tela.*

Agora, vamos ver para que serve a linha 45 (GOTO 45). Tire essa linha do programa. Basta digitar 45 e pressionar ENTER. Depois disso rode o programa.
O que aconteceu? Os pontos não apareceram. De fato, 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 que não aconteça o fim do programa e a conseqüente mensagem OK, é que usamos a linha 45 GOTO 45. Ela serve para bloquear o andamento do programa. Quando encontra essa linha, o Computador entra em *loop* até que você pressione BREAK.

DE PONTO EM PONTO

Até agora, você gerou pontos distantes de diferentes cores. Mas, para gerar mais de um ponto no mesmo bloco, você precisa fazer uma pequena alteração no programa.

Digite este programa:

```
10 CLS(0)
20 SET(32,14,3)
30 SET(33,14,3)
40 GOTO 40
```

Ao rodar esse programa, vão aparecer dois pontos azuis lado a lado na tela. Na realidade, a impressão é que tem um só ponto alongado. Vamos tentar mudar a cor do ponto da direita para vermelho. Altere a linha 30 para:

30 SET (33,14,4)

Agora você está esperando que, ao executar o programa, apareçam dois pontos, um azul e o outro vermelho no centro da tela.
Rode o programa... Os dois pontos ficaram vermelhos?! Por quê será?

Consulte novamente 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
localização	32	14
localização	33	14
localização	32	15
localização	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 significa que um bloco só pode ter duas cores por vez: o preto de fundo; e uma das demais oito cores (1 a 8).

Por isso é que o seu programa gerou os dois pontos vermelhos.

Quando o Computador executou a linha 20, ele de fato gerou um ponto azul na tela, pois todos os outros pontos do grupo não existiam (pretos). Aí, quando ele executou a linha 30, para colocar um ponto vermelho nesse mesmo grupo, ele teve que deixar todos os ponto com a mesma cor e, nesse caso, ele fica sempre com a última cor escolhida, o vermelho.

Agora que já descobrimos por que os pontos tiveram a mesma cor, vamos tentar fazer aparecerem pontos de cores diferentes, juntos. Digite isto:

30 SET(31,14,4)

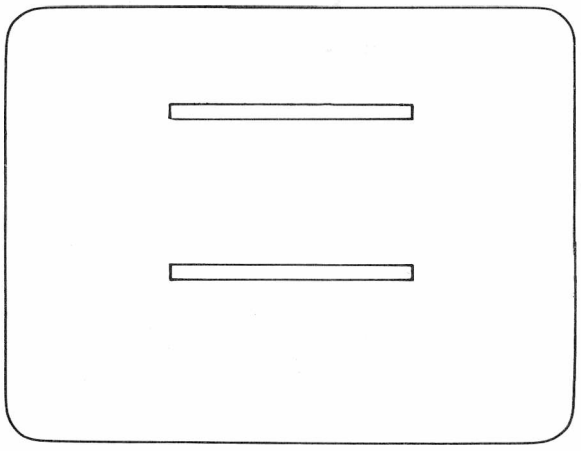
Deu certo! Como o ponto (31,14) está num grupo diferente do anterior (32,14), o Computador o faz vermelho sem alterar a cor do outro.

O COMPUTADOR TEM CARA?

Com o conhecimento anterior você pode desenhar alguma coisa na tela. Vamos começar com um rosto na cor cinza bem simples. Primeiro os contornos superior e inferior da cabeça.

```
5 CLS(0)
10 FOR H = 17 TO 50
20 SET(H,5,5)
30 SET(H,25,5)
40 NEXT H
50 GOTO 50
```

Rode o programa. 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 17 até 50. H representa a posição horizontal dos pontos traçados nas linhas 20 e 30.
Os contornos laterais serão feitos por uma outra parte do programa:

```
50 FOR V = 7 TO 22
60 SET(17,V,5)
70 SET(50,V,5)
80 NEXT V
90 GOTO 90
```

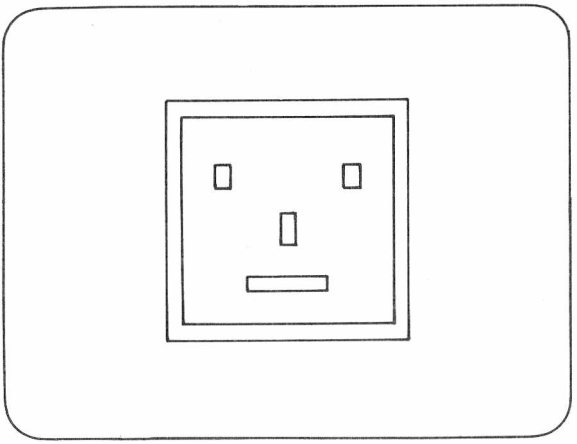
Observe que nós começamos a modificar desde a linha 50.

O programa a seguir gera o nariz vermelho, a boca laranja 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 sua tela é alguma coisa parecida com:



NUM PISCAR DE OLHOS

Acrescentando algumas linhas podemos fazer o Computador piscar.

Digite as seguintes linhas:

```
170 FOR T = 1 TO 400
180 NEXT T
190 RESET(42,10)
200 RESET(22,10)
210 FOR X = 1 TO 200
220 NEXT X
230 GOTO 150
```

Rode o programa. Observe que foram colocadas 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 desenvolver outras figuras, mais complexas. Ou ainda, usando a função RND faça seu computador compor uma música, e a cada tom abrir e fechar a boca, exibindo uma cor ao acaso.

PONTO MÓVEL

Usando SET e RESET, podemos movimentar ou animar figuras. Digite este programa para mover um ponto pela tela:

```
5 CLS(0)
10 FOR H = 3 TO 60
20 SET(H,15,7)
30 RESET(H,15)
40 NEXT H
```

Lembre sempre de apagar o programa anterior usando NEW.

Todo ponto colocado pela linha 20 é apagado pela linha 30. Vamos acrescentar algumas linhas para fazer o ponto retornar:

```
50 FOR H = 60 TO 3 STEP —1
60 SET(H,15,7)
70 RESET(H,15)
80 NEXT H
```

Para mover o ponto de um lado para o outro sem parar, acrescente esta linha:

```
90 GOTO 10
```

Você vai notar que o ponto mexe muito rápido. Vamos acalmá-lo um pouco, com as seguintes linhas:

```
30 IF H > 4 THEN RESET(H—1,15)
70 IF H> 59 THEN RESET(H+1,15)
```

O sinal > significa o mesmo que em matemática, maior que. O sinal <, menor que.

SE TIVER UM JOYSTICK...

Se você tem joysticks, analógicos ou digitais, existe uma infinidade de opções de uso para eles. Se ainda não os conectou, faça-o agora. Simplesmente encaixe-os nos conectores correspondentes na lateral do Computador. Eles só se encaixam de uma única maneira, não há como errar. Digite agora este programa demonstrativo curto:

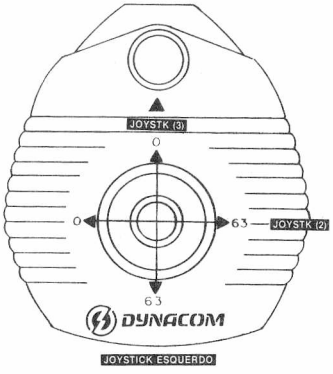
```
10 CLS
20 PRINT @ 0, JOYSTK (0);
30 PRINT @ 5, JOYSTK (1);
40 PRINT @ 10, JOYSTK (2);
50 PRINT @ 15, JOYSTK (3);
60 GOTO 20
```

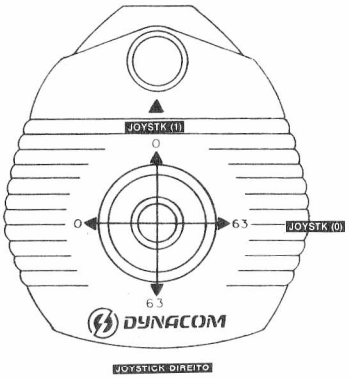
É muito importante colocar os pontos e vírgulas (;) no final das linha 20, 30, 40 e 50.

Ao rodar o programa você vai ver quatro números que se alteram 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 número muda de 0 a 63, passando por valores intermediários.

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 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 número se altera passando de 0 a 63.

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:





Cancele agora a linha 20 e mude a 60 para:

60 GOTO 30

Rode o programa e experimente mover qualquer joystick. Nenhum dos números aparece. 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 alterar 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

e rode o programa.

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

JOYSTICK VIROU PINCEL

Digite este programa:

```
10 CLS(0)
20 H = JOYSTK(0)
30 V = JOYSTK(1)
40 IF V > 31 THEN V = V-32
80 SET(H,V,3)
90 GOTO 20
```

Rode o programa e veja que, ao movimentar o joystick direito, uma linha será traçada na tela. A linha azul acompanhará os movimentos que você fizer. (Tome cuidado para não fazer movimentos muito rápidos, pois a verificação dos joysticks não é muito ágil).
Uma observação importante é o truque usado nas posições verticais das instruções SET (foi usado V=V-32). 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.

Digite este programa:

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

> *< = significa menor ou igual a.*

Ao mover o joystick esquerdo para a direita o traço fica amarelo (C=2). Um movimento para a esquerda desse mesmo joystick muda a cor para azul (C = 3).

E os botões de tiro? Como o Computador sabe quando apertamos estes botões? Vamos introduzir essas linhas no programa e ver o que acontece:

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

Agora digite:

RUN 100 **ENTER**

Esta instrução diz ao Computador para executar o programa a partir da linha 100. Seu Computador vai entrar em um *loop* e ficar imprimindo 255 ou 127.
A instrução PEEK manda o Computador ler 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 valor contido nesse endereço de memória muda para 126 ou 254.

> *Se você apertar os botões sem que o programa esteja sendo executado, podem aparecer mensagens como @ ABCDEFG ou HIJKLMNO. Ignore-as.*

Aperte o botão do joystick esquerdo. O conteúdo deve mudar para 125 ou 253. A partir desses valores, você pode mandar o Computador tomar qualquer decisão durante um jogo ou mesmo durante uma aplicação séria. Por exemplo, podemos mandar o Computador limpar a tela se você apertar o botão direito. Digite estas linhas para alterar as linhas 110 e 120:

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

Elimine a linha 90 e acrescente esta linha:

```
130 GOTO 20
```

Rode o programa. Comece a fazer suas pinturas. Quando quiser começar uma nova figura pressione o botão do joystick direito.

Para estas tarefas aconselhamos o uso dos joysticks analógicos. Os joysticks digitais (do tipo para videogame) responde tão rápido que o computador só consegue verificar para qual das oito direções você aponta, sem todavia alocar um valor específico a esta posição.

CAPÍTULO 5

Atenção! Gravando

Você logo estará escrevendo longos e mais poderosos programas. Talvez você já o faz, e certamente irá querer usá-los novamente. Como fazer isso se quando desligamos o Computador, o programa é perdido?
Neste capítulo você aprenderá como guardar seus programas num gravador cassete.

Praticamente qualquer gravador cassete à pilha ou à luz pode ser conectado ao seu MX-1600. O que pode variar são os nomes das tomadas onde serão feitas as ligações e o volume de reprodução, que revemos mais adiante. As tomadas (*jacks*) que usaremos no gravador são três:

• Controle remoto do motor do gravador. Em geral, nos gravadores portáteis esta tomada é indicada pela palavra REMOTE ou as vezes REM. O Computador usa esta tomada para controlar o motor durante a gravação ou reprodução. Ela tem um furo menor que as outras.

• A tomada que recebe as informações que o Computador envia para gravação, tem o nome de AUXILIAR, MIC (em gravadores portáteis) ou AUX. Quando usado para gravar um programa na fita, o Computador deve ser ligado como qualquer fonte de música a esta entrada.

A tomada da qual o Computador receberá a informação durante a reprodução da fita em geral leva o nome EARPHONE (EAR), em gravador portáteis, ou LINE OUT. Se esses nomes não coincidirem com os de seu gravador, procure no manual deste identificá-los pelas funções que realizam.

A) 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 nas tomadas atrás de seu Computador, e no gravador da seguinte forma:
- O plugue menor na tomada REMOTE (ou REM), do gravador e REMOTE no MX-1600.
- O plugue maior na tomada AUX. (LINE IN ou MIC), do gravador e OUT no MX-1600.
- O plugue na tomada EARPHONE (LINE OUT ou EAR), do gravador e IN no MX-1600.

3. Ligue o gravador à rede elétrica.

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

• 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 de gravá-lo em fita.

• Coloque a fita no gravador. Retorne-a para o início e pressione PLAY e RECORD do gravador, ao mesmo tempo.

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

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

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

O gravador vai começar a gravar, e você estará transferindo o programa que digitou no Computador para uma fita, como se fosse uma música! Observe a tela. Quando o Computador imprimir:

OK

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

REPRODUZINDO

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

1. Primeiro rebobine a fita (coloque-a no início), e certifique-se que os cabos do gravador estão ligados como ilustrado anteriormente.

2. Pressione o botão PLAY do gravador. Ajuste o volume do gravador para nível médio da sua escala.

3. Digite NEW e pressione **ENTER** para limpar qualquer programa da memória do seu Computador.

4. Use o comando CLOAD com o nome do seu programa. Exemplo:

CLOAD "NOME" **ENTER**

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

S

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 F seguida pelo nome do programa, assim:

F 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 outros 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.
Se você tentar carregar um programa cujo nome não está na fita, o Computador não vai parar de procurar. Nesse caso você deve usar a tecla **RESET**.

ARMAZENANDO MAIS DE UM PROGRAMA

Para gravar vários programas na mesma fita, você deve certificar-se de não gravar um programa por cima de outro. Eis uma forma fácil para posicionar a fita no final de seu último programa:

1. Rebobine a fita para seu início.
2. Pressione a tecla PLAY do gravador.
3. Digite SKIPF e o nome do último programa que você gravou na fita. Por exemplo, se o último programa for intitulado "NOME", digite:

SKIP "NOME" **ENTER**

O Computador irá notificá-lo quando encontrar o programa chamado NOME. Quando ele chega ao fim deste programa, o motor do gravador pára e na tela aparece:

OK

4. Sua fita, uma vez posicinada, pressione as teclas PLAY e RECORD do gravador, dê um nome ao novo programa e grave-o (CSAVE).
5. Se vocêr 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 ao fim da fita, vai aparecer uma mensagem I/O ERRO. Não se preocupe com ela. Nesta altura você teria visto o nome de todos os programas e com certeza reconheceu aquele que queria. De posse deste nome você dá o comando SKIPF outra vez para posicionar a fita convenientemente. Mas antes, não se esqueça de rebobinar a fita.

DICAS PARA UMA BOA GRAVAÇÃO

Eis algumas dicas para fazer boas gravações:
• Quando você não estiver usando o gravador para gravar ou carregar um programa, não deixe as teclas RECORD e PLAY pressionadas. Libere-as com a tecla STOP.
• Não tente regravar uma fita já usada pelo Computador. Apesar do processo de gravação apagar a informação anterior, os sinais residuais são suficientes para confundir a nova gravação. Se você pretende usar a mesma fita uma segunda ou terceira vez, utilize um apagador de fita de boa qualidade para limpeza completa.
• Para proteger seus programas, você pode quebrar as linguetas de proteção da fita. Assim, mesmo uma pressão acidental da tecla RECORD não afetará a fita.

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

CAPÍTULO 6

Um professor muito paciente

Seu Computador vem dotado de todos os atributos de um professor nato. Afinal, ele é paciente, incansável e consciente. Dependendo do programador (estamos falando de você, é claro), ele pode ser imaginativo, afetivo e bastante entusiasmado.
Portanto, vamos aproveitar! Nós podemos usar RND para que o Computador nos treine em um problema de matemática atras do outro. 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 você a tabuada de multiplicação, de 1 a 15 e verificará sua resposta. A partir deste programa, você pode fazer com que o Computador efetue tabelas de qualquer operação. Tente a tabela de soma de 1 a 100.
Para fazer o programa um pouco mais interessante podemos instruir o Computador para manter uma contagem total de todas as respostas corretas.
Acrescente estas linhas:

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

T conta quantas questões foram feitas pelo Computador. Quando você roda o programa pela primeira vez, T é igual a zero. Aí, toda vez que o Computador passar pela linha 15, ele adiciona 1 a T.

Quando você liga o Computador, ou quando digita NEW ENTER, *todas as variáveis numéricas são iguais a zero.*

C faz a mesma coisa. Só que marca o número das respostas corretas. Como ela está na linha 95, o Computador não incrementará C a menos que você acerte a resposta. Há muitas maneiras de deixar este programa mais interessante. Acrescente mais algumas linhas ao programa para que o Computador

1 — Chame você pelo nome.
2 — Indique sua resposta correta com um som e um show de luzes.
3 — Imprima o problema e mensagens atraentes em sua tela (Utilize PRINT @).
4 — Mantenha contagem final da porcentagem das respostas corretas.
5 — Encerre o programa se você obtiver 10 respostas, em uma série, corretas.

Use sua imaginação.

VAMOS CONSTRUIR O VOCABULÁRIO DELE

Para criar 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 acrescente 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 — Alocar à UVAS uma 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 de novo? Digite:

```
60 GOTO 10
```

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

```
50 RESTORE
```

> *Lembra como fazer para o Computador parar a execução de um programa? Acione a chave PAUSE ou pressione* **SHIFT** **@**. *Pressione qualquer tecla para que ele continue.*

Agora cada vez que ele passa pela linha 50, fica como se o Computador não tivesse lido nada. O Computador lerá toda a lista de novo. O interessante das linhas DATA é que você pode colocá-las em qualquer lugar que desejar no programa.

AGORA VAMOS MELHORAR O SEU VOCABULÁRIO

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

	(PALAVRAS)	(DEFINIÇOES)
10 DATA	TACITURNO,	FALA POUÇO
20 DATA	LOQUAZ,	FALADOR
30 DATA	VOCIFERRANTE,	BERRANTE
40 DATA	CONCISO,	BREVE
50 DATA	EFUSIVO,	ILUSTRATIVO

Agora vamos fazer com que o Computador escolha uma palavra ao acaso da lista. Bem... há 10 itens na lista. Vamos ver se funciona desta forma:

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

Rode-o várias vezes para ver se funciona. Ele não funciona do jeito que queremos. O que nós queremos é que o Computador escolha ao acaso uma palavra dos itens 1, 3, 5, 7 ou 9. Felizmente, existe uma palavra que explica isso ao Computador. Digite:

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

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

Observe como a linha 65 funciona. 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 a resposta corresponde à verdade, 5 é igual a 5, o Computador completa a instrução THEN e faz N igual a 9 (isto é, 10-1).
Por outro lado, se o Computador escolher 9, ele faz o seguinte cálculo:

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

Como a resposta não corresponde à 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 aleatória ele deve também ler a definição da palavra. Você pode criar esta condição adicionando estas linhas no fim do programa:

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

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

5 CLEAR 100

Adicione, também, estas linhas no fim do programa, para dar ao Computador espaço livre para outros Strings:

130 RESTORE
140 GOTO 60

É desta forma que o Computador escolhe uma nova palavra aleatória e sua definição de uma lista de dados qualquer.

Se você quiser, adicione mais palavras e definições à linha DATA.

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

1 — Imprima somente a definição.
2 — Solicite a você a resposta.
3 — Compare sua resposta com a palavra aleatória correta.
4 — Lhe diga se a resposta está correta. Se não, imprimir a resposta certa.

Para variar este programa você pode tentar Estados e Capitais, cidades e municípios, palavras estrangeiras e seus significados. Tem outras idéias?

Aqui está o nosso programa:

```
5 CLEAR 300
10 DATA TACITURNO, FALA POUCO
20 DATA LOQUAZ, FALADOR
30 DATA VOCIFERANTE, BERRANTE
40 DATA CONCISO, BREVE
50 DATA EFUSIVO, ILUSTRATIVO
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 COM A MATEMÁTICA

Solucionar complicadas fórmulas matemáticas com alta velocidade e precisão é uma área onde seu Computador é brilhante. Mas antes de introduzir você nestas loucas fórmulas matemáticas, veremos alguns caminhos mais curtos e dicas para resolvê-las, as quais, acreditamos, você gostaria de conhecer. Um modo fácil de manipular fórmulas é usando as sub-rotinas. Digite e rode este programa:

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

A linha 20 diz ao Computador para ir à sub-rotina que começa na linha 500. O RETURN diz ao Computador para voltar à instrução BASIC que vem imediatamente após o GOSUB. Elimine a linha 40 e veja o que acontece quando rodar o programa. 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 porque tirando o END na linha 40 este erro ocorre? No início o Computador segue o programa da mesma forma que fez antes que você cancelar a linha END. Ele foi para a sub-rotina na linha 500 por causa de GOSUB e retorna para a linha imediatamente após a conclusão GOSUB, isto é, linha 30.
Então, quando você tirou a linha END, ele foi 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 a esta sub-rotina por um GOSUB.
Aqui está a sub-rotina que eleva um número à potência que você quiser;

```
10 INPUT "DIGITE UM NUMERO";N
20 INPUT "DIGITE A POTENCIA A QUAL SERA' ELEVADO";P
30 GOSUB 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
```

Observe que introduzimos algumas coisas novas nesse programa. Verifique 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 algo novo — 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 continua lembrando você o que ele faz, sem que isto, todavia altere em nada o desenrolar do programa.

DÊ UMA MÃO AO SEU COMPUTADOR

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

Divida a substração de 22—6 por 8

Você espera que o Computador resolva este problema desta forma:

$$\frac{22 - 6}{8} = \frac{16}{8} = 2$$

Mas seu Computador resolve de outra forma. Vamos ver. Digite esta linha de instrução:

PRINT 22 — 6/8 ENTER

Não deu certo? Acontece que o Computador segue suas próprias regras.

REGRAS ARITIMETICAS

O Computador resolve os problemas aritméticos na seguinte ordem:

1. *Primeiro soluciona qualquer operação de multiplicação e divisão.*

2. *Por fim, 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.*

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

No problema acima o Computador segue estas regras. Ele resolveu a operação de divisão primeiro (6/8 = 0,75) e aí então a subtração (22 − 0,75 = 21,25). Para conseguir que o Computador resolva o problema de modo diferente, você deve usar parênteses, em adição ou subtração. Digite esta linha:

PRINT (22 — 6)/8 ENTER

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

REGRAS DE PARENTESES

1. Quando o 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 ai vai para os mais externos.

VAMOS POUPAR

Como você aprendeu no último capítulo, podemos colocar as fórmulas complicadas em sub-rotinas. O programa a seguir usa essas duas sub-rotinas. Ele serve para aqueles que depositam mensalmente a mesma quantia na poupança.

```
10 INPUT "SEU DEPOSITO MENSAL";D
20 INPUT "TAXA ANUAL DO BANCO";I
30 I = I/12* . 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
```

Observe que o programa tem uma sub-rotina "chamando" outra sub-rotina. Isto é perfeitamente possível com um GOSUB, que manda o Computador para cada sub-rotina e um RETURN em cada sub-rotina para retornar à instrução BASIC imediatamente após o GOSUB.
Consulte o Apêndice H. É provável que você queira incluir algumas fórmulas no seu programa.

PALAVRAS CRUZADAS

Uma das maiores virtudes de seu Computador é seu talento com palavras. Ele pode misturar,

combinar ou separar palavras incansavelmente e do jeito que você quiser. Como conseqüência deste talento você pode ensiná-lo a ler, escrever e até desenvolver um diálogo descente. Para iniciantes, o que pensa disto:

```
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
```

Impressionado? LEN diz ao Computador para calcular o comprimento (length) da sua sentença — string S$. Sendo obediente, seu Computador conta cada caracter da sentença, incluindo os espaços e a pontuação.
Eis aqui outra habilidade. Apague seu programa e digite este para fazê-lo compor um "poema":

```
10 A$ = "UMA FLOR"
20 B$ = " "
30 C$ = "PARA"
40 D$ = A$ + B$ + C$
50 E$ = "UMA FLOR"
60 F$ = D$ + B$
70 G$ = "DA MESMA COR"
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** — Adicione esta linha em seu programa e rode:

```
95 I$ = H$ + H$ + H$ + H$ + H$ + H$ + H$
```

Quando você roda este programa, o Computador imprime OS ERRO NA 95. 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 outra vez. Preste atenção 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 separá-las. Digite e rode este programa.

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

Na linha 10 você coloca uma **string** em L$. Vamos dizer que a string seja a palavra DYNACOM. 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$ (L$,1)D	YNAC	OM RIGHT$ (L$,2)

Adicione esta linha ao programa:

```
5 CLEAR 500
```

Veja agora se consegue alterar as linhas 20 e 30 para contar número diferente de letras.

Lembra como se limpa um programa? Digite NEW **ENTER**

Agora vamos colocar 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$)
40 INPUT X
50 PRINT "O TRECHO DA SENTENCA COMECARA' NO CARACTER"X
60 PRINT "DIGITE UM NUMERO DE 1 A"LEN (S$) — 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 algumas vezes para ver se você entendeu a função de MID$.
Eis aqui como o programa funciona. Como sentença digite "AQUI ESTÁ UMA STRING"

S$ → AQUI ESTA' UMA STRING

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 caracter número 8 e tem 5 caracteres de comprimento.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
A	Q	U	I		E	S	T	A	'		U	M	A		S	T	R	I	N	G

5

MID (S$,8,5)

Aqui está mais um exemplo que tem MID$. Apague seu programa e digite:

```
10 INPUT "DIGITE UMA SENTENCA";S$
20 INPUT "DIGITE UMA PALAVRA NA SENTENCA";I$
30 L = LEN (I$)
40 FOR X = 1 TO LEN (S$)
50 IF MID$ (S$,X,L) = I$ THEN 90
60 NEXT X
```

```
70 PRINT "SUA PALAVRA NAO ESTA' NA SENTENCA"
80 END
90 PRINT "---- COMECA NO CARACTER NUMERO" X
```

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

```
I$ — — — AQUI ESTA' UMA STRING
I$ — — — U M A
L — — — 3
```

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

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

MID$ (S$,1,3)

Na quarta vez que passar pelo *loop,* quando X for igual a 4, o Computador vê a seguinte forma:

MID$ (S$,4,3)

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

SEU COMPUTADOR PODE SER UM RIGOROSO EDITOR

Você está começando a enxergar no Computador alguém que pode corrigir seus textos. Ele pode ser programado para ajudá-lo na escrita e lhe economizar horas de datilografia.
Imagine que você tem uma frase que você quer adicionar à:

10 A$ = "MUDAR UMA SENTENCA"

Introduza estas palavras no início da sentença:

E' FACIL

Faça o programa imprimir esta nova sentença:

E' FACIL MUDAR UMA SENTENCA

Conseguiu? Este é o nosso programa:

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

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

1. Encontre o início do trecho (MID):

UMA SENTENCA

2. Cancele esse trecho formando uma nova string:

E FACIL MUDAR

3. Acrescente estas palavras no fim da string:

QUALQUER COISA QUE VOCE QUEIRA

Dica: Para formar a string E' FACIL MUDAR, você precisa obter o trecho esquerdo da sentença E' FACIL MUDAR UMA SENTENCA.

Consegui?
Eis a 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$,X,Y) = "UMA SENTENCA" THEN 90
80 NEXT X
85 ND
90 D$ = LEFT$ (C$,X—1)
100 E$ = D$ + "QUALQUER COISA QUE VOCE QUEIRA"
110 PRINT E$
```

Este tipo de programa é a base de programa de processamento de textos — um programa para reduzir gastos com datilografia no escritório.

SE VOCÊ GOSTA DE DESAFIOS

Você achará agora que o Computador pode ficar 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. A palavra INKEY$ faz isto possível. Digite este programa:

```
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, ou "não igual a" "" uma string vazia —— sem nada.*

INKEY$ diz ao Computador observar o teclado para ver se você pressionou alguma tecla. O Computador faz isto em alta velocidade. Enquanto você pressiona alguma tecla, o Computador faz pelo menos 20 verificações. O programa identifica 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 letra da tecla pressionada.

Para verificar, digite isto e rode o programa:

```
60 GOTO 10
```

Não importa quão rápido você é, o Computador é muito mais. Cancele a linha 30 do programa. Para ver qual tecla pressionou.

SINTETIZANDO MÚSICA

Tente usar INKEY$ para fazer um órgão de seu teclado. Consulte 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**

Podemos instruir o Computador para que quando você pressiona uma tecla ele 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. É só digitar qualquer tecla da terceira fila do teclado de A a K.

Como isto mudaria o programa? 120 SOUND T,1? Digite.

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 percebe imediatamente o que você digita.
Há outro modo de escrever este programa usando READ e DATA.
Você sabe como isto deverá ser feito?

Eis aqui a nossa sugestão:

```
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 DATA e READ é fácil acrescentar mais notas ao repertório de seu Computador.

Agora, o Computador vai desafiar sua habilidade. Digite este programa:

```
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 "TARDE DEMAIS"
```

Este programa se resume em escolher dois números aleatórios, 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 digitar o valor certo, ele imprime CORRETO e passa novamente para novo teste. Esta é a função principal do programa.
Quando você demorar para dar a resposta, o Computador imprime a mensagem TARDE DEMAIS e encerra o teste. Para rodar novos testes temos que digitar RUN **ENTER**. Observe que os números que o Computador vai escolher estão entre 5 e 49. Assim a soma sempre terá 2 dígitos. Se durante 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 gera um tom. Quando seu tempo termina, ao mesmo tempo que outro tom é emitido, a tela muda de cor.
E a linha 100? Você já adivinhou qual é a função de VAL? Muito fácil, não? Ele apenas dá o **valor** numérico de uma string para podermos compará-la a outro valor numérico.

CAPÍTULO 7

Errar é humano...

Até agora, você vem gastando muito tempo para apagar e redigitar linhas de programa quando necessitava fazer alguma modificação.
O BASIC oferece um modo bem mais fácil de fazer essas alterações em seus programas. Você vai aprender fazer "edição" de programa. Com ela você poderá fazer toda sorte de modificação em seus programas, sem suar muito...

EDITE ESSA LINHA!

Digamos que você cometeu um erro quando digitava seu programa e a linha 20 ficou assim:

20 ORINT "EU SOU ESE COMPUTADOR"

Você pode apagar esta linha e digitar tudo de novo, ou então editar a linha 20, digite:

EDIT 20 ENTER

e a tela ficará assim:

20 ORINT "EU SOU ESE COMPUTADOR"
20 ■

Você agora está no modo de edição.

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

Primeiro você precisa mudar ORINT para PRÍNT.
- Posicione o cursor sobre a letra incorreta (0)
- Pressione C para corrigir a letra;
- Pressione a tecla com a letra correta (P)

A mudança está feita. A linha então será:

20 P ■

Para verificar a linha toda, pressione L e a linha será apresentada na sua forma nova mas ainda poderá ser alterada.
Para trocar mais que um caracter 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 corrigir;
- Digite os novos caracteres.

> *Observação: Uma vez que você digita* o número *seguido de* C *para trocar, você deve pressionar* n *teclas antes que possa sair deste comando. Ou seja, você deve completar a correção.*

SAIA DESTA!

O primeiro erro já foi sanado, mas há ainda um E na frente do "SEU". Para isso, devemos proceder do mesmo modo que na correção.

- Posicione o cursor sobre o caracter a ser apagado;
- Pressione **D**;

e a operação está completada. Neste caso:

20 PRINT "EU SOU

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

Há outro jeito para apagar um determinado número de caracteres:

- Posicione o cursor sobre o primeiro caracter a ser apagado;
- Digite o *número* especificando quantos caracteres serão apagados;
- Pressione **D**.

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

NÃO ACABOU AINDA!

Agora a linha 20 deve estar assim:

20 PRINT "EU SOU SE COMPUTADOR"

Ainda não está certo. Falta o U no SEU. Portanto, mãos à obra:

- Posicione o cursor sobre o caracter ou o espaço onde a inserção será feita;
- Pressione **I**;
- Digite o(s) caractere(s) que serão inseridos.

Observe que há uma grande diferença, entretanto, entre o modo de inserção e as formas de edição já descritas. Mesmo depois de ter completado a inserção, você ainda permanecerá neste modo. Assim, caso você tente avançar o cursor (pressionando **ESPACO**), você vai estar inserindo espaços. Para sair deste modo pressione **ENTER** ou **SHIFT** **↑**

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).*

CORTE ESSA!

O BASIC tem uma função que abrange as características de apagar e inserir. Esta função é chamada Corte, porque ela permite alterar uma linha cortando o seu fim e inserindo novos caracteres. Para usar esta função, você deve:

- Posicionar o cursor sobre o *primeiro* caracter 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 PRINT "EU SOU SEU"

e que você quisesse mudar tudo depois de SEU. Você precisa posicionar o cursor sobre as aspas (").

20 PRINT "EU SOU SEU ■
Agora pressione **H** e digite a nova mensagem:

20 PRINT "EU SOU SEU COMPUTADOR"

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

ESTIQUE ESTA LINHA!

As vêzes você precisa inserir mais uma instrução na linha 20. Um modo para fazer isto é redigitar a linha toda, mas isso é um trabalho maçante. Um outro modo é utilizar 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 BASIC do seu Computador tem uma solução simples para este problema. Você pode adicionar o que quiser na linha, o que chamam de extender no modo de Edição:

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

Por exemplo, para extender 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 PRINT "EU SOU SEU COMPUTADOR"

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

20 PRINT "EU SOU SEU COMPUTADOR": GOTO 210

PROCURE ESTA LETRA

Existe ainda outro modo de mover o cursor para frente rapidamente é incumbir o Computador de procurar na linha do programa por um determinado tipo de caracter. Se, por exemplo, você está no modo de Edição e decide tirar o R da linha 20, como moveria o cursor sem pressionar ESPACO? A resposta é a seguinte:

- Pressione S; (para procurar)
- Pressione a tecla do caracter que você quer alterar (nesse caso R)

O cursor se encarregará de buscar por toda linha, até encontrar o caracter específico. Você pode então alterar 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 R), mas o que acontece numa linha de programa que tem mais de uma ocorrência? Neste caso, o que acontece se você coloca inadvertidamente um R a mais?

20 PRINT "EU SOUR SEU COMPUTADOR": GOTO 210

Se você simplesmente disser ao Computador para buscar um R ele vai parar no primeiro. Você pode mandar o Computador buscar o segundo R, assim: entrar no modo de Edição da linha 20, pressionar 2 (significa segunda ocorrência) seguido pelo S (para procura). E, então, pressionar ESPACO (o caracter que você quer). Sua tela terá:

20 PRINT "EU SOU ■

MATE O... ERRO

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

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

Suponhamos que você quer eliminar a primeira parte da linha 20 (tudo que vem antes do comando GOTO). Primeiro entre no modo de edição da linha 20, pressione 1 (para a primeira ocorrência) e então K. Depois, pressione G (a primeira letra do texto que restará). Se você então listar a linha, verá o seguinte:

20 GOTO 210

MOVIMENTOS RÁPIDOS

Você já sabe como mover o cursor para frente ou para trás èm 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 você prefere mover o cursor rapidamente mas sem usar nenhum modo de edição. Eis como fazer:

- Digite o número de posições que você quer que o cursor avance;
- Pressione ESPACO

Por exemplo, se você pressionar **5** **ESPACO**, o cursor avançará 5 posições para a direita. Você pode então acionar qualquer função de edição.

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

20 PRINT "EU SOU SEU COMPUTADOR": GOTO 210 ■
O cursor irá parar sobre a letra O do GOTO.

TABELA DE EDICAO

TECLA	FUNÇÃO
EDIT (n.º da linha)	*Edita* a linha especificada
L	*Lista* a linha que está sendo editada.
C novo caracter	*Corrige* um caracter
n **C** novo caracter	*Corrige* os proximos *n* caracteres por novos caracteres
I	*Insere* caracteres
D	*Anula* um caracter
n **D**	*Anula n* caracteres
H	*Corta* o resto da linha a partir do cursor e permite a inserção de texto
X	*Extende* a linha (cursor se move para o fim da linha e o modo de Inserção é ativado)
S caracter	*Busca* pela 1.ª ocorrência do caracter especificado.
n **S** caracter	*Busca* pela enésima ocorrência do caracter especificado.
K	*Elimina* (mata) o resto da linha
n **K** caracter	*Elimina* (mata) até a enésima ocorrência do caracter especificado
n **ESPACO**	*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

ELIMINAR EM MASSA

A função DEL (Delete) é muito útil quando estamos corrigindo um programa extenso. Ela que permite eliminar 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

Mas se você quer apagar apenas a linha 20? Digite:

20 ENTER

Note que neste caso não é necessário usar o comando DEL.

COMANDO	FUNCAO
DEL —	Apaga todo o programa da memória
DEL (n.º da linha)	Apaga a linha especificada
DEL (n.º da linha) —	Apaga a linha especificada até o fim do programa
DEL (n.º da linha) — (n.º da linha)	Apaga do 1.º n.º da linha ao 2.º n.º da linha inclusive
DEL — (n.º da linha)	Apaga todas as linhas do início do programa até a linha especificada, inclusive

FAÇAM SEUS NÚMEROS!

Agora que você já sabe quase tudo, existe uma outra função que funciona em massa é a que permite escolher um trecho do programa na memória e mudar a numeração de suas linhas. Essa função tem várias aplicações tanto para abrir espaço para novas linhas na memória, como também para organizar um programa já verificado.

NORMAS PARA UTILIZACAO DA FUNCAO RENUM		
RENUM (N)	**nova linha**	especifica o novo número da primeira linha a ser renumerada. É opcional; se omitido, é usado 10.
RENUM (N), (I)	**linha inicial**	especifica o n.º da linha no programa original onde você quer iniciar a renumeração. Também é opcional; se omitido o programa inteiro será renumerado.
RENUM (N), (I), (C)	**incremento**	especifica o intervalo a ser usado na numeração das linhas. Como as demais, também é opcional; se omitido, é usado 10.

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

Tente renumerar as linhas do programa conforme as normas:

```
5 CLS
10 PRINT "PRIMEIRA"
20 PRINT "SEGUNDA"
25 PRINT "TERCEIRA"
30 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:
Digite: LIST **ENTER**

```
100 CLS
105 PRINT "PRIMEIRA"
110 PRINT "SEGUNDA"
115 PRINT "TERCEIRA"
120 GOTO 100
```

Observe que o Computador também renumera o número da linha depois da instrução GOTO, na linha 120.
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? Digite:

RENUM 100, 10, 100 ENTER

Agora, liste o programa. Você verá que o Computador nao alterou a linha 5, mas renumerou todas as linhas subseqüentes começando por 100, 110 etc. Se você digitar:

RENUM 1000, 100 ENTER

o Computador renumerará a linha 100 e todas as linhas seguintes. A primeira linha renumerada se tornará linha 1000, 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 da seguinte forma: Digite:

RENUM, ,5 ENTER

Agora o programa inteiro será renumerado, iniciando com um novo número (10) para a primeira linha, e um incremento de 5, nas linhas seguintes.

O comando RENUM não pode ser usado para alterar 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 aparecerá, 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 indicado no seu programa original, RENUM imprime uma mensagem de advertência, do tipo:

UL uuu NA nnn

mas apesar disto renumera o programa, onde: uuu é o número da linha original nnn é o novo número da linha uuu

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

CAPÍTULO 8

Retoques Finais

Neste capítulo, há algumas instruções em BASIC que queremos mostrar. Você deve conhecê-las, pois, na certa, tornarão a sua programação mais fácil.
A primeira palavra é STOP. Fácil? 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 a execução do programa ao chegar à linha 30, onde você colocou um STOP. Neste ponto você pode digitar uma linha de instrução 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 que 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 e corrigir o que está errado. Após a correção você pode retirar as instruções STOP.

PARA PROGRAMADORES AMBICIOSOS

Digite NEW para limpar a memória e então digite:

PRINT MEM ENTER

O Computador imprime quanto espaço está disponível para armazenar programas e dados em BASIC, na memória do Computador. Quando você está digitando um programa muito extenso, vai querer que o Computador execute um PRINT MEM de tempo para tempo, para ter certeza que você está trabalhando dentro de espaço hábil na memória.

Para não gastar memória, você pode omitir espaços em seu programa antes e depois da pontuação, operadores e instruções de BASIC.

AJUDA COM A DATILOGRAFIA

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.
De fato 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 menos 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 que se encontra logo após a instrução GOSUB. Se N é 2, o Computador vai para a sub-rotina pertencente à linha do segundo número; se N é 3, o Computador vai para a subrotina pertencente à linha do terceiro número, após a instrução GOSUB. E se N for igual a 4? Como não há 4 números de linha, o Computador vai simplesmente para a próxima linha do programa.

Aqui está 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 INPUTA 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
```

Preste atenção à 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.

Você pode usar a instrução ON...GOTO de modo similar a ON...GOSUB. A diferença é que neste caso, o Computador pode ser enviado a qualquer linha, e esta, não precisa ser necessariamente uma sub-rotina.
Aqui está parte de um programa usando ON...GOTO:

```
10 CLS
20 PRINT @ 134, "(1) DADOS"
30 PRINT @ 166, "(2) VINTE E UM"
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 DADOS"
1010 END
2000 PRINT @ 236, "JOGO VINTE E UM"
2010 END
3000 PRINT @ 235, "JOGO DE POQUER"
3010 END
```

LÓGICA DE COMPUTADOR

Qualquer um que entende um pouco de inglês sabe a diferença entre AND e OR — até seu Computador. Por exemplo, vamos imaginar que a DYNACOM tenha uma vaga para um programador. Para conseguir o emprego você deve ter:

UM DIPLOMA DE PROGRAMADOR
AND (e)
EXPERIÊNCIA EM PROGRAMACAO

Aqui está a idéia num programa. Apague 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, 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, NAO PODEMOS CONTRATA-LO
```

Digamos agora que a DYNACOM decidiu ser menos exigente. Eis as novas qualificações para o trabalho:

UM DIPLOMA DE PROGRAMADOR
OR (ou)
EXPERIENCIA EM PROGRAMACAO

Foi trocado apenas 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, NAO PODEMOS CONTRATA-LO"
```

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

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

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

MAIS AJUDA COM A MATEMÁTICA

Existem mais algumas palavras que você pode querer usar para seus programas matemáticos.

SGN

SGN indica se um número é **positivo, negativo** ou **0**. 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. Introduza alguns números, tais como:

12, −25, −.015, 0, .33

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 introduza alguns números parecidos com os anteriores.

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"
```

Rode e verifique o que acontece. Mais alguma coisa antes de terminar este capítulo. Digite e rode este programa:

```
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 nímeros maiores que 1E + 38 ou menores que -1E+38. Ele arredonda para zero números próximos de 1E −38 e −1E −38.

As 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)m, por exemplo, pode ser escrito "1E + 09". Isto significa "o número 1 seguido por nove zeros."

*Ou, tecnicamente, $1*0^9$, 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, 5 casas para a esquerda inserindo zeros onde for necessário. Tecnicamente, isto significa $5*10^{-6}$ ou 5 milionésimos (0.000.005), isto é 5/10/10/10/10/10/10

Uma vez acostumado, torna-se muito fácil manipular valores com essa notação.

ASC e CHRS

Cada caracter ou tecla do seu teclado tem um código. Nos seus programas vocêr pode se referir ao caracter ou ao seu código para instruir o Computador a tomar uma ação quando esta tecla é pressionada.

Digite:

PRINT ASC("A") ENTER

e o Computador imprime:

65

65 é o código ASCII pra o caracter A.
Digite:

PRINT CHR$(65) ENTER

e o Computador imprime:

A

para nos dizer: o caracter 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. Caso contrário o Computador não vai nos entender pois ao pressionar esta tecla, durante a digitação do programa, o Computador vai mover um espaço a esquerda e imprimir absolutamente nada.
Digite:

```
10 CLS(0)
20 H = 63
30 SET (H,14,3)
40 A$ = INKEY$
50 IF A$ = CHR$(8) THEN
60 GOTO 40
70 H = —1
80 SET (H,+1,14)
100 GOTO 40
```

Rode o programa e pressione a tecla ←. Agora escreva um programa que faz mover um ponto para a direita utilizando a tecla → (CHR$(9)).

SÓ FALTA FALAR

Quem disse que o Computador não pode falar? Sua voz, entretanto, soará estranha. Vamos conseguir que o Computador fale usando a gravação de sua própria voz. Assim você aumentará o interesse e as animações de seus programas, particularmente os jogos e programas educativos. Mesmo que você não tenha um gravador, poderá ainda usar algumas idéias gráficas que apresentaremos neste capítulo. Desconecte os 3 cabos que ligam seu gravador com o Computador. Coloque uma fita, ponha no começo, pressione PLAY e RECORD, e grave algo. Diga qualquer coisa que você quiser no microfone.

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

Agora digite este programa:

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

Pronto? *Antes* de executar o programa você precisa preparar sua fita para tocar:
- Retorne a fita até o começo da gravação;
- Conecte seu gravador ao Computador.
- Pressione PLAY no seu gravador.
- Aumente o volume de sua TV.

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

Adicione estas linhas:

```
35 CLS
40 A$ = INKEY$
50 PRINT @ 225, "PRESSIONE(X)PARA PARAR 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, o ÁUDIO e o MOTOR do gravador são desligados.
Agora que você já entendeu o princípio de funcionamento, que tal transformar o Computador num professor que "fala"?

CAPÍTULO 9

Atenção, Ação!

Você está pronto para praticar tênis ou jogo espacial ou então tiro ao alvo? Você poderá ensinar seu Computador a jogar qualquer jogo assim que aprender mais uma palavra em BASIC. E esta palavra é POINT. Apague 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) < > THEN GOSUB 100
70 NEXT H, V
80 END
100 PRINT @ 0, "0 PONTO" H ";" V "ESTA' LIGADO"
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.

VIAJANDO PELOS ASTERÓIDES

Neste jogo, nós vamos usar o joystick direito, portanto verifique que esteja conectado. Nós podemos criar asteróides colocando os pontos aleatoriamente na tela, como abaixo. Apague a memória e digite este programa:

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

Para marcar o planeta que seu avião deve alcançar use:

```
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, acrescente estas linhas:

```
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, acrescente 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 geram o bloco. Rode o programa. Experimente girar seu joystick. A linha de cor ciano se moverá para onde você deslocar o joystick. Agora vamos ao jogo em si. 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 gera um tom. Note que a linha 216 faz duas coisas se (IF) o ponto for laranja:

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

Adicione estas linhas ao seu programa:

```
235 IF POINT (H,V) = 3 THEN PRINT @ 0, "PARABENS! 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), 1
1030 NEXT X
1040 PRINT @ 228, "SUA NAVE EXPLODIU"
```

Rode o programa. Você gostaria de ter as instruções do jogo impressas na tela? Adicione 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 NAVE ESPACIAL
2020 DATA ATRAVES DOS ASTEROIDES
2030 DATA PARA O PLANETA AZUL
2040 DATA BATENDO EM MAIS DE DEZ ASTEROIDES
2050 DATA SUA NAVE EXPLODE!
2060 DATA PRESSIONE QUALQUER TECLA QUANDO
2070 ESTIVER NO CANTO SUPERIOR ESQUERDO
```

Rode de novo.

ALTA VELOCIDADE

Neste capítulo vamos mostrar uma alternativa para os gráficos do 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

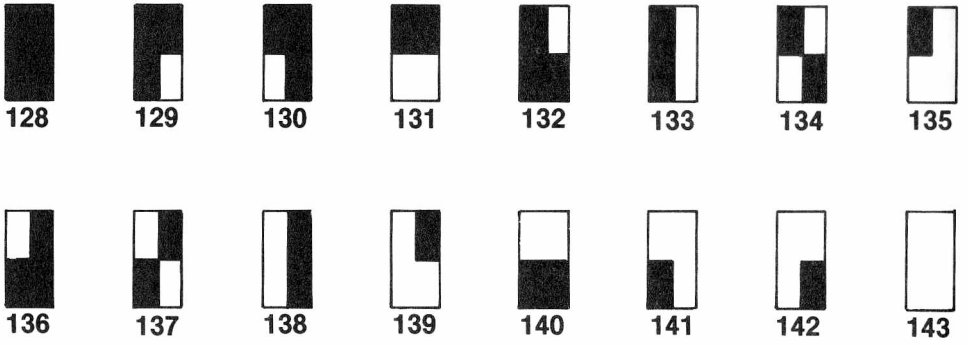
O Computador imprime 3 blocos com diferentes combinações de verde e preto:

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

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

As posições da tela para PRINT @ está no Apêndice (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 caracter que ele representa. Por exemplo, CHR$(65) converte o código 65 no carácter "A". Os códigos 128, 129, 130 e 131 são códigos para caracteres gráficos. Veja no Apêndice E as linhas escuras dividem a grade em blocos. Cada bloco contém 4 blocos menores. Esses blocos podem ser combinados de 16 maneiras diferentes para formar estes caracteres gráficos:



Para imprimir todos estes caracteres gráficos, digite e rode 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 terminar seus caracteres na tela. Caso contrário, ele continua imprimindo o seu costumeiro fundo verde o resto da linha.

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

PRINT CHR$(129+16) ENTER

O Computador imprime o caracter 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 — magenta
112 — laranja

*Note 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, acrescente estas linhas e rode o programa:

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

Como estes caracteres gráficos são similares a A, B, C e D tanto vocêr pode combinar como armazenar do mesmo modo que strings. Limpe a memória e digite:

```
10 A$ = CHR$ (129+16) + CHR$(131+16)
20 B$ = CHR$ (133+112) + CHR$ (143+112) + CHR$ (130+112)
```

E você pode posicioná-los no centro da tela do mesmo modo que posicionaria 2 palavras, usando PRINT @.

Observe a diferença. Você imprime caracteres gráficos usando PRINT @ conforme a tabela "Posições do PRINT @ na tela" enquanto que para "ligar" um ponto na tela você deve usar o comando SET e a tabela "Posições gráficas na tela".

Digite:

```
30 CLS(0)
40 PRINT @ 237, A$;
50 PRINT @ 241, B$;
60 GOTO 60
```

e rode o programa. O programa imprime a imagem de um carro amarelo e um caminhão laranja no centro da sua tela. Vamos melhorar o tráfego. 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
```

O Computador vai gerar aleatoriamente um carro e um caminhão coloridos. Introduza estas 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 vamos aumentar 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 S: NEXT X
140 CLS(0)
150 NEXT P
```

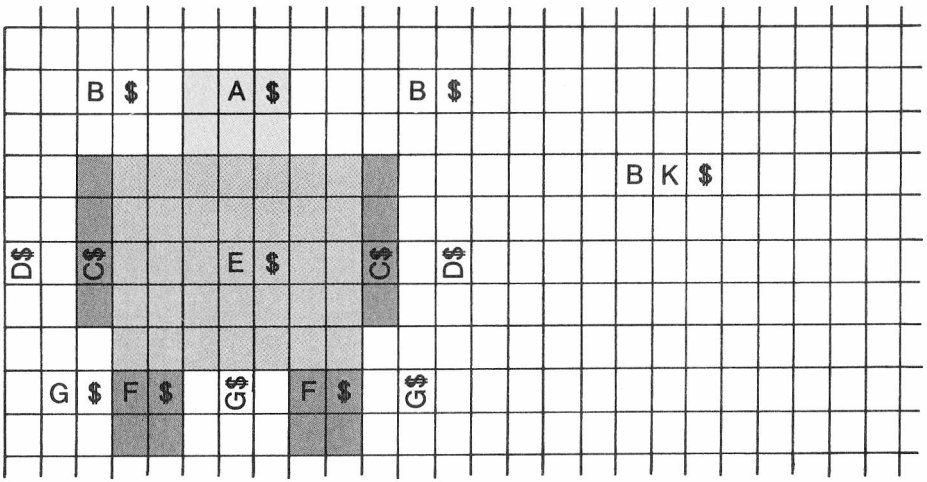
e rode. O tráfego se moverá do canto superior esquerdo ao canto inferior direito da tela. A linha 120 imprime o tráfego na localização P (0 até 480). A linha 130 coloca uma pausa no programa para a velocidade que você digitou. Para fazer o tráfego mover-se através de sua tela continuamente, digite:

```
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
```

Lembra? Mostramos como usar LEFT$ e RIGHT$ no Capítulo 3.

MÚSICA E DANÇA

Neste capítulo terá 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.
Aqui está o nosso "dançarino"



Como isto tomará muito espaço para as strings, digite:

1 CLEAR 1000

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 BK$ = B$ + B$ + B$ + D$ + D$
```

Na tela do Computador o branco será cinza e o tracejado, vermelho.

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

PRINT B$ ENTER
PRINT D$ ENTER
PRINT G$ ENTER
PRINT BK$ ENTER

Para formar a seqüência cinza, digite:

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

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

```
PRINT A$ ENTER
PRINT C$ ENTER
PRINT F$ ENTER
```

B$ tem cinco caracteres de comprimento.

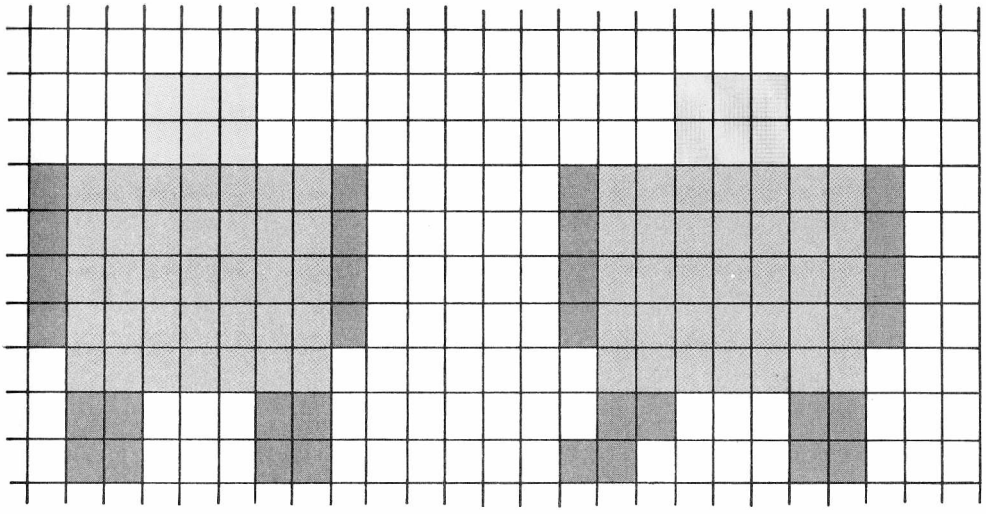
D$ tem dois caracteres; G$ tem 3; BK$ tem 19 e A$ tem 3.

C$ tem um caracter de comprimento, F$ tem 2 e E$ tem 7

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$ + BK$ + B$ + A$ + B$ + BK$
120 FOR X = 1 TO 4
130 BD$ = BD$ + D$ + C$ + E$ + C$ + D$ + BK$
150 L1$ = G$ + E$ + G$ + BK$ + G$ + F$ + G$ + F$ + G$ + BK$ + G$ + F$
+ G$ + F$ + G$
```

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



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

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

Para ver os 3 "passos" do Computador acrescente 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 e localizações. A linha 1010 imprime a cabeça e o corpo 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. Controlando estas posições, você pode facilmente fazer o Computador se mover. Eis aqui como nós fizemos. Mude as linhas 500 e 510 e adicione as seguintes linhas:

```
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 127, 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
```

Rode o programa e assista-o dançar.

Lembre-se de READ e DATA do Capítulo 6.

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 sua imaginação e o que você já aprendeu.

CAPÍTULO 10

Administrador por excelência

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:

- Lista de Compras
- Livros de receitas
- Itens da despensa
- Imposto de Renda
- Recibos médicos
- Lista de endereços
- Lista de telefone
- Agenda de reuniões
- Preço de ações
- Coleções de livros, selos ou discos
- Estoque comercial
- Vendas
- Contas a receber e a pagar
- Cartas
- Jogos
- Etc.

Você já tentou escrever um programa para manipular informações? Se você tentou saiba que o seu Computador foi construído para "rotular e organizar" sistemas, fazendo esta programação muito fácil. Para começar, como conseguir que o Computador lembre a votação para o grêmio da escola para os 14 candidatos.

CANDIDATO	VOTOS
1	283
2	401
3	567
4	221
5	417
6	115
7	303
8	382
9	514
10	102
11	80
12	601
13	616
14	181

Para lembrar o Computador sempre usa variáveis. Para conseguir que o Computador se lembre dos votos para os candidatos, digite:

A = 283 ENTER

B = 401 ENTER

C = 567 ENTER

Veja o Capítulo 2 para revisar variáveis

Podemos melhorar ainda mais. Digite deste modo:

A(1) = 283 ENTER

A(2) = 401 ENTER

A(3) = 567 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 tabela. Cada valor é um item da tabela.

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 A

```
10 DATA 283, 401, 567, 221, 417
20 DATA 115, 303, 382, 514, 102
30 DATA 80, 601, 616, 181
40 READ A, B, C, D, E
50 READ K, L, M, N
60 READ K, L, M, N
70 INPUT "NUMERO DO CANDIDATO(1-14)";Z
75 IF Z > 14 THEN 70
80 IF Z = 1 THEN PRINT A "VOTOS"
90 IF Z = 2 THEN PRINT B "VOTOS"
100 IF Z = 3 THEN PRINT C "VOTOS"
110 IF Z = 4 THEN PRINT D "VOTOS"
120 IF Z = 6 THEN PRINT E "VOTOS"
130 IF Z = 6 THEN PRINT F "VOTOS"
140 IF Z = 7 THEN PRINT G "VOTOS"
150 IF Z = 8 THEN PRINT H "VOTOS"
160 IF Z = 9 THEN PRINT I "VOTOS"
170 IF Z = 10 THEN PRINT J "VOTOS"
180 IF Z = 11 THEN PRINT K "VOTOS"
190 IF Z = 12 THEN PRINT L "VOTOS"
200 IF Z = 13 THEN PRINT M "VOTOS"
210 IF Z = 14 THEN PRINT N "VOTOS"
220 GOTO 70
```

Programa B

```
10 DATA 283, 401, 567, 221, 417
20 DATA 115, 303, 382, 514, 102
30 DATA 80, 601, 616, 181
40 DIM A(14)
50 FOR X = 1 TO 14
60 READ A(X)
70 NEXT X
80 INPUT "NUMERO DO CANDIDATO (1-14)";Z
90 PRINT A(Z) "VOTOS"
100 GOTO 80
```

Na verdade foi fixado (DIM) espaço para 15 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. Digite e rode o segundo programa. Eis como ele funciona:
A linha 40 prepara uma lista de informações, uma matriz (tabela) chamada A, com 14 itens distintos.
As linhas 50 e 70 produzem um ciclo para contar de 1 a 14. A linha 60 põe o seguinte na memória:

Memoria do seu Computador	
A(1) = 283	A(8) = 382
A(2) = 401	A(9) = 514
A(3) = 567	A(10) = 102
A(4) = 221	A(11) = 80
A(5) = 417	A(12) = 601
A(6) = 115	A(13) = 616
A(7) = 303	A(14) = 182

Isto armazena todas os os votos na matriz chamada A. Esta matriz contém 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 os votos armazenados em uma matriz, fica fácil trabalhar com eles.
Vamos supor que a estes votos não são os definitivos e uma segunda votação é possível. Digite estas linhas:

```
92 INPUT "QUER ADICIONAR VOTOS";R$
94 IF R$ = "NAO" THEN 80
96 INPUT "QUANTOS VOTOS A MAIS";X
97 A(Z) = A(Z) + X
98 PRINT "O CANDIDATO" Z "TEM AGORA" A(Z) "VOTOS"
```

Se você quer imprimir uma tabela com todos os votos, adicione estas linhas:

```
72 INPUT "VOCE QUER VER TODOS OS VOTOS?";S$
74 IF S$ = "SIM" THEN GOSUB 110
110 PRINT "CANDIDATO","VOTACAO A", "VOTACAO B"
120 FOR X = 1 TO 14
```

```
130 PRINT X, A(X),
140 NEXT X
150 RETURN
```

e mude a linha 100:

```
100 GOTO 72
```

CANDIDATO	VOTACAO A	VOTACAO B
1	283	282
2	401	402
3	567	566
4	221	220
5	417	419
6	115	110
7	303	308
8	382	382
9	514	511
10	102	105
11	80	80
12	601	602
13	616	616
14	181	182

Nós podemos usar outra matriz no nosso programa para a votação B. Este programa grava os votos da votação A (Matriz A) e da votação B (Matriz B):

```
10 DATA 283, 401, 567, 221, 417
20 DATA 115, 303, 382, 514, 102      } Dados para a Matriz A
30 DATA 80, 601, 616, 181
40 DATA 282, 402, 566, 220, 419
50 DATA 110, 308, 382, 511, 105      } Dados para a Matriz B
60 DATA 80, 602, 616, 182
70 DIM A(14), B(14)
80 FOR X = 1 TO 14
90 READ A(X)  ------------------------> Lê dados da Matriz A
100 NEXT X
110 FOR X = 1 TO 14  ------------------> Lê dados da Matriz B
120 READ B(X)
130 NEXT X
```

```
140 INPUT "CANDIDATO NUMERO";Z
145 IF Z > 14 THEN 140
150 INPUT "PRIMEIRA (1) OU SEGUNDA (2) VOTACAO"; R$
160 IF R$ = "1" THEN PRINT A(Z)
170 IF R$ = "2" THEN PRINT B(Z)
180 GOTO 140
```

TESTE DE MEMÓRIA

Pronto pra outra? Este programa confronta sua memória e a memória do seu Computador. Digite NEW para apagar o programa anterior e digite:

```
5 DIM A(5)
10 PRINT @ 165, "MEMORIZE ESTES NUMEROS"
20 PRINT @ 230 "VOCE TEM 5 SEGUNDOS"
30 FOR X = 1 TO 5
40 A(X) = RND(100)
50 PRINT A (X)
60 NEXT X
70 FOR Y = 1 TO 460*S: NEXT Y : CLS
80 FOR X 1 TO 5
90 PRINT "QUAIS FORAM OS NUMEROS"X
100 INPUT R
110 IF A(X) = R THEN PRINT "CORRETO"
    ELSE PRINT "ERRADO — NUMERO CERTO E'" A(X)
120 NEXT X
```

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

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

A linha 5 reserva espaço para uma matriz chamada: **"A"** com 5 itens.
Da linha 30 à 40 são designados 5 números aleatórios para a matriz. Na linha 50 estes números são impressos na tela.
Da linha 80 à 90 o Computador pergunta a você quais foram os números.
Da linha 110 à 120 ele compara com os números corretos e dá a resposta.
Você pode aumentar a dificuldade deste jogo aumentando a faixa de RND.

DÊ FORÇA ÀS PALAVRAS

Na última parte somente usamos matriz para listas de números. Mas a matriz é também usada para palavras. Não exatamente para uma simples lista de palavras, mas como você verá neste capítulo seu Computador poderá editar e imprimir um texto cheio de palavras.
Iniciaremos com uma lista simples de palavras — uma lista de coimpras:

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

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

S$(1) = "CARNE" ENTER

S$(2) = "FEIJAO" ENTER

S$(3) = "BATATAS" ENTER

Não esqueça do cifrão $ É a única diferença entre estas variáveis indexadas 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$(12)
10 DATA CARNE, FEIJAO, BATATAS, TOMATES
20 DATA OVOS, ALFACE, PEIXE, SAL
30 DATA ACUCAR, PAO, LEITE, ARROZ
40 FOR X = 1 TO 12
50 READ S$(X)  ----------> Lê itens na matriz S$
60 NEXT X
70 PRINT "LISTA DE COMPRAS"
80 FOR X = 1 TO 12
90 PRINT X;S$(X) ----------> Imprime a matriz S$
100 NEXT X
```

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

```
110 INPUT "QUER MUDAR ALGUM ITEM?";R$
115 IF R$ <> "SIM" THEN 200
120 INPUT "DIGITE O NUMERO DO ITEM";N
125 IF N > 12 THEN 130
130 INPUT "QUAL E' O ITEM SUBSTITUTO"; S$ (N)
135 GOTO 80
```

ESCREVENDO UM...

Aqui está um programa que utiliza matriz (tabela) para ajudá-lo nos seus textos. Apague a memória do seu Computador e digite:

```
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

Entendeu como funciona? Aqui está sua explicação:

- A linha 1 prepara uma grande quantidade de espaços para o Computador usar.
- A linha 5 reserva espaço para uma matriz (tabela) 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 manda o Computador de volta à linha 40.
- A linha 60 imprime a tecla que você digitou.
- A linha 70 manda o Computador para a linha que imprime seu texto quando você pressionar a tecla /.
- A linha 80 constrói cada string e o identifica com o número X. X é igual a 1 até você pressionar ponto. Então a linha 80 faz X igual a X + 1.

Por exemplo, se a primeira letra que você pressionar for "D" A$(1) será igual a D.
Se a segunda letra que você pressionar for "y" A$(1) será igual a A$(1) + "y", que é o "D" + "y" ou "Dy". Isto pode continuar até que A$(1) seja igual a primeiro parágrafo. Aí você pressiona "." A próxima letra que pressionar estará junto com A$(2), seu segundo parágrafo. As linhas 140 até 160 imprimem seus parágrafos.

EXERCÍCIO DO CAPÍTULO

Aqui está um programa difícil, mas possível 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.

PARA QUEM TEM IMPRESSORA

Se você tem uma impressora, então deve estar ancioso para colocá-la em uso.
Ligue o cabo da impressora à saída serial (marcada SERIAL I/O) do seu Computador e coloque o papel. O manual que vem com a impressora mostra como fazê-lo. Ligue a impressora. Pronto? Digite este pequeno programa:

```
10 INPUT A$
20 PRINT # —2, A$
```

Agora digite:

LLIST [ENTER]

e observe a impressora trabalhar. Isto com certeza vai tornar fácil qualquer listagem de programas longos. Se seu programa não lista na impressora, verifique se ela está ligada, preparada e conectada ao seu Computador. Tente digitando LLIST novamente.
Agora rode o programa. Digite 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]

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

Digite uma frase com ambas as letras (maiúsculas e minúsculas). Digite:

M[INHA] [IMPRESSORA] [IMPRIME] [MINUSCULAS]

Uma impressora com letras maiúsculas e minúsculas seria ideal 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 da tela? Achou? Você simplesmente mudaria a linha 150 para:

150 PRINT # —2, A$(Y);

CATALOGUE SUA COLEÇÃO

Você sabe que pode armazenar programas em fita. Mas pode também usar a fita para armazenar qualquer lista que você queira. Toda vez que 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 Quinze — Raquel de Queiroz
2. Dona Flor — Jorge Amado
3. 1984 — George Orwell
4. O Chefão — Mario Puzo
5. Luxúria — Judith Crantz

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 "O" indica SAÍDA (Output, em inglês).

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

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

Agora digite:

15 CLS: PRINT "DIGITE SEUS LIVROS — OU <XX> NO FINAL"
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 imrpime-o — não na tela, mas no dispositivo # —1, que é o gravador.
Adicione 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. Acrescente 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:
- Ligar seu gravador ao Computador.
- 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"
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 ligação com o gravador

Está tudo digitado? Então rode-o Note que assim que você pressionar **ENTER**, *o motor do gravador começa a funcionar. 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 QUINZE
TITULO? — DONA FLOR
TITULO? — 1984
TITULO? — O CHEFÃO
TITULO? — LUXURIA
TITULO? — XX
```

O Computador limpará a tela depois de cada título.

Cada vez que você digitar 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 ligará novamente. O Computador está gravando todos os títulos na fita (linha 30) e então encerrando a comunicação com o gravador (linha 50). Agora todos os títulos estão na fita. Para carregá-los de volta para o Computador e digite:

```
60 CLS: PRINT "RETORNE 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 entrada ao Computador. Acrescente estas linhas:

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

A linha 90 introduz o primeiro título — T$. Novamente, T$ não está sendo introduzido 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 em inglês) dos livros, então vai para a linha 120, que encerra a comunicação com o gravador.

Você está imaginando o que o —1 significa? A função EOF gera um —1 se você alcançou o final do arquivo.

Lembre-se em colocar a 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 "RETORNE A FITA E PRESSIONE PLAY"
70 INPUT "PRESSIONE <ENTER> QUANDO PRONTO";R$
80 OPEN "I", #—1, "LIVROS"
85 IF EFO(—1) THEN 120 ————————► Abre a linha para o gravador
90 INPUT #—1, T$
100 PRINT T$ ————————————————► Imprime títulos na fita
110 GOTO 85
120 CLOSE # —1 ————————————————► Fecha linhas para o gravador
```

Agora rode-o. Digite:

```
RUN 60
```

Lembre-se de não apertar **RECORD** *— apenas* **PLAY**. *Além disso, não esqueça de rebobinar a fita.*

Se o seu Computador interromper a comunicação com o gravador, você pode recuperar o controle pressionado **RESET**. *Depois então verifique em quais linhas aparecem erros.*

Quando pressionar **ENTER**, note que o motor do gravador começa a girar. O Computador lê seus itens da fita. Assim que eles entrarem, o Computador vai imprimir os 4 itens na sua tela. Para conferir, seu programa deve aparecer assim:

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

Na linha 30 armazenamos T$ (o título dos livros) na fita. Para fazer isto, 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 introduzimos T$ do gravador. Para fazer isto, temos que abrir comunicação com o gravador para entrada. Depois de terminada a "entrada" fechamos a comunicação.
Pegou? Pense nas respostas destas três perguntas:

PERGUNTAS E RESPOSTAS

P — 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.
P — Seria possível tirarmos as linhas 50 e 80 e rodarmos 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).
P — Funcionaria se você mudasse as linhas 90 e 100 assim:

```
90 INPUT #—1, X$
100 PRINT X$
```

R — Sim, funciona. O Computador mão 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 ARQUIVO ELETRÔNICO

Está ficando interessante? Que tal mudar o programa para que você possa colocar todos estes itens na fita:

TITULO	AUTOR	GENERO
O Quinze	R. de Queiroz	Aventura
Dona Flor	J. Amado	Romance
1984	G. Ornell	Ficção
O Chefão	M. Puzo	Romance
Luxúria	J. Clantz	Drama

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

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

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

```
30 PRINT #—1, T$, A$, S$
```

Agora, para a segunda metade do programa, como você mudaria as linhas de 90 a 100 para que o Computador leia na fita e imprima o título, autor e gênero? Digite:

```
90 INPUT #—1, T$,A$,S$
100 PRINT "TITULO:" T$
102 PRINT "AUTOR:" A$
103 PRINT "GENERO:" S$
```

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

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

BUSCA

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

```
130 CLS
140 INPUT "QUAL GENERO";C$
150 PRINT "RETORNE 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
RETORNE A FITA - PRESSIONE PLAY
PRESSIONE <ENTER> QUANDO PRONTO
LIVROS DE FICCAO:
1984          G. ORWELL
```

CONTROLANDO SUA CONTA CORRENTE

Agora é sua vez de tentar.
Digamos que você tenha emitido a seguinte relação de cheques:

Nº	DATA	PAGO A	CONTA	QUANTIA
101	13/05	PADARIA	ALIMENTO	Cr$ 32.000
102	13/05	POSTO	CARRO	Cr$ 180.000
103	14/05	SUPERMERCADO	ALIMENTO	Cr$ 420.000
104	17/05	PASSAGENS	FÉRIAS	Cr$ 380.000
105	19/05	HOTEL ILHA	FÉRIAS	Cr$ 530.000

Vamos escrever um programa para 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. E o Computador lhe dirá quanto gastou neste item. Lembra ainda como gravar informações na fita?

NORMAS PARA ARMAZENAR DADOS EM FITA

Para armazenar dados na fita você deve:
1. (OPEN) Abrir comunicação com-a fita para saída;
2. (PRINT) armazenar dados na fita;
3. Continuar imprimindo dados na fita até terminar, e, então,
4. (CLOSE) Fechar comunicação com a fita;

NORMAS PARA LER (OBTER) DADOS DA FITA

Para ler dados da fita você deve:
1. (OPEN) Abrir comunicação com a fita para entrada;
2. Usar EOF(—1) para ver se já alcançou o fim do arquivo na fita;
3. (INPUT) Obter dados da fita;
4. Continuar entrando dados até que você tenha terminado ou alcançado o fim do arquivo e, então,
5. (CLOSE) 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
```

ARQUIVAR: TÃO FÁCIL QUANTO 1, 2, 3

Qualquer arquivista lhe dirá que encontrar pastas arquivadas em ordem alfabética é muito fácil, mas organizá-las assim é maçante. Portanto, vamos fazer com que o Computador faça isso por você. Digite:

```
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. Digite várias palavras até você se convencer de que o Computador já sabe organizar em ordem alfabética. Com strings, os sinais maior que, menor que e igual, que discutimos no capítulo 5, têm agora um novo significado. Eles definem 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 organizar em ordem alfabética, você pode escrever um programa para ordenar uma longa lista de palavras. Aqui está o nosso:

```
10 DIM A$(5)
20 FOR I = 1 TO 5
30 IMPUT "DIGITE UMA PALAVRA"; A$(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.*

Antes de explicar como ele funciona, mostraremos o que está acontecendo quando o programa está sendo executado.
Digite:

```
30 READ A$(1)
200 DATA MIGUEL, TAVARES, DANIEL, ALEX, SUZANA
```

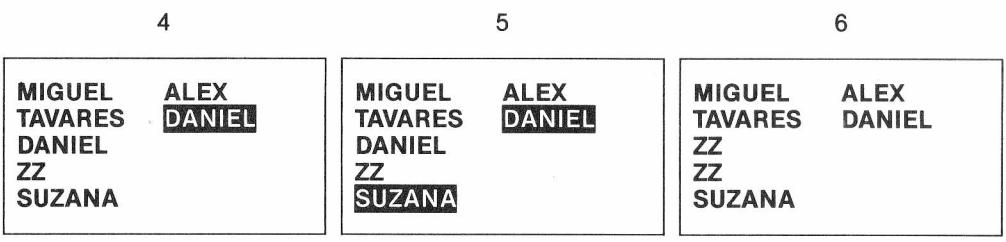
Cancele a linha 120 e digite:

```
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 a aparência da tela — assim você pode ver melhor o que está acontecendo. Rode o programa.
Está muito rápido? Digite esta linha para reduzir a velocidade, assim você pode ver o que está acontecendo.

```
107 FOR T = 1 TO 600 : NEXT T
```

Rode o programa novamente. Veja com atenção a segunda coluna. Veja como o primeiro nome muda de MIGUEL para DANIEL e para ALEX. Depois note o que acontece com ALEX na primeira coluna. Ele se torna ZZ.
Aqui estão os quadros de como o Computador determina a primeira e a segunda posição:



Quando o programa começa, MIGUEL é comparado com MIGUEL para ver qual antecede o outro alfabeticamente. MIGUEL permanece no alto e é comparado com TAVARES. MIGUEL ainda permenece no alto. Em seguida, MIGUEL é comparado com DANIEL. Como este antecede MIGUEL, ele agora assume a posição de MIGUEL no alto da 2.ª coluna. Agora DANIEL é comparado com ALEX que vem para cima e é comparado com SUZANA. ALEX permanece no alto.
Agora que todos os nomes foram comparados com a primeira posição, o Computador repete o ciclo para determinar as demais posições. ALEX 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$(1) — MIGUEL — com cada um dos outros nomes na matriz A$, até alcançar a palavra que antecede A$(1).
Em nosso exemplo, a terceira palavra — DANIEL — o antecede. A linha 100, então, faz A$(X) igual a A$(3) — o lugar de DANIEL na matriz. Quando DANIEL é comparado com a primeira palavra — ALEX — A$(X) se torna A$(4).
A linha 120 imprime A$(4) — ALEX —, a linha 130 faz A$(4) igual a ZZ.
Neste ponto, as figuras 50 e 60 fazem X igual a 1 novamente. A$(1) — MIGUEL — será novamente comparado com outros nomes na matriz.
Quando a posição de MIGUEL 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) — TAVARES — 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 complexos com classificações mais rápidas. Se você tem um número maior de itens para classificar, pode querer desenvolver um outro método de classificação.

FICANDO ANALÍTICO

Rotulando a sua informação sob vários itens, você estará apto a visualizar os diferentes modos. Como exemplo, use o programa das votações do candidato do início deste capítulo.

CANDIDATO	PRIMEIRA VOTACAO	SEGUNDA VOTACAO
1	143	678
2	215	514
3	125	430

Para simplificar, usamos três candidatos apenas.
Dando a cada item dois índices, nós podemos colocar tudo em uma única tabela (matriz).

CANDIDATO	MATRIZ D PRIMEIRA VOTACAO	SEGUNDA VOTACAO
1	D(1,1) 143	D(1,2) 678
2	D(2,1) 215	D(2,2) 514
3	D(3,1) 125	D(3,2) 430

O primeiro índice indica o candidato o segundo, a primeira ou a segunda votação relacionada a ele. Por exemplo, usamos D(1,2) para indicar os votos da segunda votação.

Vamos colocar todos em um programa. Digite :

```
5 DIM D(3,2)
10 DATA 143, 678, 215, 514, 125, 430
20 FOR A = 1 TO 3
30 FOR B = 1 TO 2
40 READ D(A,B)
50 NEXT B
60 NEXT A
70 INPUT "CANDIDATO nº. <1—3>"; A
80 IF A < 1 OR A > 3 THEN 70
90 INPUT "1A. <1> OU 2A. <2> VOTAÇÃO"; B
100 IF B < 0 OR B > 2 THEN 90
110 PRINT D(A,B)
120 GOTO 70
```

A linha 5 reserva um espaço na memória para um matriz chamada D. Cada item pode ter 2 índices. O primeiro índice não pode ser maior que 3, nem o segundo maior que 2.
Rode o programa. Note que os votos foram indexados. Tente diferentes combinações de número de candidatos, votos etc.
As linhas 20 até 60 lêem os votos para 1ª e 2ª votação dos candidatos 1, 2 e 3, e, antes de colocá-los na memória do Computador, que deve estar assim:

MEMORIA DO SEU COMPUTADOR	
D(1,1) = 143	D(1,2) = 678
D(2,1) = 215	D(2,2) = 514
D(3,1) = 125	D(3,2) = 430

Agora que você tem dois aspectos para identificar os votos, seu programa pode ficar mais interessante.
Elimine as linhas 70-120 e digite:

```
70 INPUT "DIGITE <1> CANDIDATO OU <2> VOTACAO"; R
80 IF R<1 OR R> 2 THEN 70
100 ON R GOSUB 1000, 2000
110 GOTO 70
1000 INPUT "CANDIDATO N.(1-3)"; D
1010 IF D <1 OR D> 3 THEN 1000
1015 CLS
1020 PRINT @ 132, "1A. VOTACAO" A "VOTOS"
1030 PRINT
1040 FOR A = 1 TO 2
```

```
1050 PRINT "VOTACAO" A
1060 PRINT D(A,B)
1070 NEXT B
1080 RETURN
2000 INPUT "VOTACAO <1> OU VOTACAO <2> "; D
2010 IF D <1 OR D> 2 THEN 2000
2015 CLS
2020 PRINT @ 132, "VOTOS DE VOTACAO" D
2030 PRINT
2040 FOR A = 1 TO 3
2050 PRINT "VOTACAO" A
2060 PRINT D(A,B)
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 — 1.ª e 2.ª votação. No capítulo anterior nós programamos matrizes unidimensionais — seus itens tinham apenas um índice. Mas não há razão pela qual seu Computador não possa permitir a programação em 3 ou 4 dimensões.

A TERCEIRA DIMENSÃO

Suponhamos que queremos pesar agora os votos em função dos votos obtidos em 3 classes diferentes, para os membros candidatos — a matriz de 3 dimensões. As estatísticas dos votos mostrados acima eram todos da mesma classe. Aqui está a informação:

CANDIDATO	1ª CLASSE (1) 1ª VOTACAO	2ª VOTACAO
1	143	678
2	215	514
3	125	430

	2ª CIASSE (2) 1.ª VOTAÇAO	
1	525	54
2	318	157
3	254	200

	3.ª CLASSE (3) 1.ª VOTAÇAO	
1	400	119
2	124	300
3	75	419

Eis aqui como nós indicamos todos estes votos na Matriz D

MATRIZ D 1ª CLASSE (1)		
CANDIDATO	1ª VOTACAO	2ª VOTACAO
1	D(1,2) 143	D(1,1,2) 678
2	D(1,2,1) 215	D(1,2,2) 514
3	D(1,3,1) 125	D(1,3,2) 430

2.ª CLASSE (2)		
1.ª VOTAÇAO		
1	D(2,1,1) 525	D(2,1,2) 54
2	D(2,2,1) 318	D(2,2,2) 157
3	D(2,3,1) 254	D(2,3,9) 200

3.ª CLASSE (3)		
1.ª VOTAÇAO		
1	D(3,1,1) 400	D(3,1,2) 119
2	D(3,2,1) 124	D(3,2,2) 300
3	D(3,3,1) 75	D(3,3,2) 419

Para ter tudo isso na memória de seu Computador, apague seu programa e digite:

```
5 DIM P(3,3,2)
10 DATA 143, 678, 215, 514, 125, 430
20 DATA 525, 54, 318, 157, 254, 200
30 DATA 400, 119, 124, 300, 75, 419
40 FOR A = 1 TO 3
50 FOR B = 1 TO 3
60 FOR C = 1 TO 2
70 READ D(A,B,C)
80 NEXT C
90 NEXT B
100 NEXT A
110 INPUT "CLASSE <1>, CLASSE <2>, CLASSE <3>"; A
120 IF A <1 OR A> 3 THEN 110
130 INPUT "CANDIDATO (1-3)"; B
140 IF B <1 OR B> 3 THEN 130
150 INPUT "VOTACAO" <1>, VOTACAO <2>"; C
160 IF C <1 OR C> 2 THEN 150
170 PRINT D(A,B,C)
180 GOTO 110
```

Rode o programa e teste todos esses índices. As linhas 40 a 100 colocam isto na memória do seu Computador:

MEMORIA DE SEU COMPUTADOR	
D(1,1,1) = 143	D(1,1,2) = 678
D(1,2,1) = 215	D(1,2,2) = 514
D(1,3,1) = 125	D(1,3,2) = 430
D(2,1,1) = 525	D(2,1,2) = 54
D(2,2,1) = 318	D(2,2,2) = 157
D(2,3,1) = 254	D(2,3,2) = 200
D(3,1,1) = 400	D(3,1,2) = 119
D(3,2,1) = 124	D(3,2,2) = 300
D(3,3,1) = 75	D(3,3,2) = 419

Para aproveitar de todas as três dimensões, elimine as linhas 110 a 180 e digite:

```
110 PRINT : PRINT "DIGITE <1> PARA CLASSE"
120 PRINT "<2> P/ CANDIDATO OU <3> P/ VOTACAO"
130 P = 224 : INPUT R
140 ON R GOSUB 1000, 2000, 3000
150 GOTO 110
1000 INPUT "CLASSE <1>, CLASSE <2>, CLASSE <3>"; A
1010 IF A <1 OR A> 3 THEN 1000
1020 CLS
1030 PRINT @ 102, "VOTOS DA CLASSE" A
1040 PRINT @ 168, "1A. VOTACAO"
1050 PRINT @ 176, "2A. VOTACAO"
1060 FOR B = 1 TO 3
1070 PRINT @ P "CANDIDATO" B
1080 FOR C = 1 TO 2
1100 PRINT @ P + 8*C, D(A,B,C)
1110 NEXT C
1120 P = P + 32
1130 NEXT B
1140 RETURN
2000 INPUT "CANDIDATO(1-3)"; B
2010 IF B <1 OR B> 3 THEN 2000
2020 CLS
2030 PRINT @ 102, "VOTOS DO CANDIDATO" B
2040 PRINT @ 168, "1A. VOTACAO"
2050 PRINT @ 176, "2A. VOTACAO"
2060 FOR A = 1 TO 3
2070 PRINT @ P, "CLASSE" A
2080 FOR C = 1 TO 2
2100 PRINT @ P + 8*C, D(A,B,C)
2110 NEXT C
2120 P = P + 32
2130 NEXT A
2140 RETURN
3000 INPUT "VOTACAO <1>, VOTACAO <2>"; C
3010 IF C <1 OR C> 2 THEN 3000
3020 CLS
```

```
3030 PRINT @ 102, "VOTOS DA VOTACAO" C
3040 PRINT @ 168, "CANDIDATO1"
3050 PRINT @ 176, "CANDIDATO2"
3060 PRINT @ 184, "CANDIDATO3"
3070 FOR A = 1 TO 3
3080 PRINT @ P, "CLASSE" A
3090 FOR B = 1 TO 3
3100 PRINT @ P + 8*B, D (A,B,C)
3110 NEXT B
3120 P = P + 32
3130 NEXT A
3140 RETURN
```

Tente escrecer um programa para distribuir as cartas do baralho usando uma matriz de 2 dimensões. Uma dimensão pode ser um dos 4 naipes; a outra, os valores das cartas (1-13).

CAPÍTULO 11

Som e Imagem

Uma das características mais excitantes do MX BASIC é sua capacidade de criar desenhos mais precisos, mais variados e mais fáceis de usar: São chamados de Gráficos de Alta Resolução. Vamos começar com o elemento gráfico mais básico — um simples ponto — e construir a partir daí.
Com o MX BASIC fica realmente simples colocar um ponto sobre a tela. Digite o seguinte programa e observe o que acontece:

```
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 ficará 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 colocar um ponto de uma determinada cor em qualquer lugar da tela. PSET tem o seguinte formato:

PSET (h,v,c)

h especifica uma posição sobre o eixo X (horizontal) com valor numérico entre 0 e 255.
v especifica uma posição sobre o eixo Y (vertical) com valor numérico entre 0 e 191.
c especifica a cor do ponto com valor numérico entre 0 e 8. Na ausência de C, é assumida a cor atualmente presente.
Observação: *A coordenada X deve vir sempre antes da coordenada Y.*

Chamamos este quadro de "Quadro de sintaxe". Outros quadros aparecerão muito nesta seção. Em geral, eles explicam parâmetros tais como PSET. Preste muita atenção na sua organização e pontuação.

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 um 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 × 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 acha que consegue colocar um segundo ponto sobre a tela da mesma cor que o primeiro, porém no centro da tela ? Acrescente uma outra linha de programa para colocar esse ponto. Digite e rode:

35 PSET (128,96,8)

Que tal agora um ponto no centro direito inferior da tela. Substitua a linha 35 (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
```

Até aqui, sabemos duas coisas sobre o *c*:
1. Ele especifica a cor do ponto.
2. Ele pode ser qualquer número de 0 a 8.
Isto é verdade com certa reserva...
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.
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.
Para o modo gráfico atual você pode utilizar as cores correspondente aos códigos 5, 6, 7 e 8.

ACENDE, APAGA

Você faz alguma idea como apagar este ponto. 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 MX BASIC usando PRESET, mas muito mais fácil.
Você não precisa especificar a cor só a coordenada (o ponto).

PRESET (h,v)
h - especifica a posição sobre o eixo X (horizontal) com valor numérico entre 0 e 255. v - especifica a posição sobre o eixo Y (vertical) com valor numérico entre 0 e 191.

Observe que não é necessário especificar a cor com PRESET, pois o Computador usa automaticamente a cor do fundo.

LINHA E PONTO

Então, você já sabe como colocar um ponto em qualquer lugar da tela e até vários pontos. Muito bem. E o que você pode fazer com isso?
Para responder esta pergunta, vamos mostrar como seu Computador pode traçar uma linha

horizontal ou vertical fina, e ainda uma diagonal. E, mais ainda, você pode variar a largura da linha. Um jeito de fazer isso é usar a instrução LINE do MX BASIC. Para ver a LINE em operação, use o programa dos pontos, porém um pouco modificado (e, para facilitar, no futuro nós o chamaremos "Linhas"). Mude primeiro a linha 30 para:

30 LINE (0,0)—(255,191), PSET

Agora cancele a linha 35:

35 ENTER

O seu programa deve ficar assim:

```
5 PMODE 1,1
10 PCLS
20 SCREEN 1,1
30 LINE (0,0)—(255,191), PSET
40 GOTO 40
```

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.
Vamos mudar a *direção* da linha, digamos, do canto inferior esquerdo para o superior direito? Difícil? 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.

QUE TAL CRUZAR AS 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, voce pode traçar tantas linhas quanto quiser — desde que você tenha aprendido o formato. Como uma linha reta é definida entre dois pontos, a primeira coisa 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 indicados pelas coordenadas (h,v). 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 da seguinte forma:

LINE (h1,V1)-(h2,v2),a,b

(h1,v1) especifica o ponto inicial da linha com valor numérico entre 0 e 255; entre 0 e 191.
(h2,v2) especifica o ponto final da linha com valor numérico entre 0 e 255; entre 0 e 191.
a é PSET ou PRESET.
b B ou BF. Isso é opcional.

Falaremos sobre os parâmetro ***a*** *e* ***b*** mais tarde.

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 indicar esse PSET.

Experimente usar o quadro com as posições gráficas para traçar (localizar) os pontos usados 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 alnda não tiver usado a instrução LINE no seu programa, o Computador começa a linha no centro da tela (128,96). Por exemplo:

```
20 LINE (0,0)-255,191), PSET
30 LINE-(191,0), PSTE
```

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.
Vamos analisar o programa que gerou as linhas concorrentes:

```
25 LINE (0,0)-(255,191), PSET
30 LINE (0,191)-(255,0), PSET
```

Até aqui 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 PST e PRESET, experimente mudar PSET na linha 25 para PRESET e rode o programa:

```
25 LINE (00)-(255,191), PRESET
```

O que aconteceu? A tela de sua TV deve mostrar uma única linha alaranjada (criada na linha 30) que corre do conta 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?

OBSERVAÇÃO: PSET e PRESET não definem a cor da linha, apenas:
- **PSET** *traça a linha com a cor predeterminada para primeiro plano.*
- **PRESET** *utiliza a cor predeterminada como de fundo para tracar a linha. Isso equivale, na prática, a "apagar" a linha.*

QUADRADOS OU RETÂNGULOS

Existem ainda alguns aspectos relativos à instrução LINE. O B, por exemplo, vem do inglês "box", que quer dizer caixa. O MX 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 usar o programa "linhas" de novo:

```
 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 4O
```

Apague a linha 30 e adicione o sufixo "B" à linha 25. Assim:

```
30 ENTER
25 LINE (0,0)-(255,191),PSET,B
```

Agora rode o programa. Que tal, "encaixou-se"?

COMO ENCHER 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 característica 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 obteve uma grande caixa alaranjada (256 × 192) sobre um fundo cinza.

FUNDO COLORIDO

Como já faz algum tempo que estamos mencionando cores de fundo e de primeiro plano, já é tempo de explicar (pelo menos em parte) o modo como se controla essas cores.
Dentro de certos limites, a função gráfica COLOR permite 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 correpondendo a um código de cor.
Fundo é um número de 0 a 8 e correspondendo a um código de cor.

Até agora, apenas utilizamos a cor cinza (5) e laranja (8), mencionadas 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 4 opções disponíveis). Insere a linha 6 no seu programa e apague a linha 25:

```
6 COLOR 5,7
```

Rode-o e veja o que acontece.
Agora inverta a ordem das cores digite e rode:

```
6 COLOR 7,5
```

No próximo capítulo vamos ver como conseguir mais cores da função COLOR.

CAPÍTULO 12

Coisas de Cinema

Quando você envia qualquer tipo de informação para a TV, ela é armazenada numa parte da memória chamada "RAM de vídeo". O gerador de vídeo do Computador lê essa "parte da memória" para saber o que deve aparecer na tela. A área "normal" de memória é suficiente grande para texto (letras e números), mas não para gráficos (desenhos). Conseqüentemente, o Computador precisa de uma área (espaço) maior para uso com gráficos.
Esta parte da RAM de vídeo pode conter até 8 "páginas" de memória, e, dependendo de suas necessidaes, você pode reservar de uma a oito páginas com PCLEAR. (Uma "página" contém 1 536 posições de memória.) PCLEAR determina o tamanho da memória reservado para gráficos. Quando você não usar PCLEAR, o Computador reservará 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.

CORES "A LA MODE"

Altere 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
```

Observe a linha 5. Ela contém uma das mais importantes funções gráficas do seu Computador:

```
5 PMODE 1,1
```

PMODE permite escolher o modo gráfico com sua resolução específica (quantidade de detalhes) e selecionar em que "página" da memória seu gráfico vai começar.

PMODE-MODO, PAGINA DE INICIO

Modo é um valor 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).

Observe 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 MX 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 O, respectivamente) é o tamanho do ponto: a alta resolução usa a menor área para formar cada ponto da 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.

CARACTERISTICAS DO PMODE					
PMODE	PONTOS NA TELA	MODO DE COR	PAGINAS NECESSARIA	TAMANHO DO PONTO	PAGINA INICIAL
4	256 × 192	2 cores	4	□	1 a 5
3	128 × 192	4 cores	4	□□	1 a 5
2	128 × 192	2 cores	2		1 a7
1	128 × 96	4 cores	2	▦	1 a 7
0	128 × 96	2 cores	1		1 a 8

Pela tabela, 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".

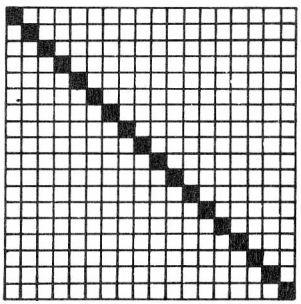
Em seguida, vêm os modos 3 e 2. Como os pontos desses modos são duas vezes o tamanho de modo 4, a resolução não é tão alta. Nós os chamamos "média resolução".

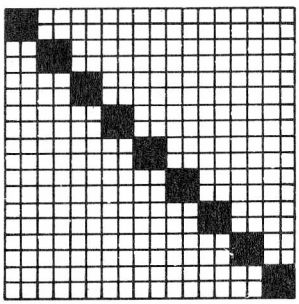
Os modos 1 a 0 têm pontos 4 vezes o tamanho do modo 4 e duas vezes os modos 3 e 2. Como esta é a mais baixa resolução dos modos gráficos, nós os chamamos "baixa resolução".

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.

As ilustrações a seguir mostram a diferença entre os modos 4 e 0 em relação ao tamanho do ponto e resolução.

ALTA RESOLUÇÃO





BAIXA RESOLUÇÃO

Tenha em mente de que a tela de gráfico está sempre cheia de "pontos". A questão simplesmente é saber o tamanho, a cor e quantos deles estão lá.

Vamos voltar ao programa "Linhas" e estudar a linha 5:

5 PMODE 1,1

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.

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 no modo de alta resolução.
2. Também deve ter havido uma mudança de cor, pois você mudou do modo de 4 cores para o de 2 cores.

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 são possíveis para cada modo que você escolher.

TABELA 2
CONJUNTO DE CORES

PMODE	CONJUNTO DE CORES (VER SCREEN)	COMBINACAO DE DUAS CORES	COMBINACAO DE QUATRO CORES
4	0	preto/verde	—
	1	preto/cinca	—
3	0	—	verde/amarelo/azul/ vermelho
	1	—	cinza/claro magenta/laranja
3	0	verde/preto	—
1	0	—	verde/amarelo/azul/ vermelho
	1	—	cinza/claro magenta/laranja
0	0	preto/verde	—
	1	preto/cinza	—

Observe que 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 ou 1, com o conjunto de cores 0.
Abaixo você têm um programa que traça uma caixa 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 MODE = 0 TO 4
10 PMODE MODE, 1
20 PCLS
30 SCREEN 1,1
40 LINE (75,50)—(125,100), PSET, B
50 FOR Y = 0 TO 500: NEXT Y
60 NEXT MODE
70 GOTO 5
```

VIRANDO A PÁGINA CERTA

Como já vimos o MX BASIC divide a memória 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 anteriormente.
Rode este programa, você terá uma idéia melhor a que nos referimo-nos:

```
 5 PMODE 3,1
10 PCLS
20 SCREEN 1,1
30 LINE (110,20)—(120,30),PSET,B
40 GOTO 40
```

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 memoria para gráficos. Quando você inicia na página 2, isso significa que foram requeridas as páginas 2, 3, 4 e 5. Porém você reservou apenas as páginas 1 a 4.
Assim a página 5 não está disponível!
Mais tarde mostraremos como contornar esse problema de modo que você possa usar o PMODE 3,2.
O seu programa "Linhas" ainda deve estar 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. Ou seja, 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 c
A cor é um número de 0 a 8 correspondendo a 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.

10 PCLS 6

Na tela devem aparecer linhas alaranjadas sobre um fundo ciano (código de cor 6).

ACRESCENTANDO PÁGINAS

Ao reservar páginas de memória, precisará eventualmente de PCLEAR para garantir um determinado número de páginas para seus gráficos, entre uma e oito páginas.
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 número de 1 a 8 e especifica o número de páginas de gráfico reservadas. Observação: *No MX BASIC, PCLEAR 4 é feito automaticamente. Você precisa do PCLEAR apenas quando você quer reservar um número diferente de páginas.*

Provelmente você está imaginando porque não deixar simplesmente o Computador usar PCLEAR 8 o tempo todo. A razão é PCLEAR 8 reduz a quantidade de memória disponível para o uso de seu programa em BASIC. As vezes 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 dimensione sua necessidade de memória.
Agora já entendemos a função de PCLEAR. Mas, 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 animação, do mesmo modo utilizado na confecção de desenhos animados — quadro a quadro.

ALTOS E BAIXOS

Você talvez pode pensar que o seu Computador é um pouco louco, mas agora mostraremos o que é um ioiô verdadeiro. Na verdade você pode chamar este programa de ioiô. Introduza-o e rode-o.

Se você não dimensionar as exigências da memória do programa com as exigências da memória de vídeo, você obterá um ?OM ERRO (Insuficiência de Memória, ou em inglês Out of Memory).

```
10 PCLEAR 8
20 FOR A=1 TO 8
30 PMODE 0,P
40 PCLS
50 LINE (128,0)—(138,10+(P—1)*15),PSET
60 CIRCLE (128,P*15),15
70 NEXT P
80 FOR P=1 TO 8 : GOSUB 110 : NEXT P
90 FOR P=7 TO 8 STEP — 2 : GOSUB 110 : NEXT P
100 GOTO 80
110 PMODEN 0,P
120 SCREEN 1,0
130 FOR T=1 TO 10 : NEXT T
140 RETURN
```

Como resultado, nós criamos animação aos gráficos do 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.

COPIAR PÁGINAS

Com PCOPY, você pode copiar os conteúdos de uma página da memória de gráficos numa outra página.

PCOPY ORIGEN TO destino
Origem e destino são valores, entre 1 e 8 especificando as páginas de memória. Observação: *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 deverá digitar:

```
PCOPY 3 TO 8
```

Uma das vantagens do PCOPY é que ele pode encurtar seus programas, eliminando a repetição das instruções de criação dos desenhos.

O ÚLTIMO PONTO

A última função P que você pode precisar, a PPOINT, lê a cor de um determinado ponto do gráfico. O seu programa pode assim utilizar esta informação para tomar alguma providência.

PPOINT (h,v)
h - especifica a coordenada horizonta (X) do ponto com um número de 0 a 255. Y - especifica a coordenada (Y) do ponto com um número de 0 a 191.

O programa abaixo mostra como o Computador testa a cor de 4 pontos sucessivos.

```
 5 PMODE 3,1
10 PCLS
20 SCREEN 1,1
30 PSET (10,11,8)
35 PSET (10,12,7)
40 PSET (10,13,6)
45 PSET (10,14,5)
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 apresentar:

```
8
7
6
5
```

o que corresponde ao código de cor dos pontos especificadas.
Agora, vamos verificar o comando SCREEN da linha 20, do programa "Linhas"

20 SCREEN 1,1

A primeira coisa que diz a instrução ao Computador é se você quer usar a tela da TV para gráficos (linhas, círculos etc.) ou texto (letras, números).
A segunda é que SCREEN diz ao Computador qual conjunto de cores você quer usar.

SCREEN MODO DE TELA, MODO DE COR
Modo de tela: 0 para a tela de texto ou 1 para a tela de gráficos. **Modo de cores** é 0 ou 1, conforme tabela. **Nota:** *Se modo de tela ou cor for um número positivo maior do que 1, seu Computador interprete e trata esse número como se fosse 1.*

o primeiro parâmetro de SCREEN diz ao Computador qual tipo de tela ele deve formar (gráfico ou texto). No programa "Linhas", experimente mudar a linha 20 para:

20 SCREEN 0,0

Em seguida, rode o programa. O seu Computador parece em dificuldades? Pressione 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:

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 DE COR	MODO DE DUAS CORES	MODO DE QUATRO CORES
0	preto/verde	verde/amarelo/azul/vermelho
1	preto/cinza	cinza/ciano/magenta/laranja

Não se iluda acreditando que SCREN 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 o 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).

CAPÍTULO 13

Você é o artista

Tudo isso que foi dito sobre SCREEN, PMODE e PCLEAR o está deixando confuso? Neste 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 (h,v), r,c,hw,início,fim

h - especifica a coordenada horizontal (X) do centro do círculo, com um número de 0 a 255.
v - especifica a coordenada vertical (Y) do centro do círculo, com um número de 0 a 191.
r - especifica o raio do círculo em pontos (posição da tela) da tela, como uma unidade de medida.
c - especifica um código de cor disponível com um número de 0 a 8. Isto é opcional; se omitido, é usada a cor de primeiro plano.
hw - especifica a razão altura/largura, com um número de 0 a 255. Isto é opcional; se omitido, é usado 1.
início - especifica o ponto de partida do círculo com um número entre 0 e 1. Isto é opcional; se omitido, é usado 0.
fim - especifica o ponto final do círculo com um número entre 0 e 1. Isto é opcional; se omitido, é usado 1.

Como, normalmente, 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), do centro do raio.
Neste caso, portanto, 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 CLS
20 SCREEN 1,1
25 LINE (0,0)—(255,191),PSET
30 LINE (0,191)—(255,0),PSET
40 GOTO 40
```

Elimine a linha 25 e mude a linha 30 do programa para:

```
30 CIRCLE (120,96),95
```

Rode o programa. A sua TV pode mostrar um círculo alaranjado sobre um fundo cinza, meio torto. Porém você esperava que o MX BASIC faria um círculo perfeito, 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 PMODE 4 (alta resolução):

```
5 PMODE 4,1
10 CLS
20 SCREEN 1,1
30 CIRCLE (128,96),95
40 GOTO 40
```

Agora, rode-o. Isso é um círculo!

CÍRCULOS DE CORES

Após decidir o raio do círculo, pode escolher sua cor. Todavia, 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 CLS
20 SCREEN 1,1
30 CIRCLE (128,96),95
40 GOTO 40
```

Primeiro, vamos fazer o círculo num tamanho mais razoável:

```
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 qualquer cor disponível.

CRIANDO ELIPSES

Você já teve bambolê ou pneu e tentou amassá-los para formar uma elipse? Você pode fazer o mesmo a um círculo se o Computador usar a relação largura/altura (hw).
A largura da elipse será sempre igual ao raio do círculo, 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, esse programa traça um círculo:

```
5 PMODE 4,1
10 PCLS
20 SCREEN 1,1
30 CIRCLE (128,96),30,,1
40 GOTO 40
```

Porém, se você mudar *hw* para:

```
30 CIRCLE (128,96),30,,3
```

o círculo ficará mais alto do que largo. Mas, se mudar *hw* para:

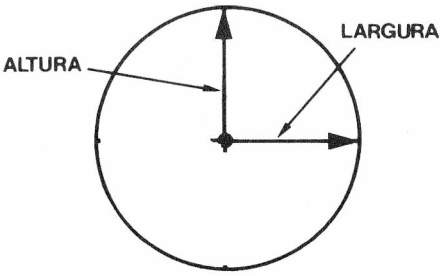
```
30 CIRCLE (128,96),30,,.25
```

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. A medida que *hw* for maior que 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

É, depois:

30 CIRCLE (128,96),30,,100



Observe que omitimos o código de cor. Isto diz ao Computador para usar a cor de primeiro plano. Mas temos de incluir a vírgula, entretanto.

DO FIM AO COMEÇO

As últimas opções disponíveis com CIRCLE lhe permitem traçar apenas uma parte de um círculo (um arco). Para utilizar esta opção, especifique o ponto onde deve começar o arco (qualquer fração entre 0 a 1), coloque uma vírgula e indique onde é o fim (qualquer fração entre 0 e 1). O ponto inicial (0) para qualquer círculo é equivalente às 3 horas num relógio de ponteiros. 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.

DÊ UMAS PINCELADAS

A função gráfica PAINT permite pintar qualquer figura com qualquer cor disponível. A regra para PAINT é:

PAINT (h,v),c,b

h - especifica uma coordenada horizontal (X) com um número de 0 a 255.
v - especifica uma coordenada vertical (Y) com um número de 0 a 191.
c - especifica o código de cor com um valor numérico de 0 a 8.
b - especifica a cor de borda na qual a pintura deve parar com um número de 0 a 8.

Os números dentro dos parênteses especificam o ponto 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, procure fazer 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 calcula 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!
Elimine 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, magenta e laranja.
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, metade do círculo será amarelo (2) chegando até a borda vermelha (4).

Lembre-se sempre de que você só pode usar cores que estejam disponíveis no modo gráfico e conjunto de cor selecionadas por PMODE e SCREEN. Quando você especificar 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.

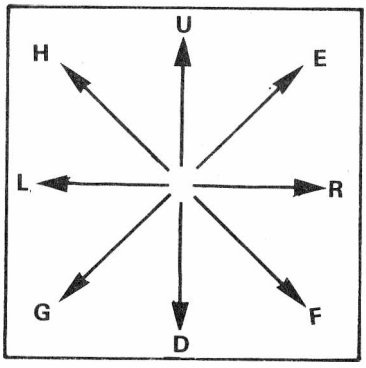
TRACE UM DESENHO

Até aqui aprendeu a criar linhas, círculo e caixas. Nós ainda mostramos como 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 mostramos a você. Este atalho é chamado DRAW, e lhe permite traçar 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 da 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 *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". Cancele as linhas 25 e 30 do programa e introduza 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 legível. Todavia, você deve separar sempre as coordenadas (h,v) 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 (128,96) neste caso.

M h,v

h - é a coordenada horizontal (X) com um valor numérico de 0 a 255.
v - é a coordenada vertical (Y) com um valor numérico de 0 a 191.

Nota: *M deve ser sempre precedido pela letra B, ou aparecerão linhas não desejadas.*

No programa acima instruímos o Computador para começar a traçar o 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.*

GIRANDO O DESENHO

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 vertical, 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 (h,v) ímpar, as linhas F e H terão um ligeiro degrau no ponto médio; se ambas as coordenadas (h,v) tiverem números pares, as linhas E e G terão o degrau. Isso é normal.

M ABSOLUTO VS. RELATIVO

Imagine que 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 (h,v).
Há uma outra forma do comando M que lhe permite especificar o movimento "relativo" ao invés do "absoluto". Até aqui, desenhamos com movimento absoluto, significando que temos os pontos especificados em termos de suas coordenadas (h,v) reais. Com movimento relativo, podemos especificar pontos em relação ao ponto *atual* (último desenho). A regra do movimento relativo é:

M sinal desloc. h desloc. v

desloc. h é um número especificando a distância para se deslocar horizontalmente da posição *h* atual. Se *desloc. h* é precedido por um sinal mais " + ", a posição *h* é incrementada pela quantidade especificada. Se *desloc. h* é precedido por sinal de menos "—", a posição *h* 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. v é um número especificando a distância para se deslocar verticalmente 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. v).*

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 (h = 128 + 15 = 143, v = 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 h = 128 + 15 = 143, v = 96 — 15 = 81.

TROCANDO A ESCALA

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 omitir o 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 revisar 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 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, 100U20R20D20L20"
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).

AINDA CORES

A função C de DRAW lhe permite determinar 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
```

Volte para a escala normal (mude S2 para S4 ou retire S2), e dentro do primeiro conjunto de ponto e vírgula na linha 30, coloque:

```
C6
```

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
```

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 um valor numérico de 0 a 8 correspondendo a 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"

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 torna-se ciano.

Se você quiser "apagar" uma linha, desenhe uma outra sobre a mesma usando a cor de fundo.

QUAL É SUA INCLINAÇÃO

Uma outra opção disponível com a instrução DRAW 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 um valor numérico 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 vertical (comprimento de 25 pontos) reta. Agora mude a linha para:

30 DRAW "A1;BM128,96;U25"

Rode o programa. A linha deve ser agora horizontal.

LINHAS INVISÍVEIS

Digamos que você queira desenhar com DRAW uma linha invisível. Você pode fazer isto com a opção B, que especifica que 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"

Rode o programa. Veja o que acontece. Lembre-se, apenas, que a linha imediatamente seguinte à B estará invisível.

O QUÊ?! AINDA OPÇÕES?

Ainda uma outra facilidade 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, analisá-la. Mude a linha 30 do programa para:

30 DRAW "BM128,96; 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 traçar 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 traçou uma linha de 25 pontos para a direita e voltou a sua posição original. Em seguida, ele traçou 25 pontos para baixo e novamente voltou à posição original. Finalmente, o Computador traçou uma linha de 25 pontos para a esquerda e, uma vez mais, voltou à posição original.

STRINGS VARIÁVEIS OU STRINGS CONSTANTES

Lembra-se quando dissemos que a string após a instrução DRAW pode ser uma constante ou uma variável. Se você deseja usar uma variável string, apenas substitua-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 criar uma caixa laranja (25 × 25), cujo canto inferior esquerdo está no centro da tela. O MX BASIC oferece uma variação disto, entretanto. Nós a intitulamos "o ato de execução (X)". Enquanto você está executando uma rotina DRAW, o ato de execução lhe permite executar uma outra string DRAW, então retornar e completar a primeira operação. Altere a linha 30 do programa para:

30 DRAW "BM95,50;U25;R25;XA$;D25;L25"

Rode o programa. O Computador deve começar uma linha iniciando em (95,50) para cima, 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$, o Computador retorna à string original e completa a sua execução (D25,L25). Observação: *A posição do traço não volta à sua posição anterior (por exemplo, o primeiro quadrado). Ela permanece onde estava após completar A$.*

TRANSFERINDO TABELAS

Você aprendeu a mover e transferir figuras de uma página de memória para outra e acha isto muito flexível. Entretanto, você pensará diferente quando ver o que Computador faz com GET

e PUT. Estas funções permitem que seu Computador "leia" na tela 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 MX-1600 COLOR. As regras para GET e PUT são:

O quê? As matrizes (ou tabela) foram explicados no Capítulo 10; assim, se você está um pouco esquecido refresque sua memória antes de ir adiante.

GET h1,v1 - h2,v2, matriz, G

h1, v1 — ponto inicial é a coordenada (x1,y1) do canto superior esquerdo de uma área retangular na tela.
h2, v2 — ponto final é a coordenada (x2,y2) do canto inferior direito da mesma área retangular na tela.
Matriz de destino — é o nome de uma área de memória que armazenará o conteúdo da área retangular.
G diz ao Computador para armazenar os conteúdos do retângulo com detalhe gráfico completo. Isso é opcional para alguns usos, como no modo 3 ou 4, com cores.

PUT h1, v1 - h2, v2, matriz, acao

h1v1 e h2v2 - ponto inicial e final são os mesmos explicados acima.
matriz de origem é o nome de uma matriz ou área de memória que contém os dados a serem transferidos para a área retangular da tela.
ação determina como os dados são apresentados no retângulo:
PSET ativa cada ponto que está na matriz de origem
PRESET desativa cada ponto que está na matriz de origem
AND, OR, NOT (Veja a seguir)
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 como funciona GET/PUT. Mais tarde, vamos nos aprofundar sobre o funcionamento destas 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
```

O programa traça um círculo laranja no canto superior da tela (posição (10,10) — linha 30 do programa). Após um ou dois segundos, ele desaparece e reaparece em outro lugar da tela (posição (110,110) — linha 45 do programa).
O Computador age assim:
- Cria uma matriz (na linha 25) do tamanho de 20 × 20 pontos
- Cria um círculo (na linha 30)
- Obtém o círculo que estava dentro do retângulo original e o armazena na matriz (linha 35).
- Limpa a tela (linha 40).
- Coloca o círculo da matriz no retângulo de destino especificado na instrução PUT (linha 45).

(h1, v1) (h2, v2) GET → MATRIZ → PUT (h1, v1) (h2, v2)

Observação: cada ponto da matriz consome 5 bytes da sua memória RAM.

COMO DIMENSIONAR A MATRIZ

Observe que antes de usarmos GET e PUT, nós temos de criar um arranjo bidimensional para armazenar (linha 25). Como então 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 × 20 requer uma matriz 40 × 20. Como o MX BASIC utiliza um elemento zero, a seguinte instrução de dimensão cria um arranjo adequado:

```
DIM V(39,19)
```

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 2 **não** use as opções.
- No PMODE 4 ou 3, use as opções.

As opções PUT realizam as funções como mostrado na tabela no fim do capítulo.

OUTRAS OPÇÕES

Observe como o próximo programa usa as instruções GET e PUT para simular em movimento. Para acelerar a ação, você pode diminuir o incremento STEP. Preste atenção particular ao modo como o programa usa as funções que você já conhece (LINE, CIRCLE, PAINT etc.) para criar 15 retângulos na tela:

```
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 —10
60 PUT (I,81—I/5)—(I + 60,111—I/5),V,PSET
65 NEXT I
70 GOTO 70
```

A única coisa que fizemos foi mudar (atualizar) a posição do retângulo pelo valor de I. Assim PSET coloca o mesmo retângulo da matriz em posição diferente na tela. Esta tabela é o resumo das funções e suas ações.

FUNCAO	ACAO
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 um deles estiver ativado, o ponto permanece ativado.
NOT	Um operador lógico. Converte o estado de cada ponto no retângulo de destino independente dos conteúdos do ar ranjo PUT, ou seja, desativa os pontos ativados e vice-versa.

CAPÍTULO 14

Toque outra vez, MX

Se você acha que o Computador é um bom artista, então ainda não viu nada! Espere até ouvi-lo tocar música! Pronto, então vamos explorar sua habilidade com a função **PLAY.**
Antes de mais nada, devemos observar que PLAY não é uma função gráfica. Portanto, você não terá de começar seus programas com PMODE, PCLS ou SCREEN. Vamos então realaxar pondo isso tudo de lado e ouvir um pouco de música.

OUÇA COM ATENÇÃO

Eis a regra para a função PLAY:

PLAY musica

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 segunda oitava (2).*
comprimento de nota — ***L** seguido de um número de 1 a 255. Se omitido, é mantido o último comprimento.*
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 "N" 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.
Digite o seguinte para ver o que isto significa:

PLAY "A" ENTER

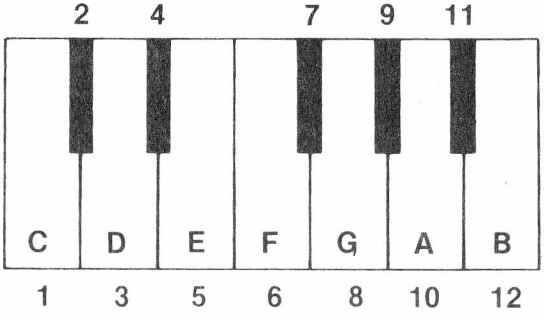
Para ouvir a mudança que um bemol ou sustenido pode proporcionar, digite:

PLAY "A;A#" ENTER
PLAY "A—;A;A#;A;A—" ENTER

Você pode fazer o mesmo com todas as 7 notas (A-G) da escala. (**Nota:** há apenas meio-tom separando B e C. Assim B# = C,B = C—. Neste caso o Computador somente reconhece C e B).

NOTAS OU 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 ser 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.

Numero	Nota	Numero	Nota
1	C	9	G#/A —
2	C#/D —	10	A
3	D	11	A#/B—
4	E—/D#	12	B
5	E/F—		
6	F/E#		
7	F#/G—		
8	G		

UMA PORÇÃO DE NOTAS

Como as notas no programa "Escala" não tem duração especificada, elas duram 1/4 do tempo padrão. 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. De fato, você **pode** usar qualquer número de 1 a 255. (Você já ouviu falar sobre 1/25 de nota?)
Vamos variar as durações das notas para tentar perceber, na prática, como isso funciona. Digite esta linha:

PLAY "L2;A;L4;A;L2;A" **ENTER**

TABELA DE COMPRIMENTO DE NOTAS		
Numero L	**Comprimento de nota**	**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.
Digite este:

PLAY "L1;A;A#;A—" **ENTER**

Observe 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".

OITAVA A MAIS

Como conhecedor de música, você sabe que há outras durações além da 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;AE;C;L4;A"

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** da instrução PLAY nos permite selecionar outras oitavas. Para alterar a oitava use o **O** seguido de um número entre 1 e 5 (qualquer número, além deste, resultará num erro). Seu Computador usa automaticamente a segunda oitava se você não especificar qual oitava você quer. Vamos tentar tocar essa oitava.

PLAY "CDEFGABAGFEDCBA" ENTER

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;ABGFEDCBA" ENTER

assim passamos para a terceira (O3) oitava.

TOQUE MAIS ALTO!

Claro, você pode ajustar o volume de sua música apenas controlando o volume da TV. Mas quem 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:

```
5 CLS
10 PLAY "V5;C;V10;C;V20;C;V30;C;"
20 GOTO 10
```

Use BREAK para interromper o ciclo, ou você acaba ganhando uma dor de cabeça.

> **Nota:** *O Computador usa o valor atual de V até que você o altere.*

UM MOMENTO DE SILÊNCIO

Pode ser que o último programa seja mais agradável de se ouvir se as notas não forem tocadas

todas juntas. Vamos usar a função pausa (P) para alguns momentos de silêncio entre as notas e ver se soa 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;A;P2;V10;A;P2;V20;A;P2;V30;A;P2"

Realmente uma pausa de meia nota (P2) entre todos esses A não soa melhor, 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. Entretanto, os espaços não são necessários.

ENTRANDO NO RITMO

Seu programa pode ser 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 **T**2. Inicie reduzindo o ritmo de seu programa:

10 PLAY "T1;V5;A;P2;V10;A;P2;V20;A;P2;V30;A;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 habilidade 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 tocar (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 cifrão ($) sempre que você utiliza a função de execução. Nesse exemplo, você pode retirar todos os pontos e vírgulas, exceto aqueles seguintes ao cifrão ($).*

UMA NOTA A MAIS

Não, nós não inventaremos uma nova nota, como H ou J, para você. 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. Vamos ilustrar 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 levantar vôo. Deu para ouvir a mudança da marcha?

Verifique o ritmo com o menos (—) executando este programa:

```
5 CLS
10 PLAY "T255"
20 PLAY "T —;A;A#;A—"
30 GOTO 20
```

Depois de uma rápida partida o Computador gradativamente começa a diminuir o ritmo. Não é que a multiplicação é mais rápida que a adição? Na linha 10, ajuste o ritmo para 2, troque o T na linha 20 para T > e rode-o:

SUFIXO	PROPOSITO
+	Adiciona 1 ao valor atual.
–	Subtrai 1 do valor atual.
>	Multiplica o valor atual por 2.
<	Divide o valor atual por 2.

```
10 PLAY "T2"
20 PLAY "T>; A;A#;A—"
```

Nós começamos 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 STRAUSS

Depois de tanto trabalho envolvendo retas, círculos, pontos, oitavas, bemóis, semitons e outras coisas, você deve estar "meio" esgotado. Pois, então, num último esforço, introduza o programa abaixo, relaxe e ouça 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#;"
25 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;"
35 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;"
45 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;03;
   G#;O4;A;L2;F#;L8;D;O3;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$;XH$;P1P1P1"
230 GOTO 20
```

Pressione **BREAK** quando quiser interromper.

CAPÍTULO 15

Matemática Avançada

Além das funções matemáticas descritas em capítulos anteriores você tem agora a sua disposição várias funções avançadas. Neste capítulo terá um breve resumo de cada função e como usá-la para desenvolver qualquer operação Matemática complexa. Antes disso, entretanto, há algumas funções e definições que você deve conhecer.

POTENCIAÇÃO

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:

NUMERO ↑ POTENCIA
Número especifica o número que você quer elevar à potência. Ele pode ter qualquer valor numérico. ↑ é gerado pressionando ⇧ (Lateral esquerda do teclado). **potência** é o expoente ao qual o número será elevado. Ele pode ter qualquer valor numérico.

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 10.ª potência. Sua tela apresenta:

1.00000001E+10

Já que 10 000 000 000 tem mais que 9 dígitos significativos, o Computador utiliza a notação **E**. E, isso já foi explicado anteriormente. Que tal 100 à 100.ª potência? Você obteve um? OV ERRO (overflow). Isto significa que a resposta é muito grande para o Computadore manipular. Seu Computador foi projetado para manipular qualquer número entre -10^{38} a $+ -10^{38}$. Qualquer número fora dessa faixa resultará num erro de saturação ou (estouro).

RAIZ QUADRADA

A função SQR permite encontrar a raiz quadrada de um número.
Aqui está a regra:

SQR (numero)
número é um valor numérico nunca menor que 0.

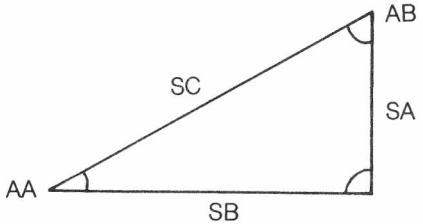
Por exemplo, se você quer a raíz quadrada de 100, digite:

PRINT SQR (100) ENTER

e você encontrará a resposta 10. (Se você ainda não sabe).

FUNÇÕES TRIGONOMÉTRICAS

Observe bem este triângulo, pois vai usá-lo durante todo o tempo que discutirmos as funções trigonométricas; assim tente se familiarizar muito bem com ele.
A trigonometria tem muitas 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 extensão da viga ou a inclinação do telhado. Assim, a matemática permite solucionar muitas coisas e este capítulo pode proporcionar 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 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, podemos definir as funções de trigonométricas 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 Vs. RADIANOS

Para definir um ângulo, você pode usar 2 unidades diferentes de medida. A unidade mais comum é grau, a outra, "mais técnica", é o radiano. **Seu Computador assume que todos os ângulos são expressos em radianos**. Podemos converter os ângulos de radianos para graus (e vice-versa) como mostrado no quadro abaixo:

Graus para radianos: graus/57.29577951
*Radianos para graus: radianos*57.29577951.*

No programa exemplo deste capítulo, incluímos um "conversor" de ângulos. Este permite entrar com graus, e o Computador automaticamente converte para radianos. (linha 60)

O SENO (SIN)

SIN (angulo)
ângulo é um valor numérico para um ângulo em radianos.

Um uso para SIN é determinar os 2 lados desconhecidosa de um triângulo, se 2 ângulos e um lado são conhecidos.
Introduza e rode o programa seguinte. Você pode introduzir 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/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 TEM 100
 90 PRINT"LADO A(SA) E" SA: PRINT "LADO B(SB) E" SB: GOTO 10
100 PRINT "DESCULPE; NAO É UM TRIANGULO, TENTE NOVAMENTE"
110 GOTO 10
```

Quando o Computador solicitar os ângulo AB e AC, você deverá digitar 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 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 realizado.

ONDA SENOIDAL

Você provavelmente já viu uma onda senoidal. Elas são usadas em tensão AC e outras medidas elétricas. Rode o seguinte programa para ver uma onda senoidal:

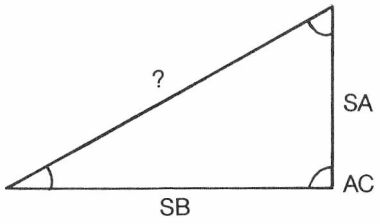
```
 5 CLS
10 FORA=180 TO -179 STEP-10
15 RD=A/57.29577951
20 CL=SIN(RD)*14+16.5
30 PRINT TAB(CL);"S"
40 NEXT A
45 GOTO 45
```

O CO-SENO (COS)

A função co-seno é relacionada com a função seno e tem a seguinte sintaxe:

COS(angulo)
ângulo valor numérico para um ângulo em radianos

Uma aplicação para o co-seno é determinar o comprimento de um lado de um triângulo dados os outros dois lados e um ângulo. Para a identificação do lado e do ângulo, veja o triângulo abaixo;



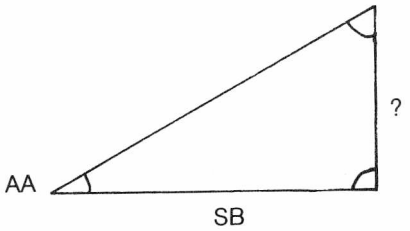

```
 5 CLS
10 INPUT "QUAL O ANGULO C(AC)"; AC:IF <=0 OR
   AC>=180 THEN 100
20 AC=AC/57.29577951 'CONVERTE GRAUS EM RADIANOS
30 INPUT "QUAL O LADO A(SA)"; SA:IF SA< 0 THEN 100
40 INPUT "QUAL O LADO B(SB)"; SB:IF SB<=0 THEN 100
50 SC=((SA↑2))+ (SB↑ -(2*(SA*SB*COS(AC))):IF SC<0 THEN 100
60 PRINT "LADO C (SC)E"" SQR(SC): GOTO 10
100 PRINT "DESCULPE, NAO E' UM TRIANGULO, TENTE NOVAMENTE
110  GOTO 10
```

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.

A TANGENTE (TAN)

Outra função trigonométrica que você pode usar no seu Computador é TAN. Ela permite calcular a tangente de um ângulo.

TAN(angulo)
ângulo é uma expressão numérica para um ângulo em radianos.



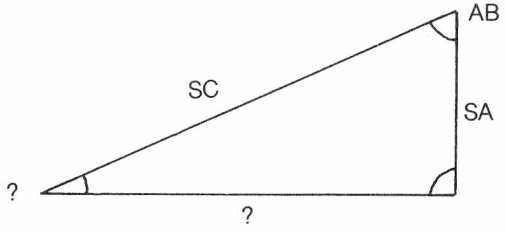
Entre outras coisas, a função tangente pode ser usada para determinar o comprimento desconhecido de um lado de um triângulo uma vez dados o comprimento do outro lado e um ângulo.

```
  5 CLS
 10 INPUT "QUAL O LADO B(SB)"; SB:IF SB<0 THEN 100
 20 INPUT "QUAL O ANGULO A(AA)" AA:IF <=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)"; A:IF AB<=0 OR
    AB>=180 THEN 150
 50 X=(180-AB) '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.

LOGARITMOS (LOG)

LOG fornece o logaritmo natural de um número. É o inverso do EXP, assim:

X=LOG(EXP(X))

LOG(numero)
número é um valor numérico 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. Se você omitir a base, o Computador assume que a base é $e = 2.718281828$.

Para encontrar o logaritmo de um número em outra base B, use a seguinte fórmula:

$$\log(X)_{\text{base } B} = \frac{\log_e(X)}{\log_e(B)}$$

Por exemplo:

$$\log_2 32768 = \frac{\log 32768}{\log 2}$$

Digite:
PRINT LOG (1) ENTER
PRINT LOG (100) ENTER
PRINT LOG (2.718281828) ENTER

EXPONENCIAÇÃO (EXP)

A função EXP é a exponononencial natural de um número, isto é, $e^{\text{número}}$. Esta função é o inverso do LOG; por essa razão, X=EXP(LOG(X)).

EXP (numero)
número é um valor numérico menor que 87.3365.

Rode este programa para ver como EXP funciona:

```
10 CLS
20 INPUT "INTRODUZA X";X
30 PRINT "EXP(X)="; EXP(X)
40 GOTO 20
```

FIXANDO UM PONTO

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 ver 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 eliminando todos os que estão à direita do ponto decimal.

FIX (numero)
número é um valor numérico

Por exemplo:

```
PRINT FIX (2.34690)
2
OK
```

Digite e rode o seguinte programa. Ele separa a parte inteira da decimal de um número qualquer dado por você.

```
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
```

UMA FUNÇÃO

O MX BASIC tem uma função numérica, **DEF FN**, diferente de todas as outras que já mencionamos. **DEF FN** permite criar sua própria função matemática e que a use da mesma maneira que as outras (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 nome que você designou para essa função.

DEF FN nome (lista de variaveis) = formula
nome é o nome designado para a função que você criou. **lista de variáveis** é a variável que representa cada variável a ser usadas 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, nunca no modo imediato.*

Quando você usar esta função não se esqueça de usar a instrução DEF FN antes de executar a função que ela define.
Caso contrário, um ?UF ERRO (função indefinida) ocorrerá.

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 "embutida" 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. Acrescente estas linhas no programa:

```
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 de conversão de graus para radianos.

CAPÍTULO 16

String de Novo

STRING$

A função STRING$ é usada para criar gráficos, tabelas, ou qualquer similar, pois cria uma seqüência de caracteres iguais.

Eis a regra de STRING$:

STRING$ (comprimento, caracter)
comprimento é um valor numérico entre 0 e 255 **caracter** é uma expressão string numérica que fornece um caracter para o código ASCII correspondente. Se uma string é usada constantemente, ela deve ser postas entre aspas.(" ").

O número de caracteres exibido depende do número que você especificou no comprimento. Os caracteres usados dependem do caracter ou código ACSII especificado. Veja no Apêndice uma lista completa dos códigos dos caracteres ACSII.

```
 5 CLS
10 X$=STRING$ (13,"*")
15 PRINT @ 10, X$; "AGENDA"; X$
20 Y$=STRING$ (4,"*")
25 Z$=STRING$ (9,42)
30 PRINT @32, "NOME";Y$;"ENDEREÇO";Z$;
   "TEL";Y$
60 GOTO 60
```

Na linha 10 designamos à X$ o valor STRING$(13,"*"), isto é, uma string de 13 asteriscos.
A linha 15 diz ao Computador para imprimir (inicialmente a impressão na localização 10 da tela) X$, a palavra AGENDA seguida por X$ novamente. (Veja o quadro com as posições do PRINT@ na tela, nos Apêndices deste Manual.)
Como X$ é igual a 13 asteriscos (*), esses caracteres são impressos antes e depois da palavra AGENDA.
Na linha 20 foi designado à Y$ a string de 4 asteriscos. Já na linha 25, nós não especificamos um asterisco, ao invés disso, foi indicado o código ASCII 42. Se você recorrer ao Apêndice de, códigos ACSII dos caracteres, verá que ele representa justamente o asterisco.
A linha 30 imprime o título da sua tabela. A partir daí você pode criar qualquer gráfico ou tabela.

ALGUÉM VIU MINHA STRING?

Se você quiser procurar uma string dentro de uma outra, use INSTR.
Eis a sua regra:

INSTR (posiçao, busca, alvo)
posição especifica a posição na *busca* (string de busca), onde a busca vai começar, e é uma expressão numérica entre 0 e 255. É opcional; se omitidos, a busca começa automaticamente pelo primeiro caracter de *busca*. **busca** é a string onde será procurada e string alvo. **alvo** é a string que você está procurando. **Nota:** *busca* e *alvo são expressões de string.*

INSTR retorna um 0 se:
- *posição* é maior que o comprimento de *busca*;
- *busca* é nula;
- *alvo* não pode ser encontrado.

Observe como INSTR funciona:

```
 5 CLEAR 500
10 CLS
15 INPUT "TEXTO DE BUSCA";S$
20 INPUT "TEXTO DE ALVO";T$
25 C=0; P=1 'P=POSIÇÃO
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($), CHR$(159))+RIGHT$(S$),
LEN(S$—F—LEN(T$)+1)
50 P=F+LEN(T$)
55 IF P<=LEN(S$)-LEN(R$)+1 THEN 30
60 PRINT "ENCONTRADA"; C; "OCORRENCIA(S)"
```

Vamos rodar um exemplo:

```
TEXTO DE BUSCA? O SEU COMPUTADOR E' UM COMPANHEIRO
TEXTO DO ALVO? OM
O SEU C■ ■PUTADOR É UM COMPANHEIRO
O SEU COMPUTADOR É UM C■ ■MPANHEIRO
ENCONTRADA 2 OCORRENCIA(S)
OK
```

Vamos ver o que aconteceu:
- Você atribuiu a S$(busca) o valor: O SEU COMPUTADOR E' UM COMPANHEIRO (linha 15).
- Você atribuiu a T$(alvo) o valor: OM (linha 20).
- Você disse ao Computador para iniciar na 1ª 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 CEP 020 - conseqüentemente, *020 será o alvo da string(A$).

```
5 CLS
10 A$="*020
20 X$="PAULO SANTOS, TUCUMA 123 SP*02019: MARIA DE JESUS, PRATES 341
SP*04030: ROBERTO FLORES, SANTANA SP*03020"
```

O Computador pode buscar X$, acrescentando:

```
50 PRINT ISNTR(X$,A$)
```

Agora rode o programa.
Sua tela exibirá:

23
OK
Isto significa que o código postal procurado se encontra na posição 23 de X$

SUBSTITUINDO UMA STRING

Nós já falamos sobre MID$ em capítulo 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 anterior, posiçao, comprimento) = substituto
string anterior é o nome da string que você quer alterar. **posição** é um valor numérico especificando a posição do primeiro caracter a ser trocado. **comprimento** é um valor numérico especificando o número de caracteres que serão substituídos. É opcional. **substituto** é uma expressão de string que substitui uma parte específica de uma *string* anterior. **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.*

Para ver mais claro esta função rode este programa:

```
 5 CLS
10 A$="TIJUCA,SP"
20 MID$(A$,14)="RJ"
30 PRINT A$
```

Na linha 10 foi atribuído a A$ o valor TIJUCA, SP. Depois, na linha 20, você pediu ao Computador para substituir a string anterior (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 a função MID$:

```
 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 menos "—" no lugar.

CAPÍTULO 17

Entradas e Saídas

Instruções de entrada/saída permitem enviar dados do teclado ao Computador, do Computador à TV, do Computador à impressora. Essas funções são usadas basicamente em programas para introduzir dados, emitir resultados e mensagens.

LINHA DE ENTRADA

A primeira instrução de *entrada/saída* é LINE INPUT. Eis sua regra:

LINE INPUT lembrete; variavel string
lembrete é a mensagem lembrete. É opcional; se usada deve vir entre aspas. **variável string** é o nome atribuído à linha que será digitada e 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 pelo Computador como parte da string de entrada sem atribuí-los nenhum significado especial.
- 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 "SOBRE 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 esta incluso um sinal de interrogação e em seguida um espaço.

Rode o seguinte programa para ter uma idéia melhor do LINE INPUT.

```
10 CLEAR 300: CLS
20 PRINT TAB(8); "INSTRUÇÃO LINE INPUT": PRINT
30 PRINT: PRINT "***TEXTO A IMPRIMIR***"
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
```

TEXTO PERSONALIZADO

Já descobriu que quanto mais você trabalha com o seu Computador, mais coisas 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, instruindo 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 datilografia 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 ENTER
66
PRINT USING "#"; 66.2 ENTER
%66
PRINT USING "#.#"; 66.25 ENTER
%66.3

Você pode colocar o ponto decima em qualquer lugar do campo númerico estabelecido pelo sinal #. O Computador automaticamente arredondará qualquer digito à direta do ponto decimal que não couber no campo.

PRINT USING "# #.#" 58.76
58.8
PRINT USING "# #.# # " 10.2,5.3,66.789, . 234 ENTER
10.20 5.30 66.79 0.23

,

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 saturação (indicado pelo sinal de porcentagem na frente) é uma boa idéia colocar uma vírgula a cada três posições no campo numérico. A saturação ocorre porque o campo não tem espaços suficientes.

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 númerico.

PRINT USING "****# # # #**"; **44.0**
******44**

$

Se seu número representar dinheiro no caso dólar, coloque um $ na frente do campo numérico. Um $ será colocado na frente do número na saída.

PRINT USING "$# # #.# #";18.6735

$$

O sinal **$$** colocado no início do campo atuará como um sinal $ flutuante. O sinal ficará sempre 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, o número 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 deve ser impresso com notação exponencial.

```
PRINT USING "# #.# # # #↑↑↑↑";123456
1.2346E + 05
```

!

Um sinal de exclamação (!) faz com que o Computador imprima apenas o primeiro caracter da string.

```
PRINT USING "!"; "ANDREA"
A
```

%espaço%

Para especificar um campo de string de mais de um caracter, %espaço% é usado. O comprimento do campo será a quantidade de espaços entre os sinais de porcentagem, mais dois

```
PRINT USING "% %"; "ABCDEFG"
ABCD
PRINT USING "%      %"; "NADA A VER"
NADA A
```

Para verificar se ficou tudo bem entendido, rode esse programa:

```
5 CLS
10 A$ = "**$# #,# # #,# # #.# # DOLAR"
20 INPUT "QUAL SEU PRENOME"; P$
30 INPUT "QUAL SEU NOME"; N$
40 INPUT "QUAL SEU SOBRENOME";S$
50 INPUT "DIGITE A QUANTIDADE A PAGAR";P
60 CLS
```

```
70 PRINT "ORDEM DE PAGAMENTO PARA"
80 PRINT USING "!";P$;".";
90 PRINT S$
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 introduzir 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 fração com mais de dois dígitos será arredondado para dois.
O sinal de exclamação (!) na linha 80 diz ao Computador para usar apenas o primeiro caracter de P$(prenome) e N$(nome). Assim as linhas 80 e 90 imprimem as iniciais de seu nome junto com o sobrenome.

EM POSIÇÃO (POS)

POS é uma função de entrada/saída que permite testar a posição do cursos na tela ou na impressora.

POS (codigo do dispositivo)
código do dispositivo é 0 (tela) ou -2 (impressora)

POS fornece um número indicando a posição do cursos na tela ou a posição do carro na impressora.

```
PRINT TAB (8) POS(0)
```

O espaço na frente antes do "8" ocasiona que "8" apareça na coluna 9.

retorna o número 8 na coluna 8 da linha atual da tela (0). 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.

```
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 dle 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$(32)). Quando o Computador decide iniciar uma nova linha, ele faz isso imprindo um retorno de carro (CHR$(13)) — na verdade, o Computador imprime ENTER

Escolhemos para teste a posição 22 do cursor, já que ela tem 10 espaços a menos que a comprimento máximo da tela, 32, dando uma boa margem para completar uma palavra muito longa.

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 "em branco" enquanto a impressora trabalha, (caso você tenha uma) 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. Observe 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 caracter até:
- um caracter de retorno de carro que não foi precedido por um caracter de avanço dle linha;
- o fim de arquivo;
- o 249.º caracter 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$

Introduz 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 (CSAVE) no formato ASCII (usando a 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 CONTÉM";K"LINHAS"
```

CAPÍTULO 18

Últimas Mágicas

Há ainda umas poucas características do seu MX-1600 que você não teve a chance de conhecer. Neste capítulo, trataremos de cada uma delas.

POR VIA DAS DÚVIDAS

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 MX-1600 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#"

ou:

10 A$ = "A#"

Seu computador interpreta da mesma forma.

RASTREADOR

TRON e TROFF são dispositivos de depuração que ajudam 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 END
```

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 primeira a linha 10, depois a 20, a 30, a 40, 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"

TEMPORIZADOR

Seu MX-1600 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 retorna a zero el reco-

meça. (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 a qualquer tempo digitando:

TIMER = numero ENTER
onde número é um valor númerico 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),LS$(3) 'CH(#) = ESCOLHA, L$ = 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$="" THEN 240
250 SV = TIMER 'SE TECLA E' PRESSIONADA, SALVE CONTEÚDO DE TIMER
260 IF A$<"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 "VOCÊ LEVOU";SV/60; "SEGUNDOS"
310 INPUT "PRESSIONE <ENTER> PARA O PRÓXIMO PROBLEMA";EN
320 GOTO 80
```

Por tentativa e erro mude 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.

NÚMEROS NÃO DECIMAIS

O MX 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 octal. 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 Apêndice B.

CONVERSÃO (HEX$)

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$ (numero)
número é uma variável ou número decimal de 0 a 65535.

CAPÍTULO 19

Falando com a Máquina

Este capítulo se destina aos programadores avançados e descreve como chamar sub-rotinas em linguagem de máquina com programa em BASIC, e lista certas sub-rotinas da ROM que você pode achar proveitosas.

"Linguagem de Máquina" (ML) é a linguagem usada internamente por seu Computador. Consiste em instruções diretas ao microprocessador. (**Machine Language** em inglês) 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 ML, como segue:

1. Proteção de memória.
2. Armazenamento da sub-rotina ML 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 ML será uma sub-rotina simples. Ela obtém um caracter do teclado. O caracter é fornecido como um codigo 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 caracter 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 caracter.

Nós chamaremos a sub-roitna "INTEC". Para uma listagem desta subrotina, veja o fim desta seção.

Passo 1 — Proteger a memoria

Com o comando CLEAR, você pode reservar uma parte de RAM para armazenar sua sub-rotina ML. O primeiro parâmetro CLEAR separa o espaço para strings e o segundo determina o endereço de proteção de memória. Por exemplo:

```
10 CLEAR 25, 1200
```

separa o espaço para strings com 25 bytes e reserva a memória no final da RAM (ver mapeamento de memória). Seu programa ML pode então seguramente ser armazenado nesta área.

Passo 2 — Armazenar a sub-rotina de linguagem de maquina na RAM

Programas ML podem ser carregados da fita via CLOADM, ou colocados (POKE) na RAM. Em nosso exemplo, armazenaremos os códigos nas linhas DATA; então lemos e colocamos cada código na localização correta da RAM. Os números nas linhas DATA são os códigos da sub-rotina ML listada mais adiante nesta seção.

```
20 FOR I = 1 TO 28
30 READ B : POKE 12000 +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 — dizer ao BASIC onde esta 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 = 12000

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 ML. Quando este comando é encontrado, o BASIC chamará a sub-rotina ML.

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 hexa-decimal.*

Passo 5 — de volta 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 codigo objeto na RAM e então usa a subrotina 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**.

```
10 CLEAR 25, 12000 'MEMÓRIA DE RESERVA
15 CLS
20 FOR I=1 TO 28 'ARMAZENA CADA BYTE DO CÓDIGO OBJETO
30 READ B : POKE 12000 + I,B
40 NEXT I
45 'AQUI ESTA' O CÓDIGO 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 = 12000
110 A = USR1(0) 'CHAMA A SUB-ROTINA E PÕE O RESULTADO EM A
115 IF A=13 THEN END
120 PRINT "CÓDIGO="; A
130 GOTO 110
```

Para uma variação no programa, mude a linha 120 para:

```
120 PRINT CHR$ (A); 'EXIBE O CARACTER
```

LISTAGEM DA SUB-ROTINA EM ML

Nota: *Isto não é para ser digitado. Está aqui para aqueles que querem entender como a sub-rotina ML funciona.*

CODIGO OBJETO			
HEXADECIMAL	**FONTE**	**CÓDIGO**	**COMENTÁRIO**
AD 9F AO 00	LOOP1	JSR (POLCAT)	; Lê uma tecla
27 FA		BEQ LOOP1	; Se nada, busca outra
81 OA		CMPA ⩽10	; CTRL?
26 OC		BNE OUT	; Não, então, sai
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	TRF A,B	; Obter byte de retorno 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 codigo 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.*

VIRANDO

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 função 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.

RESERVANDO MEMÓRIA PARA 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.

ARMAZENANDO AS FUNÇÕES USR

As funções USR podem ser armazenadas 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 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 (HEX&0—FFFF).

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 USR

Para retornar ao BASIC de uma função USR, um RTS (Return Sub-Routine) 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 DE 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 LBS, 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 flutuan-

tes 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 da 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 (listados 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 atribui 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 da 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 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.

VARIÁVEIS DA IMPRESSORA SERIAL

VARIAVEL	END.HEX	END.DEC	INICIAL HEX	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 lista 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

End. decimal	Conteudo	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	MX BASIC	8000-BFFF
49152-65279	Espaço de cartucho	C000-FEFF
65280-65535	Entrada/saída	FF00-FFFF

Apêndices

EXEMPLOS DE PROGRAMA

TRIÂNGULOS

```
1 '***TRIANGULOS***
10 FOR A = 90 TO 0 STEP —4
15 S1=A*9: S2=191
20 A3=A/57.29578
30 X1=0:Y1=191
40 X2=S1 +X1: Y2=Y1
50 X3=X1+S2*COS(A3):Y3=Y1—S2*SIN(A3)
55 GOSUB 1000
90 NEXT A
99 GOTO 99
1000 PMODE 4,1
1005 PCLS
1010 SCREEN 1,0
1020 LINE (X1,Y1) — (X2,Y2), PSET
1030 LINE —(X3,Y3), PSET
1040 LINE —(X1,Y1), PSET
1060 RETURN
```

FORA-DENTRO

```
1 '***FORA-DENTRO***
2 '
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 <191 THEN 50
65 NEXT R
70 NEXT S
75 NEXT J,I
80 GOTO 30
```

CASACO ÍNDIO

```
1 '***CASACO INDIO***
2 '
5 PMODE 3,1
10 PCLS 4
15 SCREEN 1,0
20 COLOR 1,0
25 FOR X = 0 TO 255 STEP 18
30 OY = Y
35 Y = 30—OY
40 LINE (X,100—Y)—(X+10,100—OY),PSET
45 LINE (X,120+Y)—(X+10,120+OY),PSET
50 NEXT
60 FOR C = 2 TO 8
65 PAINT (0,110),C,1
70 NEXT
80 GOTO 5
```

BOMBA DE TEMPO

```
1 '***BOMBA***
2 '
10 PMODE 4,1
15 PCLS
20 SCREEN 1,1
25 CIRCLE (128,96),80
30 CIRCLE (128,96),90
35 PAINT (0,0),5
40 FOR T=30 TO —30 STEP —1
45 A=(2*3.1415)*T/60
50 LINE (128,96)—(75*SIN(A)+128,75*
   COS(A)+96),PSET
55 SOUND Q*2+1,20/(Q+1)+1
60 LINE (128,96)—(75*SIN(A)+128,75*
   COS(A)+96),PRESET
65 Q=60—2*T:FOR Y=Q TO 0 STEP —1:NEXT
70 NEXT
75 CLS
80 PCLS
85 PRINT @ 237, "BOOM!"
90 SOUND 1,30
95 PMODE 4,1
100 SCREEN 1,1
105 FOR I = 2 TO 200 STEP 2
110 CIRCLE (128,96),I
115 NEXT I
120 SCREEN 1,1
125 FOR X = 2 TO 200 STEP 2
130 CIRCLE (128,96),X,,3
135 NEXT X
140 FOR I = 2 TO 200 STEP 2
145 CIRCLE (128,96),I,3,.5
150 NEXT I
155 GOTO 155
```

ONDA SENOIDAL

```
1 '***ONDA SENOIDAL***
2 '
5 PMODE 4,1
10 PCLS
15 SCREEN 1,1
20 LINE (0,86)—(255,86),PSET
25 PI=3.14159
30 A1=—4*PI
35 A2=4*PI
40 N=180
45 R=50
50 X=(A2—A1)/N
55 F=255/(A2—A1)
60 FOR I =A1 TO A2 STEP X
65 X=I*F
70 Y=R*SIN(I)
75 PSET ((X+140),(80+Y),1)
80 NEXT I
90 GOTO 90
```

CAIXA ESTRANHA

```
1 '***CAIXA ESTRANHA***
2 '
5 PCLEAR 8
10 PMODE 3,1
15 PCLS
20 COLOR 6,5
25 DRAW"BM100,100U30NR30E15
   GR30NG15D30G16 NU30L30"
30 PAINT (105,95),8,6
35 PAINT (135,80),8,6
40 PAINT (110,65),8,6
45 SCREEN 1,1
50 FOR X = 1 TO 600: NEXT X
110 PMODE 3,5
112 PCLS
115 COLOR 6,5
120 DRAW "BM100,100U30NR30
    E20R30G20D30NL 30F20L30H20
125 LINE (100,100)—(70,95),PSET
130 LINE — (70,65),PSET
135 LINE — (100,70),PSET
140 LINE (70,95)—(40,65),PSET,B
145 LINE (130,100)—(160,95),PSET
150 LINE — (160,65),PSET
155 LINE — (130,70),PSET
160 PAINT (95,95),8,6
165 PAINT (105,95),8,6
170 PAINT (135,85),8,6
175 PAINT (45,85),8,6
180 PAINT (115,65),8,6
185 PAINT (125,114),8,6
190 SCREEN 1,1
195 FOR X = 1 TO 600: NEXT X
200 GOTO 10
```

GRÁFICOS ALEATÓRIOS

```
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)
50 NEXT: FOR P=0 TO 10
55 PAINT (RND(255),RND(191)),RND (4),F
60 NEXT: FOR H = 1 TO 7
65 FOR T=0 TO 600: NEXT T: GOTO 10
```

PING-PONG

```
5 CLEAR 12
8 INPUT "COR DE FUNDO (1—8)"; C
9 CLS(C)
10 X=13: Y=13
15 XM = 20: YM = 15
400 F=0
410 XT = X: YT = Y
420 X = X + XM: Y = Y + YM
430 TX = X: TY = Y: T1 = XM: T2 = YM
440 GOSUB 1000
450 X = TX: Y = TY: XM = T1: YM = T2
455 H = INT(XT/2)*2: V = INT(YT/2)*2
460 SET(H,V,C): SET(H+1,V,C)
462 SET(H,V+1,C): SET(H+1,V+1,C)
470 RESET(X,Y)
480 GOTO 400
1010 IF TX > 63 THEN TX = 63: T1 = —T1
1020 IF TX < 0 THEN TX = 0: T1 = —T1
1030 IF TY > 31 THEN TY = 31: T2 = —T2
1040 IF TY < 0 THEN TY = 0: 72 = —T2
1099 RETURN
```

LAÇO PINTADO

```
1 '***LAÇO PINTADO***
2 '
5 PMODE 3,1
10 PCLS
20 SCREEN 1,1
30 DRAW"BM50,180U60BU20U60R60BR20R60D60
   BD20D60L60BL20L60
40 DRAW"BM50,180U60R40BR20R80D20BL20L60
   BL20L20D20R20BR60R20U20
50 DRAW"BM50,180R60U80BU20U40L40BD20D20
   BD60D20R20U60BU20U20L20
60 DRAW"BM50,180U60BU40BR20R60BR20R20U20
   L20D60BD20D20R20
70 DRAW"BM50,180BR80U40BU20U80
80 DRAW"BM50,180BU80R80BR20R40
90 PAINT (85,128),6,8
95 PAINT (95,78),6,8
97 PAINT (155,95),6,8
98 PAINT (135,145),6,8
99 PAINT (128,185),7,8
100 PAINT (75,150),7,8
101 PAINT (160,150),7,8
102 PAINT (75,75),7,8
103 PAINT (160,75),7,8
104 PAINT (120,110),7,8
110 FOR X = 1 TO 600: NEXT X
200 GOTO 5
```

REPITA A MÚSICA

```
10 DIM M(50), T(8)
20 FOR B = 1 TO 8
30 READ T(B)
40 NEXT B
50 X = 1
60 M(X) = RND(8)
70 FOR Y = 1 TO X
80 CLS(M(Y))
90 PRINT @ 239, M(Y);
100 SOUND T(M(Y)), 8
110 NEXT Y
120 CLS
130 PRINT @ 231, "REPITA A MUSICA";
140 FOR Y = 1 TO X
150 T = 1
160 K$ = INKEY$
170 T = T + 1
180 IF T > 150 THEN 310
190 IF K$ = ""THEN 160
200 K = VAL(K$)
210 IF K <> M(Y) THEN 310
220 CLS(K)
230 PRINT @ 239, K;
240 SOUND T(K), 3
250 NEXT Y
260 X = X + 1
270 CLS: PRINT @ 230, "ESCUTE
    O PROXIMO TRECHO";
280 FOR T = 1 TO 500: NEXT T
290 CLS: PRINT @ 230, "ESCUTE
    O PROXIMO TRECHO";
300 GOTO 60
310 CLS(0)
320 PRINT @ 235, "NAO ACERTOU";
330 SOUND 1, 25
340 DATA 89, 108, 125, 133, 147, 159,
    170, 176
```

CALEIDOSCÓPIO

```
10 CLS0
20 X=RND(32)—1
30 Y=RND(16)—1
40 Z=RND(9)—1
50 GOSUB90
60 GOTO20
90 IFZ=0 OR RND(7)=3THEN150
100 SET(31—X,16+Y,Z)
110 SET(31—X,15—Y,Z)
120 SET(32+X,16+Y,Z)
130 SET(32+X,15—Y,Z)
140 RETURN
150 RESET (31—X,16+Y)
160 RESET (31—X,15—Y)
170 RESET (32+Y,16+Y)
180 RESET (32+X,15—Y)
190 RETURN
```

PLAYBACK DE SUA MÚSICA

```
5 DIM A(25), S$(13), B(200): Y=1
10 FOR X = 1 TO 25: READ A(X) NEXT X
20 DATA 89, 99, 108, 117, 125
30 DATA 133, 140, 147, 153, 159
40 DATA 165, 170, 176, 180, 185
50 DATA 189, 193, 197, 200, 204
60 DATA 207, 210, 213, 216, 218
70 FOR X = 1 TO 13: READ S$(X): NEXT X
80 DATA A,W,S,E,D,F,T,G,Y,H,U,J,K
90 CLS
92 PRINT @ 167, "COMPONHA SUA MUSICA"
94 PRINT @ 227, "UTILIZE TECLAS DA 2A.
   E 3A. FILEIRA"
96 PRINT @ 292, "DIGITE <x> QUANDO
   TERMINAR"
100 P$ = INKEY$
110 IF P$ = ""THEN 100
115 FOR X = 1 TO 13
120 IF P$ <> S$(X) THEN 150
130 SOUND A(X), 5
140 B(Y) = X
145 Y = Y + 1
150 NEXT X
160 IF P$ <> "X" THEN 100
165 CLS
170 PRINT @ 202, "SUA MUSICA EM PLAYBACK"
174 PRINT @ 264, "QUAL TECLA (1—11)";
176 INPUT K
180 FOR X = 1 TO Y—1
190 SOUND A(B(X)+K), 5
200 NEXT X
210 GOTO 165
```

TOCANDO BACH

```
5 CLS
10 PRINT @ 96, STRING$(32,"*")
20 PRINT @ 320, STRING$(32,"*")
25 PRINT @ 201, "BACK TO BACH"
40 FOR X = 1 TO 1000: NEXT X
55 A$ = "TG;O2;L2;G;L4;C;D;E;F;L2;G;C;P16;
   C"
60 B$ = "L2;A;L4;F;G;A;B;O3;L2;C;O2;C;P16;C;
   F;L4;G;F;E;D"
65 C$ = "L2;E;L4;F;E;D;C;L2;01;B;02;L4;C;D;
   E;C"
70 D$ = "L2;E;L1;D;L2;G;L4;C;D;E;F;L2;G;C;
   P16;C"
75 E$ = "L2;A;L4;F;G;A;B;O3;L2;C;O2;C;P16;C;
   F;L4;G;F;E;D"
80 F$ = "L2;E;L4;F;E;D;C;D;E;L2;F;O1;B;L1;O2;
   C"
85 X$ = "XA$;XB$;XC$;XD$;XE$;XF$;"
90 PLAY X$
```

GUERRA ESPACIAL

```
10 CLEAR 1000
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
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
```

DADO ELETRÔNICO

```
4 CLEAR 2000
5 CLS(3)
6 DIM DB$(6)
8 DIM DF(21), P(6), D$(6)
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
130 FOR X = 1 TO 6
140 READ P(X)
150 NEXT X
160 DATA 1, 2, 4, 7, 11, 16
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+96)
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
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
550 PRINT @ 113, "PRESSIONE QUALQUER
    TECLA";
560 PRINT @ 145, "PARA PROXIMA RODADA";
570 K$=INKEY$: IF K$="" THEN 570
580 GOTO 500
```

VENTILADOR

```
1 '***VENTILADOR***
2 '
5 PCLEAR 8
50 GOTO 600
60 LINE ((255—X),(191—Y))—(X,Y),PSET
61 J = J+1:IF J>A THEN J=0:
   A=RND(50)
63 RETURN
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
```

COMPOSIÇÃO DE MÚSICA

```
10 INPUT "COMPRIMENTO (1—10"; M
20 M = M*4
30 INPUT "RITMO (1—4)"; T1
40 IF T1 = 4 THEN 60
50 T = T1 : GOTO 70
60 T = 8
70 FOR K = 1 TO M*8
80 GOSUB 1000
90 B = RND(3) * T
100 SOUND P, B
110 CLS(S)
120 NEXT K
130 IF RND(10) <=8 THEN 150
140 SOUND 125, 16*T
145 END
150 SOUND 90, 16*T
160 END
1000 X = RND(100)
1010 IF X <= 20 AND X <= 25 THEN P = 90 : S = 1
1020 IF X > 20 AND X <= 25 THEN P = 108 : S = 2
1030 IF X > 25 AND X <= 40 THEN P = 125 : S = 3
1040 IF X > 40 AND X <= 55 THEN P = 133 : S = 4
1050 IF X > 55 AND X <= 75 THEN P = 147 : S = 5
1060 IF X > 75 AND X <= 85 THEN P = 159 : S = 6
1070 IF X > 85 AND X <= 95 THEN P = 176 : S = 7
1080 IF X > 95 THEN P = 58 : S = 8
1090 RETURN
```

ESTUDANDO PROJEÇÕES

```
1 '***ESTUDO DE PROJECAO***
2 '
5 PMODE 4,1
10 PCLS
15 SCREEN 1,0
20 DRAW "BM50,50R60D10NL20D20L20NU20L20NU
   20L20U20NR20U10"
25 DRAW"BM50,100R20ND20R20ND20R20D20NL20D
   10L60U10NR20U20"
30 DRAW "BM150,100R30D30L30U10NE20U20"
40 DRAW "BM150,50U5E15R10BF20BD30NR5L20H
   25U10
45 DRAW"BM150,50U5F8U15R15H8F8L15F8NR15D
   15F8ND10E15NR10H8
50 LINE (175,30) — (200,55), PSET
55 LINE — (200,80), PSET
60 LINE (167,60) — (183,46), PSET
65 GOTO 65
```

ESPECIFICAÇOES TECNICAS

FORNECIMENTO DE TENSAO CA

Tensão	105 — 130 V ou 208 — 232 V; 60 ou 50 Hz
Corrente	0.18 Amps RMS

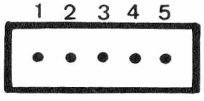
MICROPROCESSADOR

Tipo	6809E
Clock	0.895 MHz

INTERFACE SERIAL

	Sinal padrão RS-232C	Pino
Vcc	+ 5 Volts	1
CD	Deteção da portadora	2
GND	Tensão de referência zero	3
RD	Recepção dos dados	4
TD	Transmissão de dados da saída	5

LOCALIZAÇAO DOS PINOS DA RS-232C



SOFTWARE 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

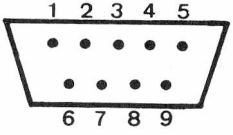
INTERFACE PARA CASSETE

Nível de 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 Vcc
Nível de entrada para reprodução do gravador:
1 a 5 volts pico a pico em uma impedância mínima de 200 ohms

TOMADA DO CONTROLADOR JOYSTICK

1 p/cima-Entrada do comparador V
2 p/baixo- Vcc
3 esquerda- Entrada do comparador H
4 direita- Vcc
6 Botão de tiro
8 Terra

LOCALIZAÇAO DOS PINOS



NOTA SOBRE A REDE ELETRICA

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. Se 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. Qunato mais complexa e séria a aplicação, maior a importância que você deve dar para a escolha de fonte de alimentação ideal para seu Computador.

TABELA DE ERROS

ABREVIAÇAO	DEFINIÇAO
/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 a tecla **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. Freqüente causado por tentar entrar um programa ou dados de arquivo em uma fita ruim.
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 caracter ilegal. Redige 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.

INSTRUÇOES DE IMAGEM E SOM

Estes são os códigos para cada uma das nove cores disponíveis no seu **MX-1600**.

CODIGO	COR
0	Preto
1	Verde
2	Amarelo
3	Azul
4	Vermelho
5	Cinza
6	Ciano
7	Magenta
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 + caracter

Por exemplo, se você quiser o caracter gráfico 7, na cor vermelha, faça:

```
CODIGO = 112 + 16 * 4 + 7
? CHR$(CODIGO);
```

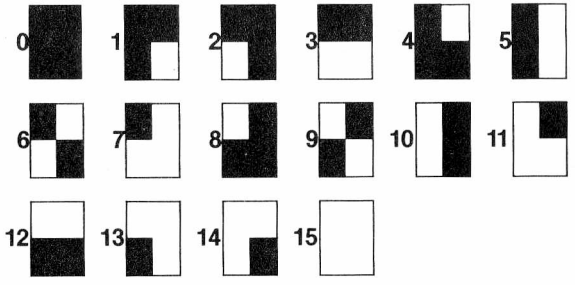
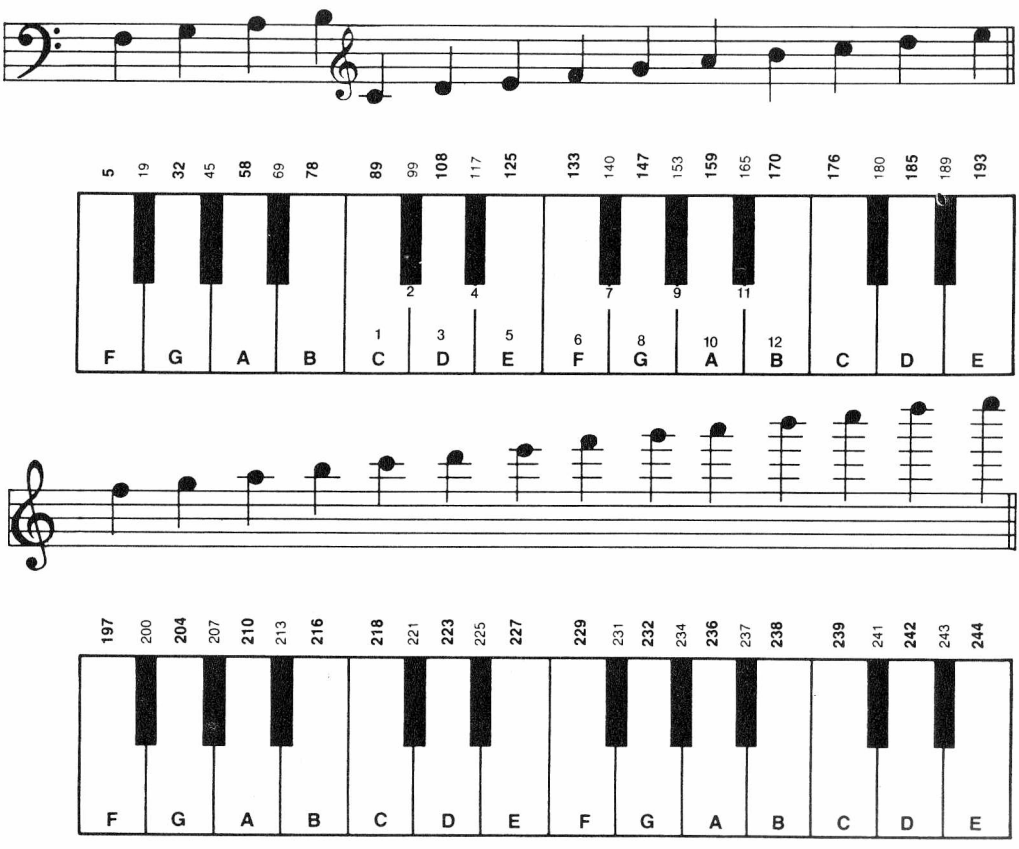


TABELA DOS CONJUNTOS DE CORES			
PMODE n	SCREEN 1,c	DUAS CORES	QUATRO CORES
0	0	preto/verde	
	1	preto/cinza	
1	0		verde/amarelo/azul/vermelho
	1		cinza/ciano/magenta/laranja
2	0	preto/verde	
	1	preto/cinza	
3	0		verde/amarelo/azul/vermelho
	1		cinza/ciano/magenta/laranja
4	0	preto/verde	
	1	preto/cinza	

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.

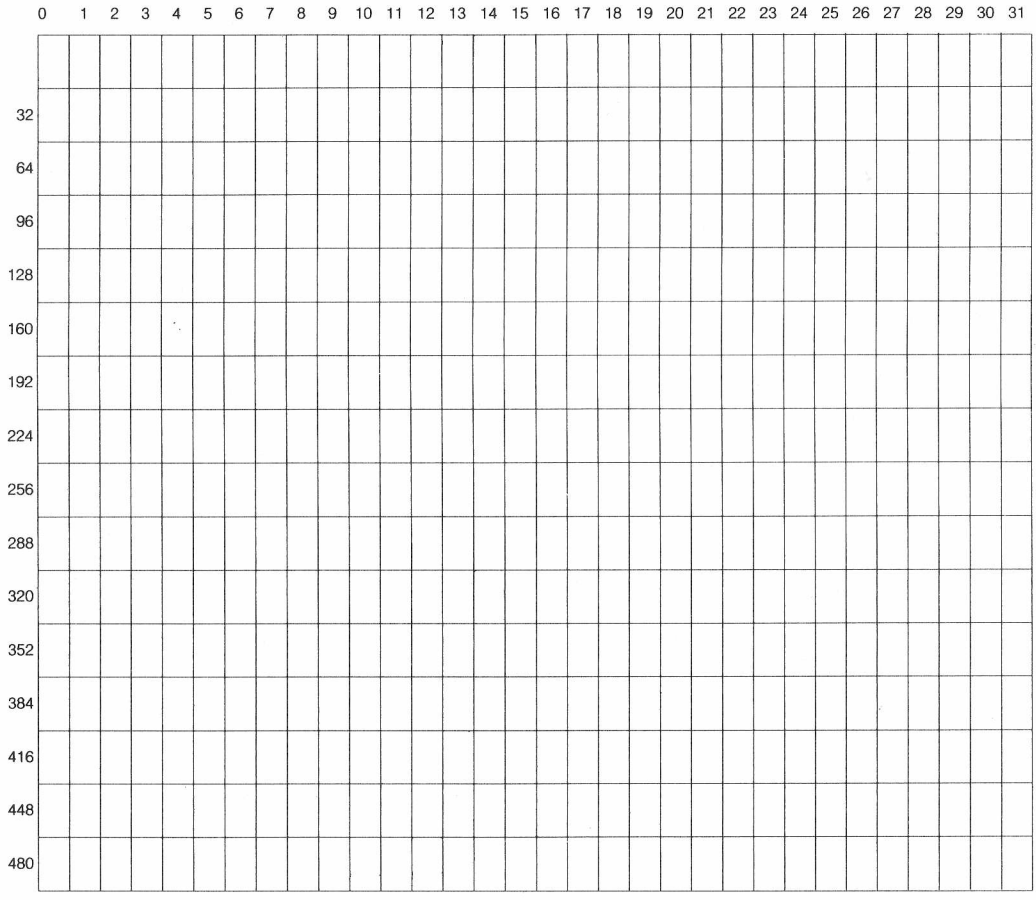
ESCALA MUSICAL



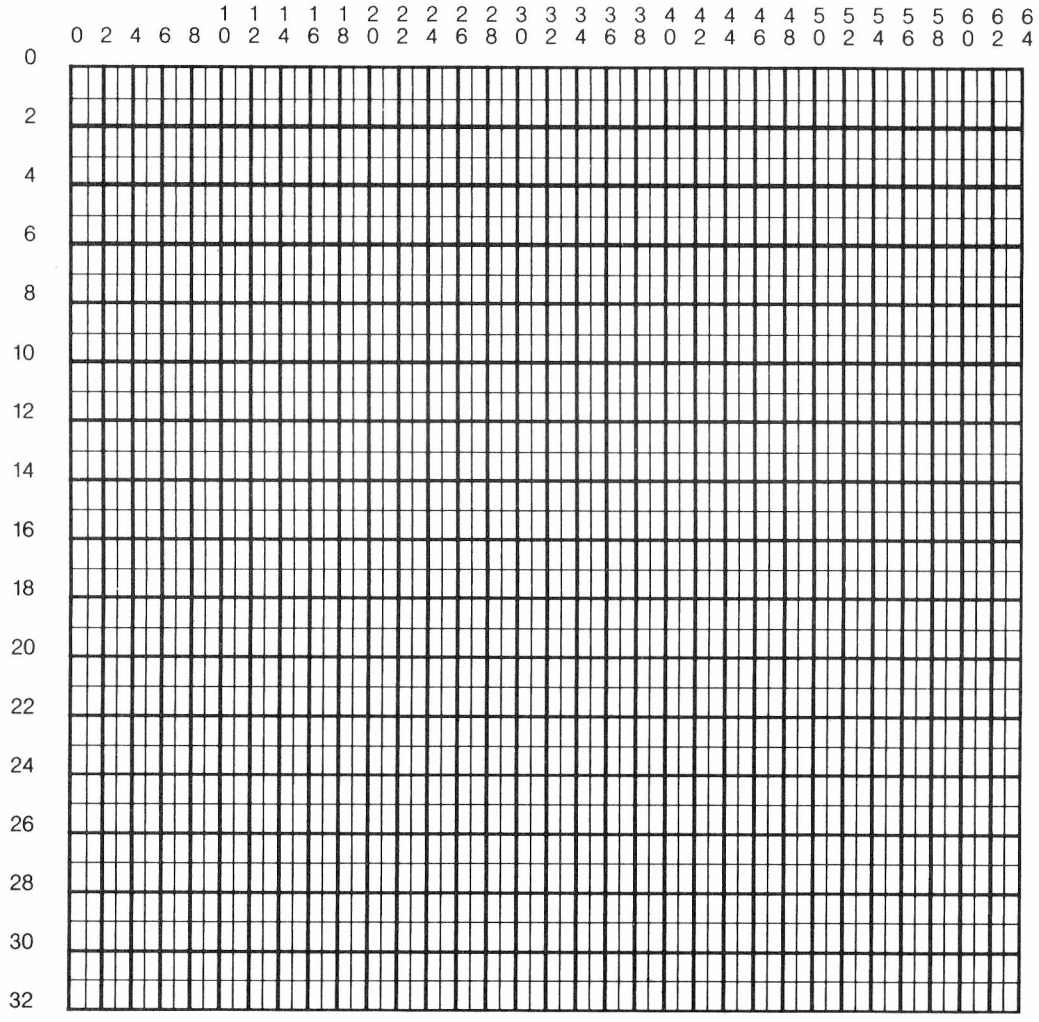
RESOLUÇOES GRAFICAS

A distribuição de mensagens na tela pode ser realizada com relativa simplicidade e utilizando-se a instrução PRINT @. A figura a seguir representa a tela e numera as diferentes posições possíveis.

POSIÇOES DO PRINT @ NA TELA



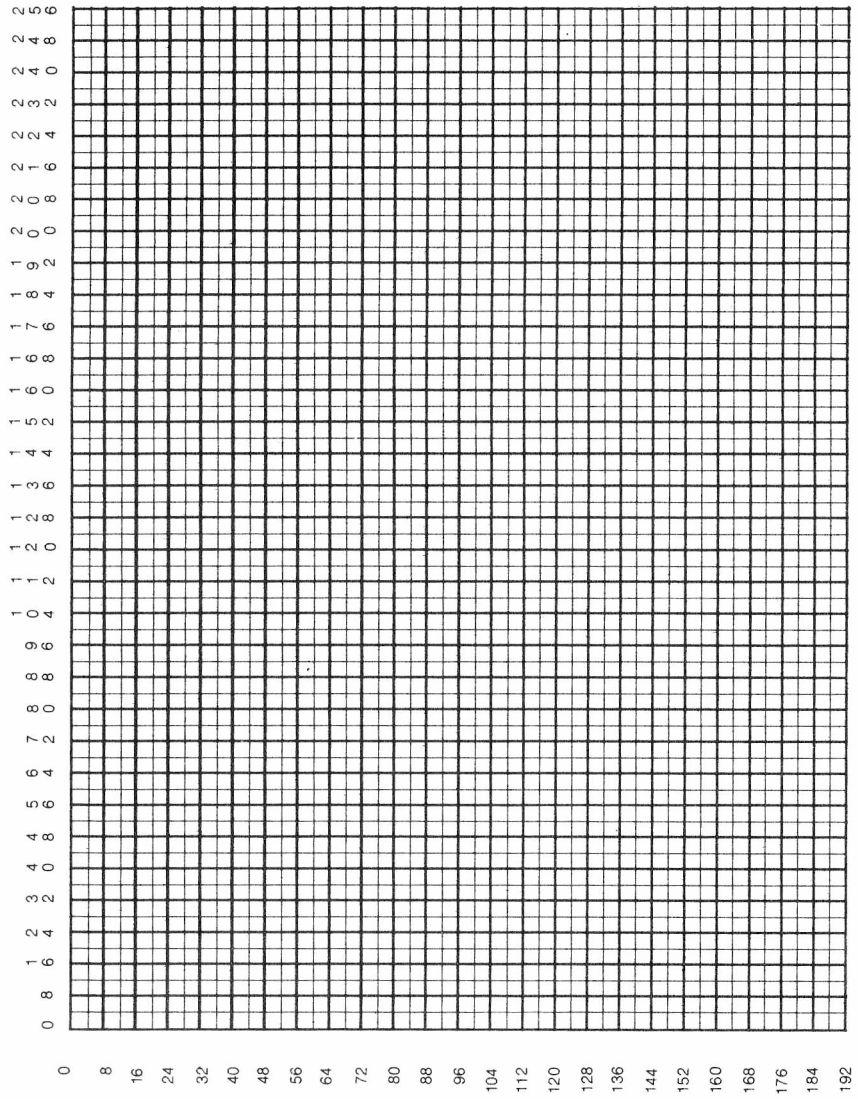
POSIÇOES GRAFICAS NA TELA
BLOCOS GRAFICOS E TEXTO



Ainda na tela de textos, utilizando-se a função CHR$ ou as instruções SET e RESET, pode-se misturar desenhos elementartes com texto. Esse recurso permite uma resolução (detalhamento) baixa, mas que pode ser útil em alguns casos.

POSIÇOES GRAFICAS NA TELA
MODOS GRAFICOS 0 A 4

Os recursos estritamente gráficos do MX-1600 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, o quadro que apresentamos a seguir, apesar do diferente número de divisões, têm a mesma numeração para os eixos X e Y.



CODIGO ASCII

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
↓	0A	5B	10	91
←	08	15	8	21
→	09	5D	9	93

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	32	—	50	—
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	—

PALAVRAS RESERVADAS DO MX-BASIC

PALAVRAS RESERVADAS					
'	CLS	FOR	MID$	PPOINT	SQR
*	COLOR	GET	MOTOR	PRESET	STEP
+	CONT	GOSUB	NEW	PRINT	STOP
–	COS	GOTO	NWXT	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	

OPERADORES DO BASIC	
OPERADOR	**CARACTERISTICA**
+	Adição
+	Combina strings
+	(PLAY)Sustenido
–	Subtração
–	(PLAY)Bemol
–	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)

SIMBOLOS DO BASIC	
SIMBOLO	DEFINIÇAO
""	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.

CARACTERES DO TECLADO	
CARACTERES	CARACTERISTICA
←	Retrocede o cursor
ENTER	Diz ao Computador que o programa ou a linha de comando chegou ao fim.
BREAK	Pára a execução do programa.
SHIFT @	Suspende a execução do programa. (Pressione qualquer tecla para continuar).
SHIFT 0	Introduz modo maiúsculo/minúsculo.
SHIFT ←	Apaga a linha atual.
SHIFT →	Introduz um colchete direito.
SHIFT ↓	Introduz um colchete esquerdo.

FORMULAS MATEMATICAS

TOPICO	FORMULAS	INSTRUÇAO EM BASIC
Soma dos ângulos de um triângulo	$180° = A+B+C$	TTL = AA+AB+AC
Solução 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))
Dados lados A, B e C	$s = ½(a+b+c)$ $\text{Área} = \sqrt{s(s-a)(s-b)(s-c)}$	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 $a = \frac{\text{sen } A.b}{\text{sen } B}$	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	$\frac{a-c}{a+c} = \frac{\tan ½(A-C)}{\tan ½(A+C)}$ ou $\tan ½(A-C) = \frac{a-c}{a+c} \cdot \tan ½(A+C)$	Y = TAN(AA−AC)/2) Y = (SA−SC)/(SA+SC)*TAN((AA+AC)/2)
Dados 3 lados, solução por um ângulo	$s = ½(a+b+c)$ $r = \sqrt{\frac{(s-a)(s-b)(s-c)}{s}}$ $A = 2 \arctan\left(\frac{r}{s-a}\right)$	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 Logarítmica	$(a^x)^y = A^{xy}$ $a^{-x} = \frac{1}{a^x}$ $\log x^y = y \cdot \log x$ $\log xy = \log x + \log y$ $\log \frac{x}{y} = \log x - \log y$	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)

FUNÇAO	FUNÇOES DERIVADAS FUNÇAO ESCRITA EM BASIC X EM RADIANOS
Secante	SEC(X) = 1/COS(X)
Co-secante	CSC(X) = 1/S/N(X)
Co-tangente	COT(X) = 1/TAN(X)
Seno Inverso	ARCS/SIN(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((SGN(X)*SQR(X*X+1)+1/X)
Co-secante Hiperbólica Inversa	ARGSCH(X)3LOG(SGN(X*X+1)+1/X)
Co-tangente Hiperbólica Inversa	ARGCOTH(X)=LOG((X+1)/(X−1))/2

FAIXAS DE ENTRADAS VALIDAS	
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 RAN(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
```

RESUMO DO MX-BASIC

PALAVRA	CARACTERISTICA	EXEMPLO
ABS	Calcula o valor absoluto.	Y = ABS(X)
ACS	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.	? CHE$(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 a terminar em pontos espedificados (0-1).	CIRCLE (128,96),50,4,1,.5,./5
CLEAR	Reserva e armazena *n* bytes de espaço string.	CLEAR CLEAR 500 CLEAR 100,14000
CLOAD	Carrega arquivo de program 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 explicado.	CLOADM"PROG" CLOADM CLOADM"PROG",1000
CLOSE	Fechar arquivos ou dispositivos.	CLOSE CLOSE-2
CLS	Limpa a tela 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	Armazena 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 4E,6F,5F

PALAVRA	CARACTERISTICA	EXEMPLO
DATA	Usar READ para atribuir esses dados a variáveis.	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 DEL 30–70
DIM	Dimensiona uma ou mais matrizes	DIM R(65),W(40) DIM (AR$(8,25)
DLOAD	Carrega um programa BASIC numa velocidade específica. 0 = 300 baud 1 = 1200 baud.	DLOAD X,1
DRAW	Desenha uma linha como ponto de partida, comprimento e cor específicos. **Também desenha em escala, linhas em branco, linhas não atualizadas e executa substrings. Se o ponto de partida não foi especificado, assume-se (128,96) ou a última posição de DRAW é usada.**	DRAW "BM100,100;S10;, 25;BR25;ND25;XA$;"
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*S*c* 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.	

PALAVRA	CARACTERISTICA	EXEMPLO
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 no 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** **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(1)

PALAVRA	CARACTERISTICA	EXEMPLO
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 LIST 20-
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 (1) na tela ou teclado (0), fita (—1), impressora (—2) para entrar (I) ou (O).	OPEN "I",0,"ARQUIVO" OPEN "O",—1,"DADOS"

PALAVRA	CARACTERISTICA	EXEMPLO
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$;"
PMODE	Seleciona resolução e página de memória para iniciar.	PMODE 4,3
POINT	Testa se o ponto gráfico especificado está aceso ou apagado, X(horizontal) = 0-63; Y(vertical) = 0-31. O valor retornado é -1, se o ponto está no modo caractere; 0, se está apagado; ou o código de cor, se está aceso.	IF POINT (X,Y) THEN PRINT "ACESO" ELSE PRINT "APAGADO"
POKE	Coloca um valor em uma localização de memória específica.	POKE 18572,255
POS	Fornece a posição atual do cursor.	PRINT TAB(8) POS(0)
PPOINT	Testa se o ponto gráfico especificado está aceso ou apagado e fornece o código da cor do ponto especificado.	PPOINT (13,15)
PRESET	Coloca a cor de fundo.	PRESET(5,6)
PRINT	Imprime mensagem especificada na tela de TV.	PRINT "OLA"
PRINT #-1	Grava dados na fita	PRINT = -1,A
PRINT #-2	Imprime um item ou lista de itens na impressora.	PRINT = -2,CAP$
PRINT TAB	Move o cursor para uma posição de coluna específica e imprime	PRINT TAB(5) "NOME"

PALAVRA	CARACTERISTICA	EXEMPLO
PRINT USING	Imprime números no formato especificado.	
	# Formata números.	PRINT USING "# # # #";12;3
	. Ponto decimal.	PRINT USING "# #.#";18.3
	, Coloca vírcula à direita de cada três números.	PRINT USING "# # # #,";12.0
	**Preenche espaços da frente com astericos	PRINT USING "** # #;# #";12.3
	$Coloca $ na frente.	PRINT USING "$# #.# #";;12.3
	$$Sinal de dinheiro flutuante.	PRINT USING "$$00.0";12.345
	**$Sinal de dinheiro flutuante.	PRINT USING "**$#.# #;1.234
	+ Na primeira posição ocasiona sinal a ser impresso. Na última posição, ocasiona sinal a ser impresso atrás do número.	PRINT USING "+ # #.#;-123
	↑↑↑↑ Formato exponencial.	PRINT USING "# #.#";123
	- Coloca o sinal depois de números negativos.	PRINT USING "# #.#';-123.4
	! Retorna o primeiro caractere da string.	PRINT USING "!";"ANIL"
	%espaço%Campo da string; o comprimento do campo é o número de espaços mais 2.	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 avriáeis 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

PALAVRA	CARACTERISTICA	EXEMPLO
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 = TANG(45 + 7)
TIMER	Fornece o conteúdo do temporizador ou permite seu ajuste.	PRINT TIMER TIMER = 0

PALAVRA	CARACTERISTICA	EXEMPLO
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))

MX-1600
64K COLOR COMPUTER

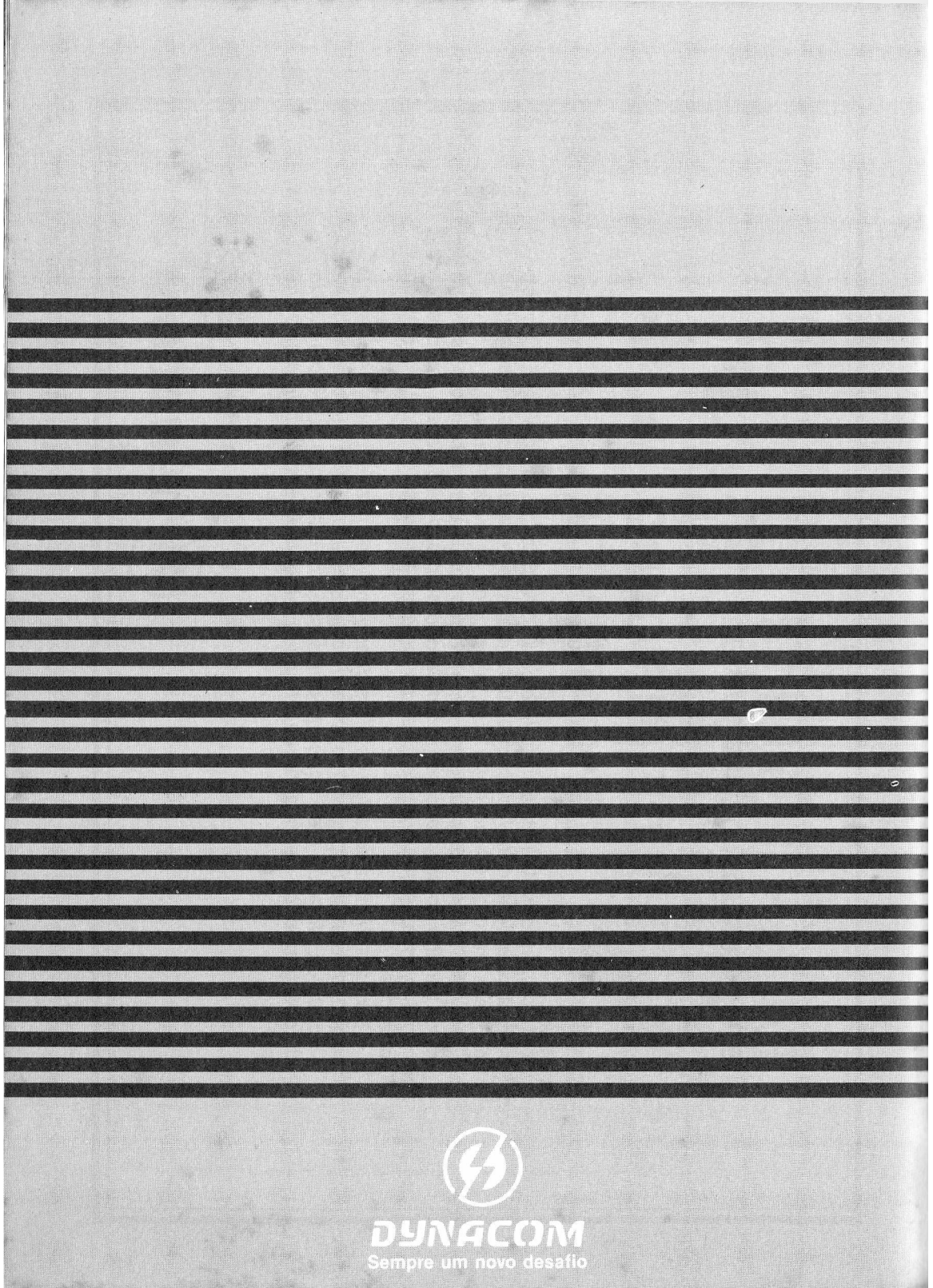
CERTIFICADO DE GARANTIA

O Computador MX-1600 é garantido pela DYNACOM ELETRÔNICA LTDA. contra qualquer defeito de fabricação, do aparelho e de seus acessórios integrantes, pelo prazo de 90 (noventa) dias, a partir da data da aquisição.

A DYNACOM se compromete a reparar, sem onus para o consumidor, qualquer aparelho apresentado como defeituoso no prazo da garantia e desde que entregue, livre de qualquer despesa de remessa às Aassistencias técnicas autorizadas.

Esta garantia não abrange defeitos decorrentes de uso indevido, negligência ou abuso do usuário, peças que sofram desgaste natural tais como conectores e soquetes, violação e ou conserto por pessoa não autorizada. Não estão incluídos, tampouco, nesta garantia, circuitos integrados, semicondutores e cristais.

DYNACOM


DYNACOM
Sempre um novo desafio

Fotocomposição, fotolito e impressão
Bandeirante S.A Gráfica e Editora
452-3444/572-0033

ERRATA DO MANUAL DE OPERAÇÃO

PÁG.	ONDE SE LÊ	LEIA-SE
1-7	20 FOR Y = 0 DO 31 ENTER	20 FOR Y = 0 TO 31 ENTER
3-5	A LINHA 10 DIZ AO COMPUTADOR QUE O PRIMEIRO NÚMERO DEVERÁ SER 15 E O ÚLTIMO 20.	A LINHA 10 DIZ AO COMPUTADOR QUE O PRIMEIRO NÚMERO DEVERÁ SER 16 E O ÚLTIMO 21.
4-2	OBSERVE QUE NAS INSTRUÇÕES IF/THEM...	OBSERVE QUE NAS INSTRUÇÕES IF/THEN (...)
4-6	50 FOR V= 7 TO 22	50 FOR V= 6 TO 24
4-8	30 IF H > 4 THEN RESET (H-1,15) 70 IF H > 59THEN RESET (H+1,15)	ACRESCENTAR: 35 RESET (3,15) 75 RESET (60,15)
5-3	SKIP "NOME" ENTER	SKIPF"NOME" ENTER
6-4	ADICIONE, TAMBÉM, ESTAS LINHAS NO FIM DO PROGRAMA PARA DAR AO COMPUTADOR ESPAÇO LIVRE PARA OUTROS STRINGS.	ISTO DÁ AO COMPUTADOR ESPAÇO LIVRE PARA OUTROS STRINGS. ADICIONE, TAMBÉM,ESTAS LINHAS NO FIM DO PROGRAMA.
	120 PRINT "O QUE SIGNIFICA A PALAVRA:" B$ 150 IF R$ = A$ THEN 190 170 PRINT "A PALAVRA CORRETA É: A$	120 PRINT "O QUE SIGNIFICA A PALAVRA:" A$ 150 IF R$ = B$ THEN 190 170 PRINT " A PALAVRA CORRETA É:" B$
6-8	40 IMPUT "QUER TENTAR OUTRA", A$	40 INPUT "QUER TENTAR OUTRA";A$
	5 CLEAR 500	5 CLEAR 517
	ADICIONE ESTA LINHA AO PROGRAMA: 5 CLEAR 500	(PARÁGRAFO SEM EFEITO,ELIMINAR)
6-10	I$ - - - AQUI ESTÁ UMA STRING	S$ - - - AQUI ESTÁ UMA STRING
6-11	85 ND	85 END
6-13	95 IF LEN (B$) 2 THEN 50	95 IF LEN (B$) < > 2 THEN 50
7-4	SE VOCÊ SIMPLESMENTE DISSER AO COMPUTADOR (...). E, ENTÃO, PRESSIONAR ESPAÇO (O CARACTER QUE VOCÊ QUER). SUA TELA TERÁ.	SE VOCÊ SIMPLESMENTE DISSER AO COMPUTADOR (...).E,ENTÃO, PRESSIONAR O CARACTER QUE VOCÍ QUER (R).SUA TELA TERÁ:
7-7	RENUM 1000 , 100 ENTER (...) O PROGRAMA INTEIRO SERÁ	RENUM 1000, 100 ENTER (...) O PROGRAMA INTEIRO SERÁ

PÁG	ONDE SE LÊ	LEIA-SE
	RENUMERADO INICIANDO COM UM NÚMERO DE NOVA LINHA 100, E INCREMENTADO DE 100.	RENUMERADO INICIANDO COM UM NÚMERO DE NOVA LINHA 1000, E INCREMENTADO DE 100. NOTA: MODO DE RENUM NÃO ACUSA ERRO AO SE INDICAR UM NÚMERO DE LINHA NÃO EXISTENTE NO PROGRAMA ORIGINAL.
7-8	RENUM 15,30 É ILEGAL, POIS TERIA QUE COLOCAR A LINHA 30 NA FRENTE DA LINHA 20	RENUM 15,30 É ILEGAL, POIS TERIA QUE COLOCAR A LINHA 30 ATRÁS (ANTES) DA LINHA 20
8-3	2010 INPUTA A :3030 RETURN 4020 IF A = X/Y THEN PRINT "CORRETO" ELSE PRINT "ERRADO"	2010 INPUT A 3030 RETURN 4020 IF A = INT (X/Y) THEN PRINT "CORRETO" ELSE PRINT "ERRADO"
8-6	OU, TÉCNICAMENTE, 1* 0^9 QUE É 1 VEZES 10 À NONA POTÊNCIA : (...)	OU, TÉCNICAMENTE, 1* 10^9 QUE É 1 VEZES 10 À NONA POTÊNCIA:(...)
8-7	50 IF A$ = CHR$ (8) THEN 70 H = -1 80 SET (H,+1,14)	50 IF A$ = CHR$ (8) THEN 70 70 H= H-1 ACRESCENTAR: 75 SET (H,14,3) 80 RESET (H + 1,14)
9-1	60 IF POINT (HV) < > THEN GOSUB 100	60 IF POINT (H,V) < > 0 THEN GOSUB 100 ACRESCENTAR: 105 FOR I = 0 TO 600 : NEXT I
9-2	94 SET (H,0,8): SET (H,1,8) 96 NEXT H 2070 ESTIVER NO CANTO SUPERIOR ESQUERDO	ACRESCENTAR: 93 FOR V = 0 TO 3 94 SET (H,V,8) 96 NEXT V,H 2070 DATA ESTIVER NO CANTO SUPERIOR ESQUERDO
9-9	500 INPUT "LOCALIZAÇÃO (0-243)";L 1025 PRINT @ L + 32*2,B$ 3000 LG$ = L2$:B$=B1$:RETURN 4000 LG$ = L3$:B$=B2$:RETURN	ACRESCENTAR: 190 NEXT X 500 INPUT "LOCALIZAÇÃO (128-242)"; L 1025 PRINT @ L + 32*2, B$; 3000 LG$=L2$:B$=BD$:RETURN 4000 LG$=L3$:B$=BD$:RETURN

PÁG.	ONDE SE LÊ	LEIA-SE
10-2	50 READ K,L,M,N	50 READ F,G,H,I,J
10-3	NA VERDADE FOI FIXADO (DIM) ESPAÇO PARA 15 ITENS (...)	NA VERDADE FOI FIXADO (DIM) 15 ESPAÇOS PARA 14 ITENS (...)
	96 INPUT "QUANTOS VOTOS A MAIS";X 97 A(Z) = A(Z) + X	96 INPUT "QUANTOS VOTOS A MAIS";N 97 A(N) = A(X) + N
10-5	70 FOR Y= 1 TO 460*S:NEXT Y :CLS 80 FOR X 1 TO 5	70 FOR Y = 1 TO 460* 5:NEXT Y : CLS 80 FOR X = 1 TO 5
10-7	115 IF R$ < > "SIM" THEN 200 125 IF N > 12 THEN 130	115 IF R$< >"SIM" THEN END 125 IF N > 12 THEN 120
10-11	80 OPEN "I" # -1, "LIVROS"	80 OPEN "I", # -1, "LIVROS" → ABRE A LINHA PARA O GRAVADOR
	85 IF EFO (-1) THEN 120 → ABRE A LINHA PARA O GRAVADOR	85 IF EOF (-1) THEN 120
	100 PRINT T$ IMPRIME → TÍTULOS NA FITA	100 PRINT # -1, T$ → IMPRIME TÍTULOS NA FITA
10-12	85 IF EOF (-1) THE 120	85 IF EOF (-1) THEN 120
10-16	30 IMPUT "DIGITE UMA PALAVRA"; A$ (I)	30 INPUT "DIGITE UMA PALAVRA"; A$ (I)
	30 READ A$ (1)	30 READ A$ (I)
10-18	NESTE PONTO, AS FIGURAS 50 e 60 FAZEM X IGUAL A 1 NOVAMENTE	NESTE PONTO, AS LINHAS 50 e 60 FAZEM X IGUAL A 1 NOVAMENTE
10-19	100 IF B < 0 OR B > 2 THEN 90	100 IF B < 1 OR B > 2 THEN 90
	1020 PRINT @ 132, "1A. VOTAÇÃO" A "VOTOS" 1040 FOR A = 1 TO 2	1020 PRINT @ 132, "VOTOS DO CANDIDATO" 1040 FOR B = 1 TO 2
10-20	1050PRINT "VOTAÇÃO" A	1050 PRINT "VOTAÇÃO" B ACRESCENTAR :2045 FOR B = D TO D
	2070 NEXT A	2070 NEXT B,A
10-21	1 D (1,2) 143 D (1,1,2) 678	1 D(1,1,1)143 D(1,1,2)678
	3 D (2,3,1) 254 D (2,3,9) 200	3 D(2,3,1)254 D(2,3,2)200

PÁG.	ONDE SE LÊ	LEIA-SE
	150 INPUT "VOTAÇÃO"< 1>, VOTAÇÃO < 2 > ";C	150 INPUT "VOTAÇÃO < 1 > ,VOTAÇÃO < 2 >"; C
10-22	1070 PRINT @P "CANDIDATO" B 1100 PRINT @ P + 8* C, D (A, B,C)	1070 PRINT @ P, "CANDIDATO" B 1100 PRINT @ P+ 11* C,D (A,B,C)
10-23	3040 PRINT @168, "CANDIDATO 1" 3050 PRINT @176, "CANDIDATO 2" 3060 PRINT @184, "CANDIDATO 3"	3040 PRINT @ 168, "CAND < 1 > " 3050 PRINT @ 176, "CAND < 2 > " 3060 PRINT @184, "CAND < 3 > "
11-2	QUE TAL AGORA UM PONTO NO CENTRO DIREITO INFERIOR DA TELA. SUBSTITUA A LINHA 35 (255, 191,8)	QUE TAL AGORA UM PONTO NO CANTO DIREITO INFERIOR DA TELA. SUBSTITUA A LINHA 35 POR (255,191,8).
11-4	20 LINE (0,0) -255,191) PSET 30 LINE - (191,0), PSTE ANTES DE EXPLICARMOS A DIFE-RENÇA ENTRE PST E PRESET, (...)	20 LINE (0,0) - (255,191), PSET 30 LINE - (191,0), PSET ANTES DE EXPLICARMOS A DIFE-RENÇA ENTRE PSET E PRESET, (...)
12-3	PMODE 4,1 PRETO/CINCA PMODE 3,1 CINZA/CLARO/MAGENTA/ LARANJA PMODE 1,1 CINZA/CLARO/MAGENTA/ LARANJA	PMODE 4,1 PRETO/CINZA PMODE 3,1 CINZA/CIANO/MAGENTA/ LARANJA ACRESCENTAR: PMODE 2,1 PRETO/CINZA PMODE 1,1 CINZA/CIANO/MAGENTA/ LARANJA
12-6	20 FOR A = 1 TO 8	20 FOR P= 1 TO 8 ACRESCENTAR: 65 CIRCLE (128, P* 15), 13
12-8	NÃO SE ILUDA ACREDITANDO QUE SCREN NÃO PODE AFETAR A TELA DE TEXTO	NÃO SE ILUDA ACREDITANDO QUE SCREEN NÃO PODE AFETAR A TELA DE TEXTO.
13-1	10 CLS	10 PCLS
13-3	30 CIRCLE (128,96),30,1,1,25,. 75	30 CIRCLE (128,96),30,1,.25,.75
13-11	h1v1e h2 v2 - PONTO INICIAL E FINAL SÃO OS MESMOS EXPLICADOS ACIMA	h1, v1e h2, v2 - PONTO INICIAL E FINAL SÃO OS MESMOS EXPLICADOS ACIMA

PÁG.	ONDE SE LÊ	LEIA-SE
14-2	10 FOR N= 1 TO 12 'N= NOTA	10 FOR N= 1 TO 12
14-5	20 B$ ="O5; X A $ "	20 B$ ="O5; XA$;"
15-3	60 AA=AA/57.29577951: AB/57.29577951: AC=AC/57.29577951' CONVERTE GRAUS EM RADIANOS 80SB= ((SIN(AA)) / (SIN(AC))) * SC:IF SB < 0 TEM 100	60 AA= AA/57.29577951: AB= AB/57.29577951: AC= AC/57.29577951' CONVERTE GRAUS EM RADIANOS 80 SB= (SIN (AA) / SIN (AC)) *SC: IF SB < 0 THEN 100
15-4	10 INPUT " QUAL O ÂNGULO C (AC)"; AC:IF <= 0 OR AC >= 180 THEN 100	10 INPUT "QUAL O ÂNGULO C (AC)"; AC:IF AC<=0 OR AC>=180 THEN 100
15-5	20 INPUT "QUAL O ÂNGULO A(AA)" AA:IF<= 0 OR AA >= 180 THEN 100 40 INPUT " QUAL O ÂNGULO B(AB)"; A:IF AB<=0 OR AB>= 180 THEN 150	20 INPUT "QUAL O ÂNGULO A (AA)";/ AA: IF AA < = 0 OR AA > = 180 THEN 100 40 INPUT "QUAL O ÂNGULO B (AB)"; AB: IF AB < = 0 OR AB > = 180 THEN 150
15-6	EXP (NÚMERO) NÚMERO É UM VALOR NUMÉRICO MENOR QUE 87.3365	EXP (NÚMERO) NÚMERO É UM VALOR NUMÉRICO MENOR QUE 88.0296
16-1	SE VOCÊ RECORRER AO APÊNDICE DE CÓDIGOS ACSII DOS CARACTERES, (...)	SE VOCÊ RECORRER AO APÊNDICE DI CÓDIGOS ASCII DOS CARACTERES, (...)
16-2	32 PRINT F 45 PRINT LEFT$ (S$,F-1)+STRING $ (LEN($),CHR$(159))+RIGHT$(S$), LEN(S$-F-LEN(T$)+1) 55IFP<=LEN(S$)-LEN(R$)+1THEN30	32 PRINT C+1 45 PRINT LEFT$ (S$,F-1)+STRING $ (LEN(T$), CHR$ (159))+RIGHT$ (S$,LEN (S$) -F-LEN (T$)+1) 55 IFP<=LEN(S$) - LEN(T$)+1THEN30
16-3	50 PRINT ISNTR (X$,A$) 20 MID $ (A$,14) ="RJ"	50 PRINT INSTR (X$,A$) 20 MID$ (A$,8) = "RJ"
17-3	PRINT USING "$ # # # . # # " ; 18.6735	PRINT USING "$ # # # . # #"; 18.6735 $ 18.67
17-5	80 PRINT USING " ! " ; P$;"..";	80 PRINT USING "!"; P$; "."; N$; "..";

PÁG.	ONDE SE LÊ	LEIA-SE
19-5	(...) (INTCNV = X'B3ED').	(...) (INTCNV = & H'B3ED').
19-6	(...) (GIVABF = X'B 4F4') (...)	(...) (GIVABF = &H'B4F4') (...)
19-7	A INTERFACE RS 232 USA UM CONECTOR DIN DE 4 PINOS. UM DIAGRAMA DOS PINOS É MOSTRADO NO CAPÍTUILO 1	A INTERFACE RS 232 USA UM CONECTOR DE 5 PINOS. UM DIAGRAMA DOS PINOS É MOSTRADO NO APÊNDICE B.
19-8	9216-2559 PÁGINA 6 2400 -9FF 2560-12387 PÁGINA 7 2A00 -2FFF	9216-9559 PÁGINA 6 2400 - 29FF 9560-12287 PÁGINA 7 2A00 - 2FFF
ÍNDICE	APÊNDICES Exemplos de Programas Especificações Técnicas Tabela de Erros Instruções de Imagem e Som Resoluções Gráficas Código ASCII Palavras Reservadas do MX-BASIC Fórmulas Matemáticas Resumo do MX - BASIC	APÊNDICE A B C D E F G H I
A	FORA-DENTRO 60 IF A <255 AND <191 THEN 50 PING-PONG 10 40 IF TY < 0 THEN TY=0 :72=-T2 PLAYBACK DE SUA MÚSICA 10 FOR X= 1 TO 25: READ A (X) NEXT X TOCANDO BACH 55 A$="TG;O2; L2; G; L4; C; D; E; F; L2; G; C; P16; C" VENTILADOR 50 GOTO 600	FORA-DENTRO 60 IF A < 255 AND C < 191 THEN 50 PING-PONG 1040 IF TY < 0 THEN TY = 0:T2=-T2 PLAYBACK DE SUA MÚSICA 10 FOR X = 1 TO 25 : READ A(X): NEXT X TOCANDO BACH 55 A$="G; O2;L2;G;L4;C;D;E;F;L2; G;C;P16;C" VENTILADOR 50 GOTO 601

F

TECLA	HEXADECIMAL		DECIMAL		TECLA	HEXADECIMAL		DECIMAL	
	SEM SHIFT	COM SHIFT	SEM SHIFT	COM SHIFT		SEM SHIFT	COM SHIFT	SEM SHIFT	COM SHIFT
>	—.—	3E	—.—	62	BRK	03	03	03	03
?	—.—	3F	—.—	63	CLR	0C	5C	12	92
@	40	13	64	19	ENT	0D	0D	13	13
A	41	61	65	97	ESPAÇO	20	—.—	32	—.—
B	42	62	66	98	!	—.—	21	—.—	33
C	43	63	67	99	,,	—.—	22	—.—	34
D	44	64	68	100	#	—.—	23	—.—	35
E	45	65	69	101	$	—.—	24	—.—	36
F	46	66	70	102	%	—.—	25	—.—	37
G	47	67	71	103	&	—.—	26	—.—	38
H	48	68	72	104	'	—.—	27	—.—	39
I	49	69	73	105	(	—.—	28	—.—	40
J	4A	6A	74	106	)	—.—	29	—.—	41
K	4B	6B	75	101	*	—.—	2A	—.—	42
L	4C	6C	76	108	+	—.—	2B	—.—	43
M	4D	6D	77	109	,	2C	—.—	44	—.—
N	4E	6E	78	110	-	2D	—.—	45	—.—
O	4F	6F	79	117	.	2E	—.—	46	—.—
P	50	70	80	112	/	2F	—.—	47	—.—
Q	51	71	81	113	0	30	—.—	48	—.—
R	52	72	82	114	1	31	—.—	49	—.—
S	53	73	83	115	2	32	—.—	50	—.—
T	54	74	84	116	3	33	—.—	51	—.—
U	55	75	85	117	4	34	—.—	52	—.—
V	56	76	86	118	5	35	—.—	53	—.—
W	57	77	87	119	6	36	—.—	54	—.—
X	58	78	88	120	7	37	—.—	55	—.—
Y	59	79	89	121	8	38	—.—	56	—.—
Z	5A	7A	90	122	9	39	—.—	57	—.—
↑	5E	5F	94	95	:	3A	—.—	58	—.—
↓	0A	5B	10	91	;	3B	—.—	59	—.—
←	08	15	8	21	<	—.—	3C	—.—	60
→	09	5D	9	93	=	—.—	3D	—.—	61

PÁG.	ONDE SE LÊ	LEIA-SE
G	PALAVRAS RESERVADAS: NWXT	PALAVRAS RESERVADAS: NEXT
H	FÓRMULAS MATEMÁTICAS LEI DOS SENOS a/b = Sen A/Sen B ou a = Sen A.b / Sen B	FORMULAS MATEMÁTICAS LEI DOS SENOS a/b = Sen A/Sen B ou a = Sen A/Sen B . b
	LEI DAS TANGENTES Y=TAN (AA-AC)/2)	LEI DAS TANGENTES Y=TAN ((AA-AC)/2)
	EQUAÇÃO LOGARÍTMICA Z= (A ↑ (-X) ou Z=1/(A ↑ X) Z=LOG (X*Y)ou Z* LOG (X)+(LOG (Y)	EQUAÇÃO LOGARÍTMICA Z = A ↑ (-X)OU Z=1/(A ↑ X) Z = LOG(X*Y) OU Z=LOG(X)+LOG(Y)
	CO-SECANTE CSC(X) = 1/S/N/(X)	CO-SECANTE CSC(X)=1/SIN(X)
	SENO INVERSO ARCS/SIN(X) = ATN (X/SQR (-X*X+1))	SENO INVERSO ARCSIN(X)= =ATN(X/SQR(-X*X+1))
	SENO HIPERBÓLICO SINH(X)=(EXP(X)-EXP(-X)/2	SENO HIPERBÓLICO SINH(X)=(EXP(X) -EXP(-X)) /2
	SE PEDIRMOS AO COMPUTADOR O VALOR DE RAN (90/57.29577951) (...)	SE PEDIRMOS AO COMPUTADOR O VALOR DE TAN (90/57.29577951) (...)
	RESUMO DO MX-BASIC CHR$ EX.: ? CHE$ (191)	RESUMO DO MX-BASIC CHR$ EX.:? CHR$ (191)
	MID$ EX.: F$=MID$(A$,3)PRINT MID$ (A$, 3,2)	MID$ EX.:F$=MID $ (A$,3): PRINT MID$ (A$,3,2)
	PLAY CARACTERÍSTICA : (...) TAMBÉM SUSTENIDO (≠ OU=) OU BEMOL (-)	PLAY CARACTERÍSTICA: (...) TAMBÉM SUSTENIDO (≠ OU +) OU BEMOL (-) .
	POINT EX: IF POINT (XY) THEN PRINT "ACESO" ELSE PRINT "APAGADO"	POINT EX: IF POINT (X,Y) < >0 THEN PRINT "ACESO" ELSE PRINT "APAGADO"

DYNACOM

Sempre um novo desafio

MX-1600

MANUAL DE OPERAÇÃO

DYNACOM