MSX

DOMINANDO O EXPERT

gradiente

DOMINANDO O EXPERT

gradiente
901-01940-000
06-2648-001

COLEÇÃO MSX

DOMINANDO O EXPERT

4ª EDIÇÃO
1986

gradiente

Editora Aleph

Este livro foi elaborado pela equipe técnica de redação e arte da Editora Aleph com exclusividade para a Gradiente Eletrônica S.A. a partir de originais técnicos fornecidos pela Srta. Denise Santoro Cruz em agosto de 1985. 4ª Edição revista e ampliada.

COORDENAÇÃO EDITORIAL: Pierluigi Piazzi
EDITORAÇÃO E REDAÇÃO FINAL: Nancy Mitie Ariga e Renato da Silva Oliveira
COPY-DESK: Regina Brito de Assumpção
PRODUÇÃO EDITORIAL: Betty F. Piazzi e Rosa K. Fromer
ILUSTRAÇÃO E ARTE: Ana Lúcia Antico, Marcelo da Rocha Ciambra, Mário Dimov Mastrotti,

CAPA: Cassiano Roda e Fátima Rossini

Editora Aleph

gradiente

ALEPH Publicações e Ass. Ped. Ltda.
Av. Brig. Faria Lima, 1451 - Conj. 31
01451 - São Paulo - SP -
Tel.: (011) 813-4555 - 212-4917

GRADIENTE ELETRÔNICA S.A.
R. Henrique Monteiro, 40
05423 - São Paulo - SP
Cx. Postal 30318
Tel.: (011) 814-2299

CIP-Brasil. Catalogação-na-Publicação
Câmara Brasileira do Livro, SP

Cruz, Denise Santoro.
C961d Dominando o Expert: como operar seu micro / Denise Santoro Cruz. — 4ª ed. São Paulo: Aleph, 1986.

(Coleção MSX)

1. BASIC (Linguagem de programação para computadores) 2. Expert (Computador) — Programação 3. Microcomputadores — Programação I. Título.

17. CDD-651.8
18. -001.642
18. -001.6424

85-1519

Índices para catálogo sistemático:

1. BASIC MSX: Linguagem de programação: Computadores: Processamento de Dados 651.8 (17.) 001.6424 (18.)
2. Expert: Computadores: Programação: Processamento de dados 651.8 (17.) 001.642 (18.)
3. Microcomputadores: Programação: Processamento de dados 651.8 (17.) 001.642 (18.)

SUMÁRIO

NOTA AO LEITOR

O EXPERT é um microcomputador extremamente versátil. Suas especificações técnicas estão de acordo com o padrão internacional MSX, adotado por mais de 2 dúzias de grandes fabricantes. Este livro contém as informações indispensáveis para sua utilização, abordando as principais instruções do possante BASIC-MSX.

Ao longo dos treze capítulos são apresentados, em linguagem fácil e acessível, comandos gerais e peculiaridades dessa versão do BASIC que fazem com que a operação do EXPERT esteja ao alcance do usuário que está tomando contato pela primeira vez com um microcomputador.

Para aqueles que já dominam com certa familiaridade o BASIC (especificamente, o M-BASIC) é aconselhável a consulta direta aos apêndices e ao livro LINGUAGEM BASIC MSX (da mesma editora), onde são apresentados, de forma resumida, todos os recursos e possibilidades do BASIC-MSX

Para os que estão tomando contato pela primeira vez com uma linguagem de programação, é indispensável a leitura atenta e paciente de cada linha de cada capítulo. Após uma primeira leitura, certamente o número de dúvidas terá aumentado ao invés de diminuir. Longe de caracterizar um mau aprendizado, isso indica apenas que o leitor estará mais consciente daquilo que ele deve saber e poderá, portanto, localizar e reler os trechos necessários à eliminação das dúvidas.

É extremamente mais proveitosa a leitura do texto diante do micro em funcionamento e, sempre que possível, essa situação deve ser procurada. A interação usuário-máquina é fundamental para a familiarização com a linguagem e conseqüente otimização de seu uso. Antes de ligar seu Expert, leia atentamente as páginas a seguir.

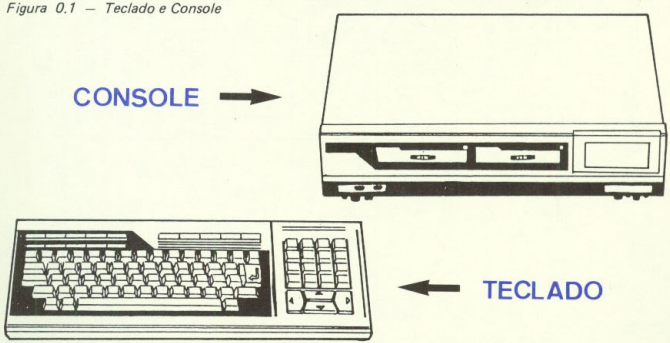
Antes de iniciar a leitura, verifique o sumário para ter uma idéia do que contém cada capítulo e cada apêndice.

Durante a leitura você vai se deparar com termos estranhos (normalmente em inglês), grafados em *itálico*. Muitos deles possuem tradução mas, mesmo assim, preferimos deixá-los em inglês, pois já fazem parte do jargão comum aos usuários de microcomputadores no Brasil. Você vai encontrá-los freqüentemente em livros e revistas especializadas. Em caso de dúvida, consulte o glossário no apêndice X.

COMO INSTALAR O EXPERT

Antes de instalar o micro você deve reconhecer cada componente e cada uma de suas partes. Vamos identificar a parte principal do seu sistema EXPERT: o teclado e o console.

Figura 0.1 — Teclado e Console



O teclado é o principal (e inicialmente, único) meio para que você dê ordens ao computador. Ele tem ligado a si um longo cabo que deverá ser ligado ao console. No console é que estão o "coração" e o "cérebro" do EXPERT e, se isso lhe ajudar, você pode considerar o teclado como sendo seus "ouvidos", seus "olhos", ou seu "tato". Levando essa analogia adiante, o vídeo (T.V. ou MONITOR) seria sua "boca".

Vamos reconhecer o console. Observe atentamente as figuras 0.2 e 0.3, acompanhando no texto a descrição de seus controles e encaixes.

Figura 0.2 — Painel Frontal do console

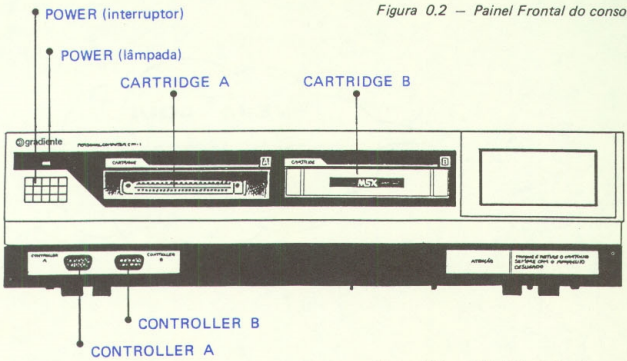
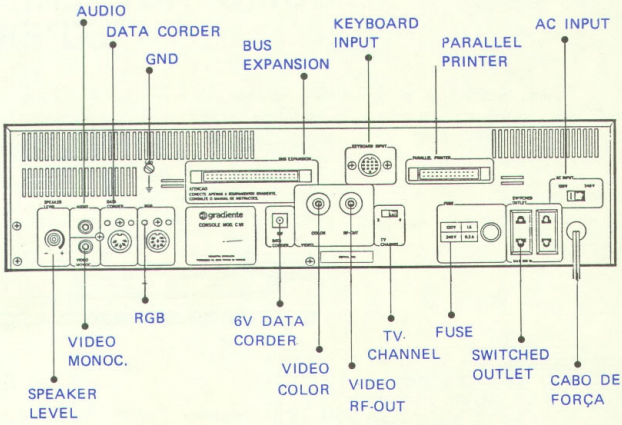


Figura 0.3 — Painel Posterior do console



PARTE FRONTAL

*	CARTRIDGE	Encaixes onde são conectados os cartuchos (memória pré-gravada, expansões ou interfaces para periféricos). Veja, mais tarde, nos apêndices III e IV maiores detalhes sobre como usar os cartuchos.
*	POWER (interruptor)	Interruptor de pressão para ligar e desligar o EXPERT.
*	POWER (lâmpada)	Indicador luminoso que acende quando o EXPERT está ligado.
*	CONTROLLER	Encaixe para JOYSTICKS (alavancas de contrôle manual).

PARTE POSTERIOR

*	SPEAKER LEVEL	Contrôle de volume de som do alto-falante interno do console.
*	AUDIO	Terminal de saída para audio (para ligação com amplificador).
*	VIDEO MONOC.	Terminal para ligação a monitor monocromático.
*	DATA CORDER	Terminal para ligação de gravador cassete.
*	RGB	Terminal para ligação ao monitor a cores (RGB)
*	GND	Encaixe para ligação de fio-terra.
*	BUS EXPANSION	Encaixe para expansões.
*	KEYBOARD INPUT	Encaixe para conexão do cabo do teclado.
*	PARALLEL PRINTER	Encaixe para conexão de uma impressora paralela.
*	FUSE	Encaixe para fusível.
*	SWITCHED OUTLET	Saída para ligação de cabo de força de outros equipamentos.
*	AC INPUT	Seletor de voltagem para rêde elétrica (VERIFIQUE QUAL A VOLTAGEM LOCAL!).
*	CABO DE FORÇA	Cabo para ligação à rêde elétrica. (NÃO LIGUE AÍNDA!)
*	6V DATA CORDER	Conector para alimentação de força do DATA CORDER.
*	VIDEO COLOR	Conector para a ligação de vídeos coloridos padrão PAL-M.
*	VIDEO RF-OUT	Conector para a ligação a TV (através de cabo de RF e comutador manual ANTENA EXTERNA/COMPUTADOR.
*	TV CHANNEL	Seletor de canal para ligação a TV.

CONEXÃO DOS CABOS

Inicialmente ligue o teclado ao console, conectando-o no KEYBOARD INPUT. A seguir, conecte o vídeo ao console. Se você tiver um monitor de vídeo monocromático, ligue-o no encaixe VIDEO MONOC.; se você tiver um monitor de vídeo a cores com entrada RGB, use o encaixe RGB; se você usar um monitor de vídeo a cores PAL-M, conecte-o no encaixe VIDEO COLOR; por fim, se você usar um receptor de TV (preto e branco ou colorido) use o encaixe VIDEO RF-OUT e não esqueça de selecionar o mesmo canal na TV e no TV CHANNEL (do console). Lembre-se, também, de posicionar o seletor manual COMPUTER/TV na posição COMPUTER.

Tanto os monitores quanto a TV podem ter seu cabo de força ligado numa das tomadas SWITCHED OUTLET. Se você quiser usar o Data Corder, conecte o cabo de alimentação ao terminal DC-6V (do Data Corder) e ao terminal DATA CORDER 6V (do console). Para ligar os cabos de gravação, leitura e controle do motor, veja o capítulo 5.

LIGANDO A MÁQUINA

Na ocasião da instalação, fique atento para as seguintes precauções:

* Verifique se a voltagem da rede é 110 ou 220 volts e posicione corretamente o seletor AC INPUT (fig.0.3).

* Evite usar o EXPERT em locais onde haja incidência de raios solares ou próximo a equipamentos que dissipem muito calor.

* Não o utilize e nem o guarde em locais onde haja poeira e/ou umidade em excesso.

* Evite batê-lo.

* Não coloque nada pesado sobre ele.

* Se ele for utilizado próximo a rádios, televisões ou outros equipamentos, poderá causar interferência (ruído) nos mesmos. Por outro lado, se você utilizar o micro onde houver campo magnético muito intenso, ele poderá ser danificado.

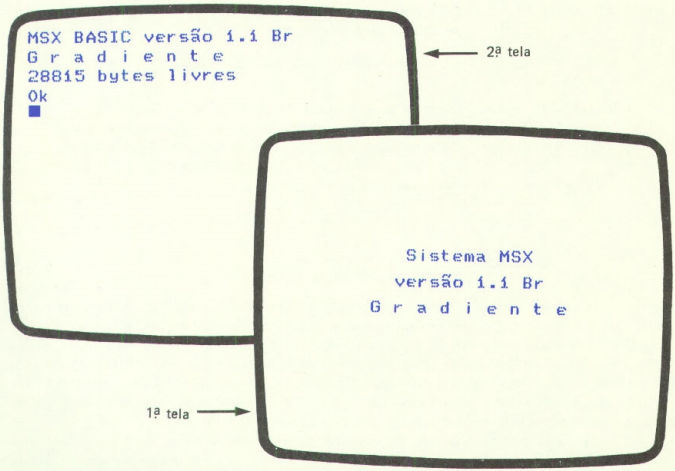
* Ligue o cabo de força do console à rêde elétrica.

* Confira se todas as etapas foram executadas.

* Finalmente, pressione o interruptor (POWER) no console, deixando o indicador luminoso aceso.

A tela deverá apresentar, seqüencialmente, os dois aspectos mostrados na figura 0.4.

Figura 0.4 — Telas iniciais do Expert



Assim, o EXPERT estará ligado e pronto para receber instruções.

Capítulo 1

COMO USAR O TECLADO

Uma das grandes vantagens do EXPERT sobre outros microcomputadores, está na versatilidade de seu teclado. À primeira vista ele se assemelha ao teclado de uma máquina de escrever comum (padrão QWERTY). Ele, porém, dispõe de diversos recursos adicionais que facilitam em muito a comunicação entre o usuário e o computador.

As peculiaridades do teclado do EXPERT são acessíveis pela digitação de algumas teclas especiais que chamaremos teclas de controle. Na figura 1.1 elas estão realçadas.

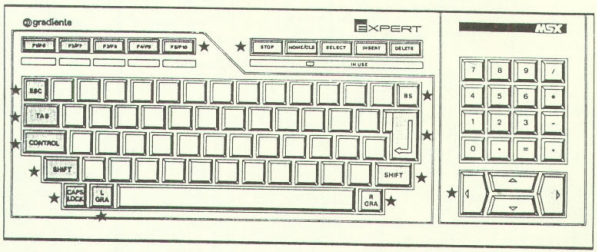


Figura 1.1 — TECLAS DE CONTROLE

As teclas de controle permitem que uma mesma tecla comum gere até seis diferentes caracteres (na verdade, as teclas das vogais podem gerar mais!).

Assim que o EXPERT é ligado, as teclas geram os caracteres mais usuais, isto é, letras minúsculas e números. A essa configuração, chamamos TECLADO NORMAL. Você pode observá-lo na figura 1.2.

Figura 1.2. — MULTITECLADO DO EXPERT — Para que os caracteres gráficos dos teclados especiais apareçam por inteiro no vídeo, é necessário que se esteja trabalhando com a SCREEN 1. Na SCREEN 0 eles aparecem "cortados" do lado direito.

USANDO UMA TECLA

Vamos, agora, explorar os recursos do teclado usando três teclas comuns como exemplo. Antes disso, porém, precisamos convencionar algumas regras relativas à notação utilizada.

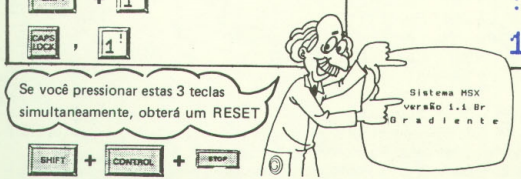
Sempre que escrevermos: A + B — significa: pressionar a tecla A e, enquanto ela é mantida para baixo, pressionar a tecla B.

Sempre que escrevermos: A , B — significa: pressionar a tecla A e, depois de soltá-la, pressionar a tecla B.

Com isso, estamos em condição de usar as tabelas a seguir. Observe-as atentamente.

Tabela 1.1 — Como usar as combinações de teclas

PRESSIONE AS TECLAS ABAIXO	E OBTENHA ESTE RESULTADO
A	a
~ ^	ã
~ ^ , A	ã
SHIFT + ~ ^	Ã
SHIFT + ~ ^ , A	ã
SHIFT + ~ ^ , SHIFT + A	Ã
L GRA + ~ ^	♠
SHIFT + L GRA + ~ ^	♦
R GRA + ~ ^	ũ
SHIFT + R GRA + ~ ^	Ũ
CAPS LOCK , A	A
SHIFT + 1 !	!
CAPS LOCK , 1 !	1



Capítulo 2

COMEÇANDO A PROGRAMAR

O QUE É PROGRAMAR?

Programar é o ato de escrever para o computador uma sequência de instruções e informações, na ordem em que ele deve executá-las, para que determinada tarefa possa ser cumprida.

Os programas são diferentes, uns dos outros. As informações utilizadas para manipular uma conta corrente, por exemplo, não são necessariamente as mesmas que controlam um jogo tipo "video-game".

Programas escritos em uma determinada linguagem não contêm as mesmas instruções e a mesma estrutura que outros programas escritos em outras linguagens.

Existem duas maneiras diferentes de se usar uma instrução BASIC: na forma de programa ou no modo "imediato".

★ Um programa em BASIC é um conjunto de instruções escritas uma após a outra, em linhas numeradas.
Observe o programa a seguir:

```
10 PRINT "O BASIC"
20 PRINT "PERMITE"
30 PRINT "A CONVERSA"
40 PRINT "ENTRE O HOMEM"
50 PRINT "E O COMPUTADOR"
```

Este é um exemplo de um programa!

Os números das linhas geralmente estão em intervalos de dez, para facilitar as eventuais correções com acréscimo de novas linhas intermediárias.

★ A segunda maneira é o chamado modo "imediato" porque, se a tecla RETURN for pressionada, o computador irá executá-la imediatamente. No modo "imediato", uma linha de instruções não é precedida por um número.

Para ter um controle maior sobre seu computador, você deverá aprender a se comunicar com ele, através de instruções, na linguagem que ele entende.

O EXPERT, como a maioria dos computadores pessoais, entende a linguagem BASIC (Beginner's All Purpose Symbolic Instruction Code).

Na realidade, o computador faz muitos esforços para transformar as instruções BASIC em suas próprias operações elementares. Tal linguagem, residente no EXPERT, é formada por uma série de palavras (de origem inglesa), que facilitam sua memorização.

ENTRANDO NO BASIC

Ligue o computador (lembre-se, o interruptor POWER está localizado no lado esquerdo do painel do console).

Você verá as telas como mostram as figuras 2.1 e 2.2.

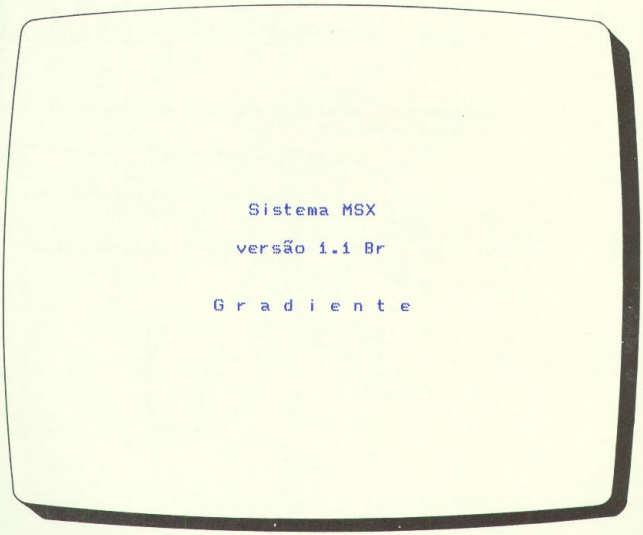


Figura 2.1

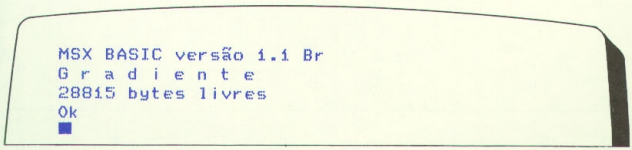


Figura 2.2

COMEÇANDO A PROGRAMAR

A palavra OK é a mensagem do computador dizendo que está pronto para aceitar seus comandos.

O quadro branco, abaixo da mensagem OK, é chamado CURSOR. Sua posição na tela indica o localizador da próxima letra, mensagem ou instrução a ser escrita.

Vamos começar enviando e recebendo informações através do EXPERT, para que você possa se familiarizar com suas características.

Comece pressionando SHIFT + HOME/CLS, para limpar a tela e, logo após, reproduza a linha mostrada na tela da figura 2.3.

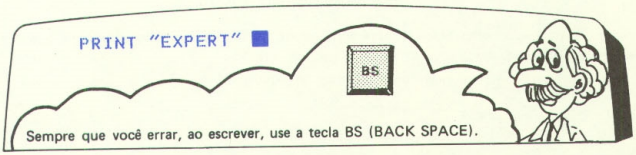
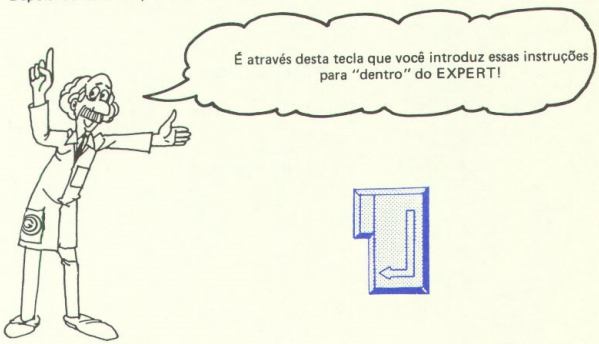


Figura 2.3

O cursor deve se mover uma posição para a direita, cada vez que você pressionar uma tecla. Depois de terminar, tecle RETURN.



Sua tela ficará como mostra a figura 2.4.

Figura 2.4

Quando você teclar RETURN, estará ordenando ao computador que execute o trabalho. Assim, você obterá o que foi especificado no programa.

Quando ele terminar de obedecer às suas instruções, a mensagem OK surgirá na tela, informando que outras instruções podem ser executadas.

Note que, se você escrever de forma incorreta uma palavra correspondente a um comando BASIC, tal como PRUNT ao invés de PRINT, a seguinte mensagem aparecerá na tela:

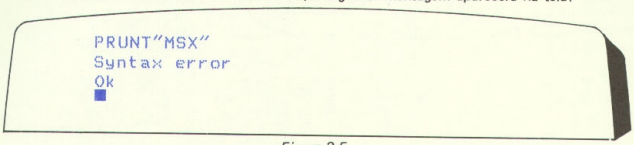


Figura 2.5

SYNTAX ERROR significa que você escreveu uma instrução de forma incorreta. Porém fique descansado. O computador é muito tolerante e paciente!

UMA OPÇÃO PARA O PRINT

O comando PRINT é utilizado para expor mensagens na tela, tanto numéricas e alfanuméricas, como alfabéticas. As mensagens alfabéticas devem estar sempre escritas entre aspas (" ").

Existe um caractere que funciona exatamente da mesma forma, como se fosse uma abreviação do comando PRINT: o ponto de interrogação [?].

Digite o que mostra a figura 2.6.

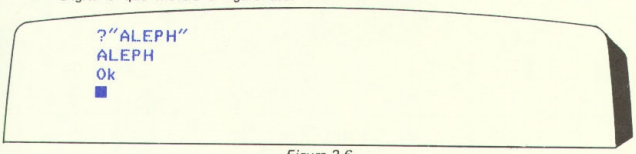


Figura 2.6

EXPRESSÕES

O termo PRINT também pode ser usado para exibir respostas de expressões matemáticas. Por exemplo, limpe sua tela e introduza os exemplos apresentados na figura 2.7

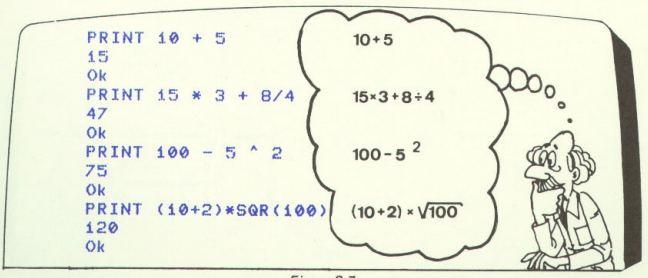
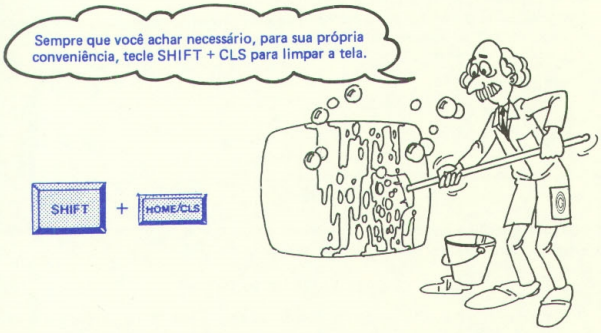


Figura 2.7

Note que após cada expressão, você deve digitar RETURN para, logo em seguida, obter uma resposta. A mensagem OK significa que você poderá continuar a digitar os exemplos.



Lembre-se: você também pode escrever suas expressões de outra maneira, obtendo o mesmo resultado:

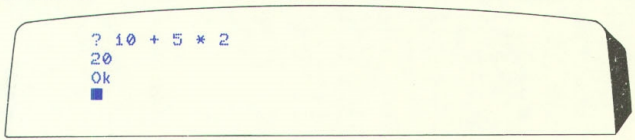


Figura 2.8

CORRIGINDO ERROS

Vamos aproveitar o exemplo anterior para aprender como corrigir erros rapidamente.
No lugar da expressão anterior vamos escrever 10+6 * 2.
Isto significa que iremos trocar o número 5 pelo 6.
Vejamos como fazer isto.
Mova o cursor até que ele fique sobre o número 5(fig. 2.9):

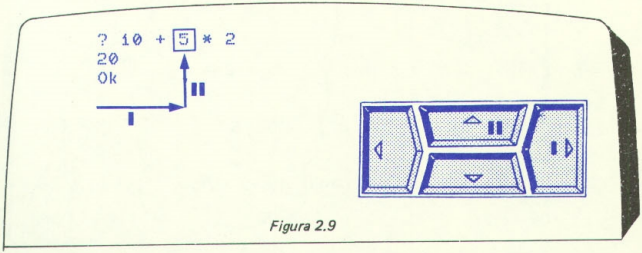


Figura 2.9

Com o cursor assim posicionado, tecle o número 6 e o RETURN. Logo a seguir você obterá a nova resposta (Fig. 2.10)

```
? 10 + 6 * 2
22
Ok
```

Figura 2.10

OUTRA FORMA DE EXIBIR NA TELA

Vamos usar mais um exemplo para esta explicação. Digite as seguintes linhas:

```
10 PRINT"E"
20 PRINT"X"
30 PRINT"P"
40 PRINT"E"
50 PRINT"R"
60 PRINT"T"
```

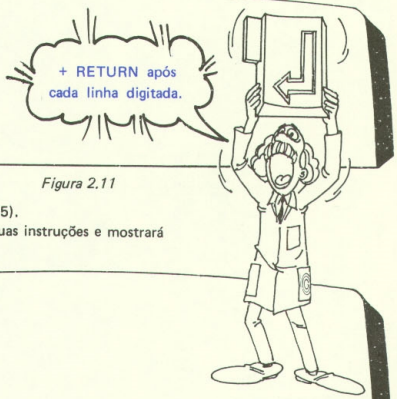


Figura 2.11

Agora, digite RUN (ou F5).

O computador executará suas instruções e mostrará o resultado. Veja a figura 2.12.

```
10 PRINT"E"
20 PRINT"X"
30 PRINT"P"
40 PRINT"E"
50 PRINT"R"
60 PRINT"T"
run
E
X
P
E
R
T
Ok
```

Figura 2.12

Se você quiser apagar ou eliminar alguma linha, existem duas maneiras: uma é digitar o número da linha e, em seguida, teclar RETURN. A outra, é usar o comando DELETE, como por exemplo:

DELETE 20-40 apaga da linha 20 até linha 40.

DELETE 20	apaga somente a linna 20.
DELETE -40	apaga desde a primeira linha do programa, até a linha 40, inclusive.

Se você quiser acrescentar uma linha qualquer, por exemplo:

```
25 PRINT"EXPERT"
```

Basta digitá-la e depois teclar RETURN. Ela será incluída entre a linha 20 e 30. Comande LIST e verifique você mesmo!!!

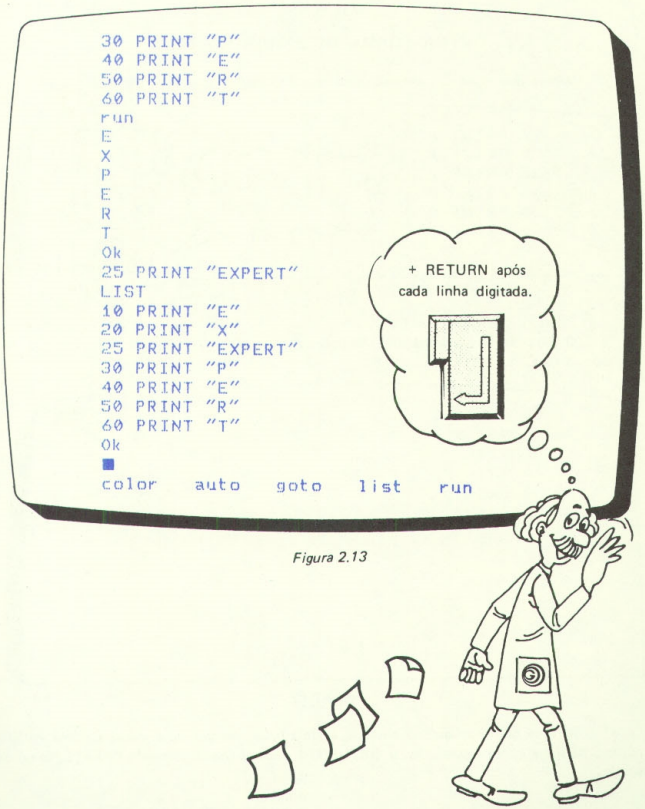
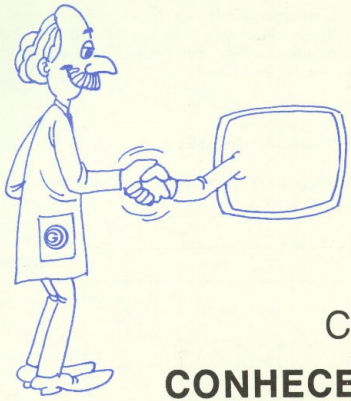


Figura 2.13



Capítulo 3

CONHECENDO A TELA

Vamos, inicialmente, entender como a tela do EXPERT funciona. A partir disso, aprenderemos a controlá-lo através de comandos BASIC.

Uma maneira fácil de compreender o funcionamento da tela, é associá-la a um tabuleiro de xadrez.

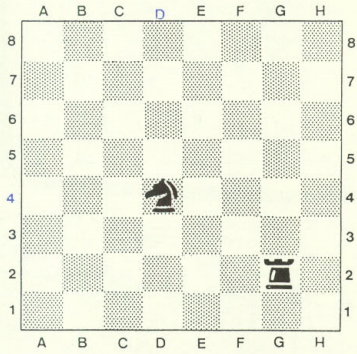


Figura 3.1

CONHECENDO A TELA

A cada posição do tabuleiro, podemos associar um "nome" formado por uma letra e um número. Assim, localizamos um cavalo na posição d4 e uma torre na posição g2. Cada posição tem um nome único e diferente. Podemos dizer que cada nome é formado pelas coordenadas de cada posição. A letra é uma coordenada e o número é outro.

COORDENADAS NA TELA

Para podermos desenhar na tela do EXPERT, temos que usar os "nomes" de cada um dos seus quadradinhos.

Veja como isso é fácil:

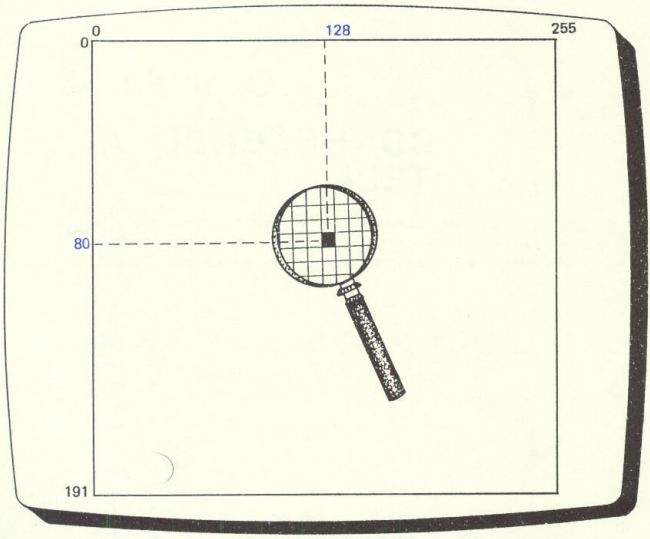
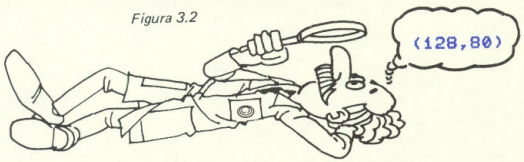


Figura 3.2



Cada quadradinho tem um nome composto por dois números: o número da coluna e o número da linha. Estes dois números juntos e em ordem, formam as COORDENADAS DE CADA POSIÇÃO na tela.

Conhecer os nomes dos quadradinhos é fundamental para a criação de desenhos.

Na figura a seguir, alguns quadradinhos estão escurecidos. Você sabe o nome destes quadradinhos? Escreva-os num pedaço de papel e depois confira com as respostas.

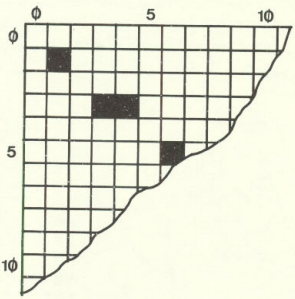
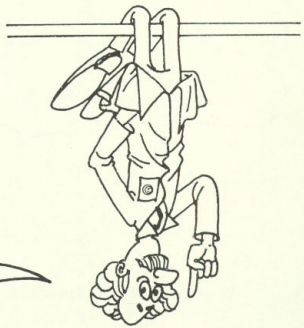


Figura 3.3

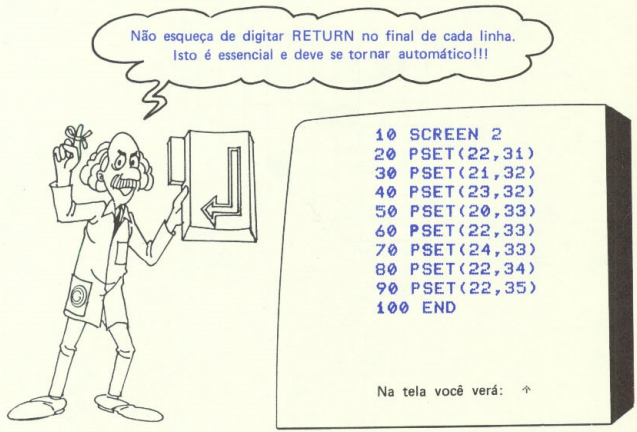
EXPERT



(1,1) (3,3)
(4,3) (6,5)

CONHECENDO A TELA

Agora nós iremos usar as coordenadas em um programa que vai marcá-las na tela. Digite, através do teclado, o programa a seguir.



Verifique se o seu programa foi digitado corretamente. Se você errou algo, reveja no apêndice II como efetuar correções usando INSERT DELETE etc...
Agora você pode teclar RUN.

Não se apavore. Este tipo de problema é fácil de resolver!! Faça o seguinte:

1. Comande:
   LIST

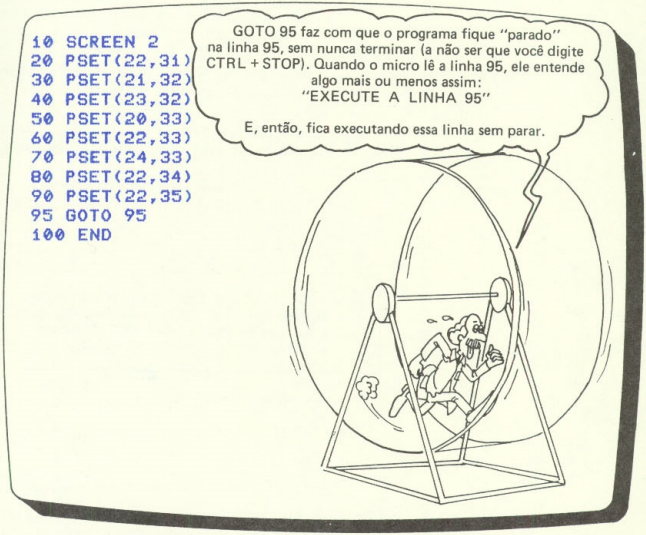
Como o próprio nome diz, "LIST" deve listar seu programa inteiro.

2. Agora, digite esta nova linha:
   95 GOTO 95
3. Comande LIST novamente e verifique se a linha 95 foi introduzida entre as as linhas 90 e 100.
4. Por fim, digite RUN.

Agora o computador deixou os pontos na tela e nós não podemos mais escrever sobre ela. Tente digitar: EXPERT.

Por que não apareceu nada na tela???

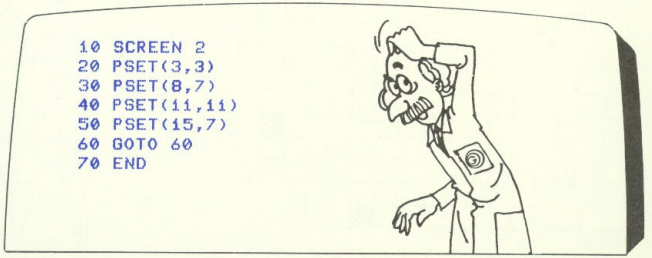
Porque a máquina está executando o programa, e continuará assim até que você pressione as teclas CTRL + STOP. Vamos olhar a listagem do programa novamente:



Antes de adicionarmos a linha 95, os quadradinhos apareciam momentaneamente e depois desapareciam.

Com a linha 95, os pontos não desaparecem e o responsável por isso é o comando GOTO.

Digite NEW (veremos para que isso serve mais adiante) e introduza o programa a seguir. Depois, tente compreendê-lo.



CONHECENDO A TELA

A linha 10 diz ao computador que você quer usar a SCREEN 2.

Existem 4 tipos de telas no EXPERT. A escolha de cada uma delas é feita através de um comando SCREEN.

SCREEN Ø	–	Esta tela permite que você introduza 24 linhas de texto com, no máximo, 40 caracteres cada. Esta é a tela que o computador apresenta logo após ser ligado.
SCREEN 1	–	Esta tela permite introduzir 24 linhas de texto, com 32 caracteres em cada linha.
SCREEN 2	–	É a tela para desenhar em Alta Resolução Gráfica, isto é, com linhas finas e muitos detalhes.
SCREEN 3	–	É a tela para desenhar em Baixa Resolução Gráfica, com linhas mais grossas e poucos detalhes.

Sempre que um programa usar SCREEN 2 ou SCREEN 3 e for digitado CTRL + STOP, a execução é interrompida e, automaticamente, a SCREEN Ø é acionada. O cursor e o "OK" surgem no vídeo.

A linha 20 introduz a instrução PSET. Ela manda o computador marcar o quadradinho de coordenadas (3,3) . Lembre-se: o primeiro número entre parênteses refere-se à coluna, e o segundo número refere-se à linha onde está localizado o quadradinho.

As linhas de 30 até 50 continuam marcando os quadradinhos, como mostra a figura 3.4.

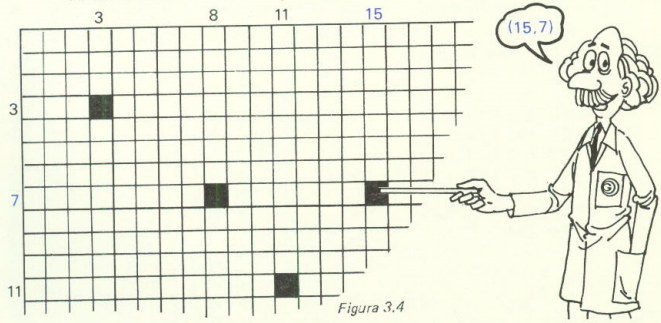
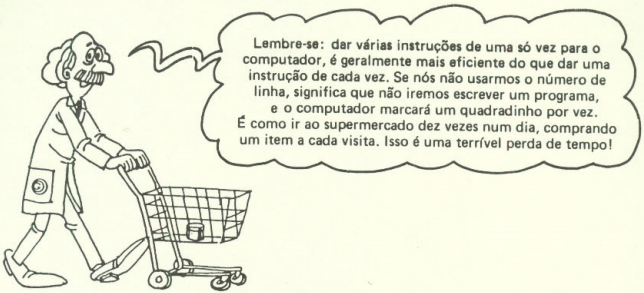


Figura 3.4

Agora tente criar mais alguns pontos.



Depois de toda esta conversa, é hora de colocar a mão na massa e praticar tudo o que aprendemos!!!

Digite NEW e, a seguir, RETURN.

O comando NEW apaga qualquer programa que foi escrito, ou seja, limpa sua área de trabalho na memória do computador. Agora tente escrever um programa que desenhe o pedaço de tabuleiro da figura 3.5. Dica: use a SCREEN 3 e faça o PSET pular de 4 em 4!

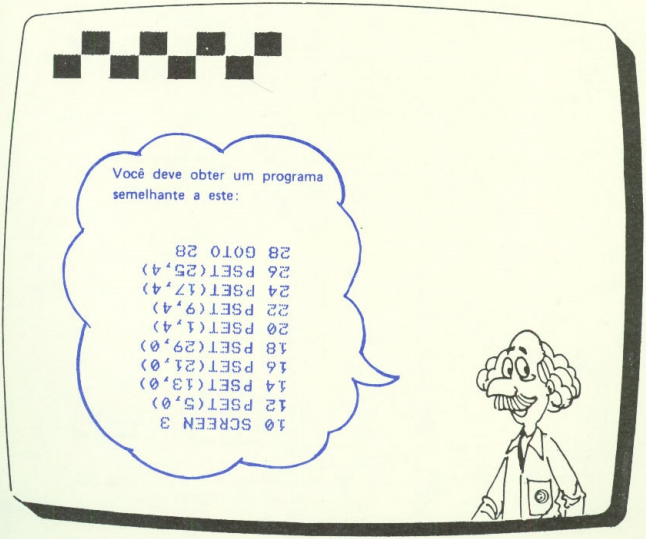


Figura 3.5

ADICIONANDO CORES

Cada um dos quadrados que você desenhou pode ser colorido em 16 diferentes cores. Cada cor que o EXPERT usa é acessada através de um número: seu CÓDIGO DE COR.

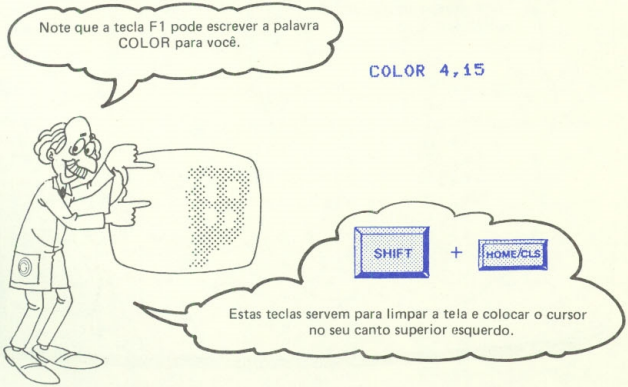
Tabela 3.1 (Veja a 4ª capa)

CÓDIGO DE COR	NOME DA COR
0	INCOLOR
1	PRETO
2	VERDE
3	VERDE CLARO
4	AZUL ESCURO
5	AZUL CLARO
6	VERMELHO ESCURO
7	CIANO (AZUL ANIL)
8	VERMELHO CLARO
9	ROSA
10	OURO
11	AMARELO
12	VERDE MUSGO
13	MAGENTA (ROXO)
14	CINZA
15	BRANCO

Não tenha pressa.

Antes de aprender como mudar as cores dos desenhos, nós vamos ver como algumas delas surgem no vídeo.

Pressione as teclas SHIFT + HOME/CLS para limpar a tela e digite o seguinte:



Depois de pressionar RETURN, as cores do texto e do fundo da tela são definidas de acordo com os respectivos códigos. Vamos ver um outro exemplo:

```
COLOR 4,4
```

O texto desaparece, e o fundo fica azul. Agora escreva seu nome. Você ouvirá o som das teclas, mas não verá os caracteres na tela. Isto acontece porque o primeiro código, logo após o comando COLOR, é o espaço que o computador usa para determinar a cor das letras do texto, e o segundo código indica a cor usada para o fundo da tela. Ambos são o mesmo: 4. É como escrever com tinta azul num papel também azul. Além desses dois números, existe ainda um terceiro, que pode ser usado para indicar a cor do contorno da tela (a borda). Nós veremos isto mais para a frente.

Para tornar o texto visível, é necessário escrever um novo comando COLOR.

Como podemos fazer isso sem ver sequer o cursor???

Para retornar a cor das letras, basta digitar SHIFT + F1 (isso equivale a F6). Antes disso, porém, temos que ter certeza que o cursor está posicionado numa nova linha. Conseguimos isso digitando RETURN duas ou três vezes.

Agora sim, pressione SHIFT + F1. A tela voltará ao normal e você verá novamente todas as letras na tela. Dê uma olhada no último comando COLOR:

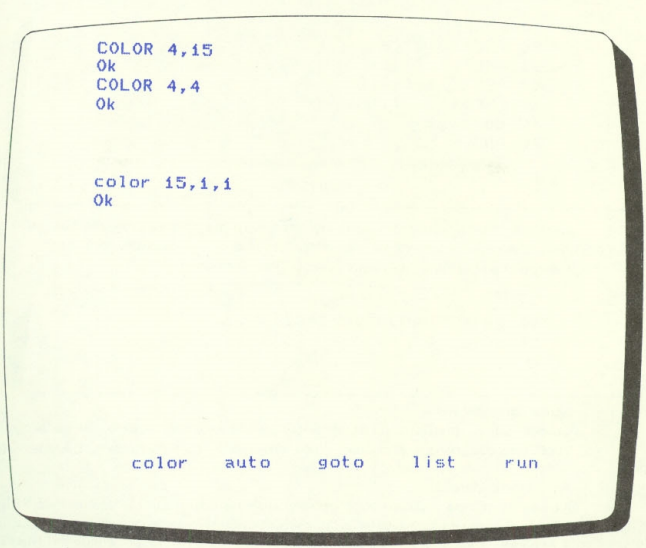


Figura 3.6

Você verá que há um terceiro número no fim do comando COLOR, como já comentamos acima.

Veja o formato deste comando:

COLOR (cor do texto), (cor do fundo), (cor da borda).

A borda, usualmente, não é exposta no modo texto (SCREEN Ø). Isto porque só é necessário especificar uma cor para a mesma, quando você usa SCREEN 1 ou SCREEN 2.

CONHECENDO A TELA

Agora experimente com outros códigos para familiarizar-se com todas as cores.

Da mesma forma que podemos mudar a cor das letras, podemos colorir os pontos marcados pela instrução PSET. Basta acrescentar uma vírgula e o código da cor, após as coordenadas do ponto. Observe os exemplos a seguir:

```
10 SCREEN 3
20 PSET(3,3),10
30 PSET(7,7),10
40 PSET(11,11),10
50 PSET(15,7),10
60 GOTO 60
70 END
```

Figura 3.7

Agora, mude as cores dos pontos . . .

```
10 SCREEN 3
20 PSET(3,3),6
30 PSET(7,7),11
40 PSET(11,11),2
50 PSET(15,7),13
60 GOTO 60
70 END
```

Figura 3.8

Antes de finalizar este capítulo, digite este pequeno e simples programa que usa o comando GOTO de uma maneira fácil de ser entendida. Nós já o utilizamos antes, lembra-se?

Primeiro digite NEW e RETURN; depois, introduza estas linhas:

```
10 PRINT"DANDO VOLTAS...",,,
20 GOTO 10
```

Agora comande RUN.

Quando quiser, pressione CTRL + STOP para interromper a execução do programa.

Você entendeu porque a máquina ficou escrevendo repetidas vezes "DANDO VOLTAS . . ."??

Aqui está o porquê:

A linha 10 manda o computador imprimir uma mensagem através do comando PRINT. A linha 20, manda o computador executar a linha 10 novamente e assim por diante, até que você interrompa a execução. Não se preocupe com as três vírgulas na linha 10 por enquanto! Nós as entenderemos mais adiante.

Capítulo 4

ÀS VOLTAS COM LAÇOS

Depois de usar o comando PSET no capítulo anterior, você provavelmente está pensando se há uma forma mais rápida de mandar o computador marcar os quadradinhos, ao invés de escrever cada uma de suas coordenadas.

Bem... de fato, há!

Antes de aprender como, digite mais este programa.

```
10 REM PSETANDO
20 CLS
30 PRINT"ESCREVA UM NUMERO ENTRE 10 E 24
0:"
40 INPUT X
50 SCREEN 3
60 PSET(X,10)
70 PSET(X,30)
80 PSET(X,80)
90 PSET(X,45)
100 GOTO 100
```

Agora, comande RUN.

Seu programa desaparecerá e a mensagem "ESCREVA UM NÚMERO ENTRE 10 e 240" aparecerá na tela (previamente limpa pelo comando CLS na linha 20).

A mensagem é produzida no vídeo através do comando PRINT na linha 30, e o cursor aparecerá depois do ponto de interrogação.

O ponto de interrogação foi colocado pelo comando INPUT. Esse comando ordena ao computador que interrompa a execução e aguarde até que alguma coisa seja introduzida como uma resposta.

Só então as instruções seguintes serão executadas. A resposta digitada pode ser um nú-

mero ou uma seqüência qualquer de caracteres.

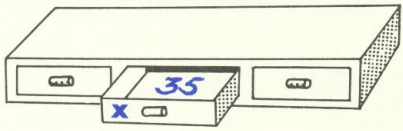
Não digite nada ainda!

Vamos primeiro entender o que acontecerá quando a resposta for digitada.

VARIÁVEIS

As variáveis que vamos estudar nesta etapa do nosso livro são as usadas somente para armazenar valores numéricos.

Podemos entender por *variável* um recipiente armazenador de informações; um lugarzinho da memória onde são guardados alguns dados.



Vamos ver como o EXPERT usa as variáveis.

Responda à interrogação na tela escrevendo um número seguido por RETURN.

Ao introduzir o número, os quadradinhos são automaticamente colocados na tela, pois a localização da qual cada um deles depende para ser desenhado, foi dada pelo número digitado.

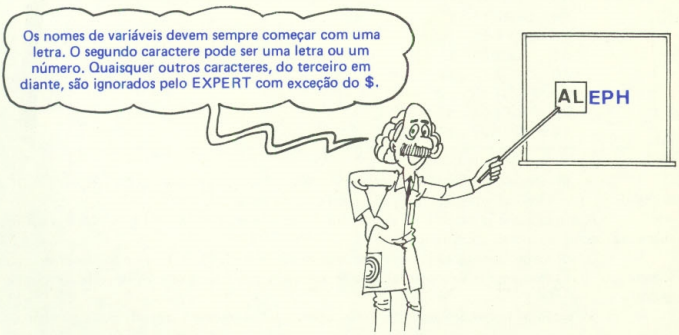
Como isso foi possível?

Imagine que a memória do computador seja composta por diversas gavetas e que uma delas está rotulada com um X. Dentro dessa gaveta você guardou uma ficha contendo o número que foi digitado. Quando o computador chega na linha 60, onde a instrução ordena que seja desenhado um quadradinho cujas coordenadas são (X,10), ele já sabe que X é uma das gavetas da memória e que nela está a informação que ele precisa. Em seguida, ele olha toda a memória, até achar uma gaveta chamada X. Feito isso, o valor encontrado é utilizado, e ele termina de executar a instrução.

Fica difícil armazenar e recuperar informações se não soubermos como usar essas informações. Desse modo, nós escolhemos o termo variável para designar uma localização especial na memória do EXPERT destinada a armazenar informações.

A letra X é somente um dos muitos símbolos que podem ser usados para dar nome a uma variável.

Se nós alimentarmos o computador com informações que não estão corretamente endereçadas, teremos dificuldades para achar as informações quando, mais tarde, precisarmos delas.



O EXPERT, como muitos outros computadores pessoais, usa dois diferentes tipos de variáveis. Um deles armazena somente números, e o outro armazena quaisquer seqüências de caracteres.

Qualquer variável, cujo nome comece com uma letra de A a Z, armazena somente números. Mas quando o símbolo $ é colocado após as letras, a variável pode armazenar tanto números quanto palavras.

O nome de uma variável pode ter mais de 2 caracteres e não pode ser uma palavra da linguagem BASIC. Por exemplo: S, VC, E1, EXT, etc., serão aceitos como nomes para as variáveis. LET, PRINT, INPUT, etc., não serão!

Experimente digitar:

```
10 CLS
20 PRINT "Aquilo que voce colocar apos o
 ponto de interrogacao sera' reproduzido
 na tela."
30 INPUT T$
40 PRINT T$
50 END
```

Agora, digite RUN e escreva qualquer número em resposta ao símbolo ? da tela. Observe o computador repetir o número escrito.

Comande RUN novamente e escreva várias palavras, de modo a totalizar mais de 255 caracteres.

O que acontece???

Você pode armazenar no máximo 255 caracteres em qualquer variável. Igualmente, você pode escrever 255 caracteres em uma linha de seu programa BASIC.

Experimente ver o que acontece quando você tenta escrever mais de 255 caracteres.

Isso mesmo!!!

Os caracteres extras são ignorados!!!

TRABALHANDO COM VARIÁVEIS

Há vários comandos que permitem ao computador colocar informações em uma variável. Nós já vimos que o INPUT é um deles. Aqui está outra maneira:

```
10 CLS
20 LET E=1541
30 LET A=3
40 LET K=E+A
50 PRINT K
60 END
```

Agora, rode (com RUN) o programa. O número 1544 deve ter aparecido na tela.

Este programa ilustra como transferir uma informação para as variáveis.

O comando LET *batiza* a variável igualando-a a um valor. A informação que você deseja colocar na variável é escrita logo após o sinal de igual.

No programa acima, foram determinados os valores 1541 e 3 para E e A, respectivamente. O computador adicionará os dois números, e colocará o resultado na variável K. Na linha 50, o conteúdo desta variável é impresso na tela.

Você não precisa usar, necessariamente, o comando LET para transferir uma informação para uma variável. O programa abaixo mostra uma forma opcional de fazer isso.

```
10 CLS
20 E=10
30 A=20
40 K=E+A
50 PRINT K
60 END
```

ÀS VOLTAS COM LAÇOS

O computador irá entender da mesma maneira. Isto significa que LET somente facilita a estrutura do programa, ou seja, quando alguém o ler ou quando, depois de muito tempo, você tornar a revê-lo.

Agora nós iremos aprender algumas outras maneiras de usar uma variável . . .

Digite e execute as linhas a seguir:

```
10 CLS
20 FOR F=1 TO 10
30 PRINT F
40 NEXT F
50 END
```

Surpreso com o resultado???

Este programa usa o mesmo tipo de laço que nós já discutimos.

Depois da linha 10 limpar a tela, a linha 20 instrui o computador a colocar todos os números entre 1 e 10 na variável F, um a um.

O obediente computador começa colocando o 1. A instrução PRINT F na linha 30, faz com que o computador olhe para o número armazenado na variável F e o mostre na tela.

Depois, na linha 40, o micro é instruído para pegar o próximo número após 1, e colocá-lo na variável F. Visto que uma variável só pode conter um item de cada vez, o 1 é apagado da localização e o 2 é colocado em seu lugar. Depois, o computador segue outra vez à linha 20 e recomeça o processo.

Sendo que o número 2 é um dos números designados para ser colocado em F, ele será impresso no vídeo.

Na linha 40 o número 2 é apagado, sendo colocado o número 3 em seu lugar, e o computador retorna para a linha 20, onde tudo se repete.

Este laço continua até que o número 10 seja colocado em F e depois exposto na tela. Desde que o número 11 não foi designado pela linha 20, o computador lê o comando END na linha 50 e pára.

O comando FOR . . . NEXT é sempre um par. Nunca use um, sem o outro.

O laço criado por este comando é fundamentalmente diferente do laço criado pelo comando GOTO. Com prática, você aprenderá quando o uso de cada um é apropriado.

Será que você consegue fazer um programa que utilize os números entre 20 e 235 para desenhar uma linha??? Antes de ver de que maneira isso pode ser feito, experimente tentar sozinho.

Sua resposta pode coincidir com esta:

```
10 SCREEN 3
20 FOR F=20 TO 235
30 PSET(F,40)
40 NEXT F
50 GOTO 50
```

Comande RUN e observe como o computador desenha a linha sem muito esforço.

O computador coloca todos os números entre 20 e 235 na variável F, um a um.

Na linha 20, o valor 20 é colocado em F. Quando ele chega na linha 30, o quadradinho de coordenadas (F,40), passa a ser (20,40), pois o computador checa qual valor está contido em F, e o substitui.

O comando NEXT F, na linha 40, pede o próximo número, e volta para a linha 20. Visto que o número seguinte é 21, o computador marcará o quadradinho (21,40).

Este laço continua, até que todos os quadradinhos tenham sido marcados na linha 40 da tela.

Agora tente mudar o laço do programa, que desenha uma linha horizontal, para que ele desenhe uma linha vertical na coluna 20. Seu programa deve ficar semelhante a este:

```
10 CLS
20 SCREEN 3
30 FOR Y=20 TO 135
40 PSET(20,Y)
50 NEXT Y
60 GOTO 60
```

Note que, desta vez, nós usamos um outro nome para a variável (Y), a qual conterá todos os números entre 20 e 135. Este laço desenhará a linha vertical.

Agora, digite o seguinte:

```
10 CLS
20 PRINT"ISTO E'REPETITIVO."
30 GOTO 20
40 END
```

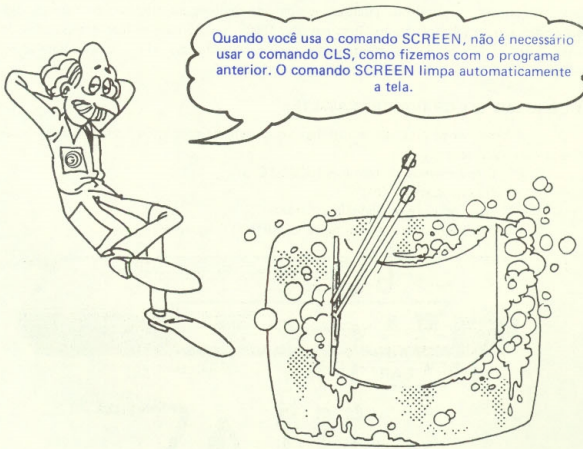
O comando END não é, necessariamente, o fim do programa. Note o modo estruturado acima. O computador nunca vai além da linha 30. Então, ele só vai interromper o laço, se você apertar as teclas CTRL + STOP.

Isso quer dizer que você não precisa colocar sempre o comando END no final de cada programa. Embora ele seja prático, você aprenderá onde e quando o END é necessário.

Semelhante a isso, você não precisa começar o seu programa sempre com:

10 CLS

Você decidirá quando tem que trabalhar com uma tela limpa ou não.



Capítulo 5

ARMAZENANDO EM CASSETE

Qualquer programa, contido na memória do EXPERT, é apagado quando o micro é desligado. Portanto, se quisermos guardar nossos programas para que não precisemos digitá-los novamente, devemos armazená-los em fitas cassete. Desse modo, sempre que precisarmos de algum programa, basta carregá-lo da fita diretamente para a memória do micro. Vamos aprender como gravar na fita e, futuramente, como carregar na memória.

1) CONEXÃO DO GRAVADOR CASSETE

Inicialmente precisamos interligar o cassete ao micro. Será mais fácil se o cassete possuir os seguintes recursos:

* Esteja munido de terminal REMOTE (a).
* Possua contador (b).
* Possa gravar e reproduzir em mono.
* Possa ser acionado com terminal REMOTE ligado.

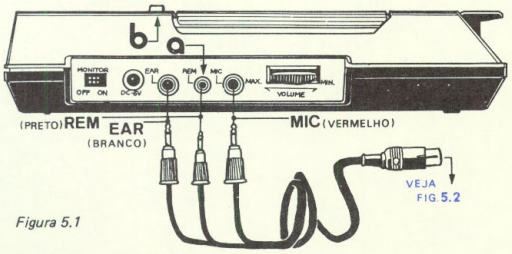


Figura 5.1

Desligue o micro antes de conectar o cassete.

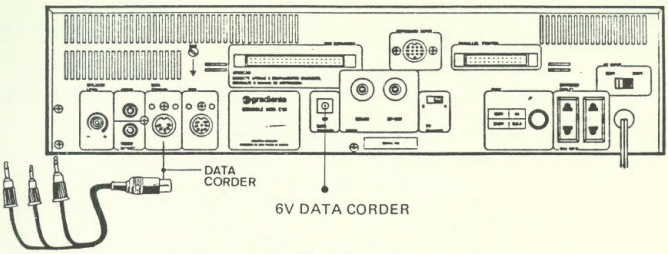


Figura 5.2

Se o seu cassete não possui terminal com *plug* REMOTE (cor preta), não se preocupe. Isso não é tão importante!

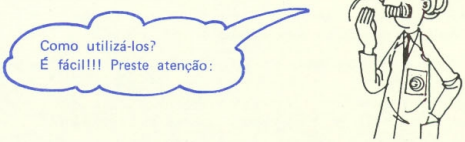
O terminal com o *plug* vermelho é usado para gravar (MIC) e o terminal com o *plug* branco serve para carregar (EAR).

2) ETAPAS DE PREPARAÇÃO DO CASSETE

- Deixar o volume em um tom médio, pois se estiver baixo, qualquer ruído interferirá com facilidade e, se estiver muito alto, haverá distorção.
- Para cassetes que possuam TONE, estes devem ser usados na posição mais aguda, ou seja, no máximo.
- É aconselhável usar o mesmo cassete para LOAD e SAVE, pois a velocidade da fita poderá não ser igual para ambos os cassetes.
- Entre os vários tipos de cassete, existem alguns que, quando em PLAY, captam ruídos do microfone. Neste caso deve-se desconectar o terminal MIC.

3) GRAVANDO / CARREGANDO

Para gravar e/ou carregar seus programas existem dois comandos muito úteis:
CSAVE (Cassete SAVE) – para gravar
CLOAD (Cassete LOAD) – para carregar



Vamos supor que se queira gravar um programa chamado PROG.1. Então deve-se escrever, primeiramente, o comando que grava programas em fita cassete. Depois, entre aspas, o nome do programa. Assim:

```
CSAVE"PROG.1"
```

Antes de pressionar RETURN, você deve ligar o gravador.

ARMAZENANDO EM CASSETE

Após algum tempo, surgirá uma mensagem OK na tela, significando que tudo correu bem e que seu programa está gravado.

Para carregar um programa qualquer, você deverá fazer o mesmo, só que o comando se apresentará da seguinte maneira:

```
CLOAD"PROG.1"
```

Primeiramente, você deverá rebobinar a fita. Então, pressione RETURN e ligue o PLAY do gravador.

Depois de alguns segundos o micro dará um aviso através da tela.

```
10 PRINT"PROGRAMA EXEMPLO"
20 PRINT"VOU SER GRAVADO EM FITA"
CSAVE"PROG.1"
Ok
CLOAD"PROG.1"
Found:PROG.1
Ok
```

significa que foi encontrado, na fita, um programa chamado PROG.1.

significa que o programa já está na memória, pronto para ser usado.

A gravação do programa na fita cassete é feita pelo computador, que o transforma em sinais elétricos. Portanto, tecnicamente, é o mesmo que gravar música. A diferença é que o programa é mais sensível à variação da rotação e do nível de gravação.

A maioria dos problemas com gravações de programas em fitas tem suas causas em uns poucos pontos chaves.

Verifique sempre os seguintes ítens:

- ★ O tipo de gravador (certos tipos não servem para programas);
- ★ A qualidade da fita usada (algumas são imprestáveis!);
- ★ A limpeza, a desmagnetização e o azimute do cabeçote de gravação e leitura;
- ★ O ajuste de volume e de tonalidade da reprodução.

Se você trabalhar sempre com um bom (e simples!) gravador, com fitas de boa qualidade e, preferencialmente, de curta duração (C-20), os problemas com suas gravações serão mínimos!

Capítulo 6

ENRIQUECENDO PROGRAMAS

Até aqui usamos o comando PRINT em sua forma mais simples. Vamos aprender alguns recursos que podem ser usados junto com o PRINT para tornar a impressão no vídeo mais estética.

Introduza o programa a seguir:

```
10 FOR F=1 TO 20
20 PRINT "MSX"
30 NEXT F
```

Comande RUN e observe o resultado.

Esse modo de usar o PRINT nós já conhecemos.

Agora, altere a linha 20 acrescentando uma vírgula ao seu final. O programa ficará assim:

```
10 FOR F=1 TO 20
20 PRINT "MSX",
30 NEXT F
```

Rode-o novamente e compare o resultado com o anterior.

A vírgula divide a tela em duas partes, uma que vai da coluna 0 até a coluna 13 e outra que vai da coluna 14 até a última coluna da direita.

Para ver melhor o que ocorreu, acrescente estas duas linhas temporizadoras ao programa e execute-o novamente.

```
23 FOR G=1 TO 300
26 NEXT G
```

Agora, substitua a vírgula por um ponto e vírgula na linha 20. O programa deverá ficar assim:

```
10 FOR F=1 TO 20
20 PRINT "MSX";
23 FOR G=1 TO 300
26 NEXT G
30 NEXT F
```

Rode-o. Você verá que essa simples modificação altera bastante o resultado (fig. 6.1).

```
10 FOR F=1 TO 20
20 PRINT "MSX";
23 FOR G=1 TO 300
26 NEXT G
30 NEXT F
run
MSXMSXMSXMSXMSXMSXMSXMSXMSXMSXMSXMSXMSXM
SXMSXMSXMSXMSXMSXMSX
Ok
```

Figura 6.1

Existe um outro recurso que facilita ainda mais a impressão no vídeo. Veja o programa a seguir:

```
10 PRINT"Qual o seu primeiro nome?"
20 INPUT N$
30 PRINT"Em que coluna devo imprimi-lo?"
40 INPUT CO
50 PRINT TAB(CO);N$
60 END
Ok
```

TAB funciona como uma espécie de tabulador de máquina de escrever, fixando a coluna para impressão. Na linha 20 você deve introduzir o seu primeiro nome; na linha 40 você deve introduzir a coluna em que ele será impresso e na linha 50 ele é impresso.

Faça agora, algumas alterações para treinar o uso da vírgula, do ponto e vírgula e do TAB.

MOVIMENTOS

Você já sabe como usar variáveis para desenhar. Vamos aprender agora um novo comando que irá nos permitir produzir movimentos na tela.

Digite o programa adiante:

```
10 SCREEN 3
20 FOR X=20 TO 200
30 PSET(X,50),10
40 NEXT X
```

Agora execute-o!

Uma linha deve ser desenhada no vídeo.

Para ver detalhadamente como isso foi feito, acrescente este laço de temporização ao programa e rode-o novamente:

```
33 FOR F=1 TO 40
36 NEXT F
```

Agora, vamos acrescentar mais uma linha ao programa para produzir um ponto em movimento ao invés de uma linha.

Digite a linha a seguir e, mais uma vez, execute o programa:

```
39 PRESET(X,50)
```

O comando PRESET funciona de modo inverso ao PSET. Enquanto PSET desenha, PRESET apaga.

Para aumentar a velocidade do ponto, basta diminuir o valor do último número da linha 33 (laço temporizador).

Agora, tente produzir um movimento da direita para a esquerda e outro de cima para baixo. Após conseguir isso, tente produzir movimentos na diagonal.

COM BOTAS DE 7 LÉGUAS

Observe atentamente o programa a seguir.

```
5 SCREEN 3
10 FOR X=0 TO 255 STEP 10
15 PSET (X,80),10
20 FOR F=1 TO 40
25 NEXT F
30 PRESET (X,80)
35 NEXT X
```

Na linha 10 existe uma palavra do BASIC que nós ainda não conhecíamos: a STEP. Essa palavra diz ao micro de quanto em quanto deve "pular" o valor da variável X. Se não existisse o STEP na linha 10, todos os valores inteiros entre 0 e 255 seriam atribuídos à variável X. O STEP 10 faz com que apenas os valores 0, 9, 19, 29, 39, etc.... sejam atribuídos a X.

Como exercício, tente fazer um programa que desenhe uma tela igual à mostrada na figura 6.2.

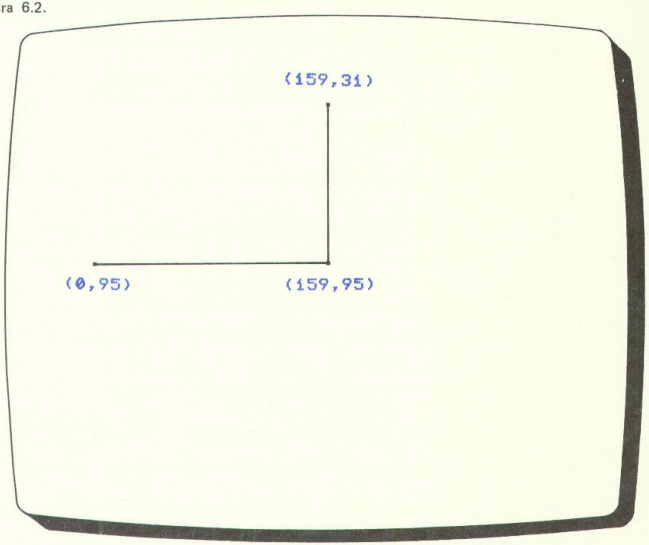


Figura 6.2

MENSAGENS NO PROGRAMA

O BASIC permite que sejam colocadas mensagens no meio de um programa. O comando que permite isso é o REM.

Observe as linhas a seguir:

```
10 REM este programa não faz nada
20 PRINT
30 RUN
```

Ao ler a linha 10 e encontrar o comando REM, o EXPERT imediatamente segue para a linha seguinte (nesse caso, a linha 20). Tudo o que é colocado à frente do REM é ignorado pelo micro.

Geralmente, usa-se o REM para colocar os nomes dos programas nas próprias listagens, ou para anexar lembretes que o próprio programador consultará no futuro.

Quando o mesmo programa é elaborado por mais de uma pessoa, o uso das linhas REM, com mensagens identificando cada bloco de listagem, é indispensável para reduzir confusões.

Capítulo 7
DECISÕES

Na execução de um programa, geralmente é necessário que se façam algumas comparações e, conforme os resultados, se tomem algumas decisões. O BASIC dispõe de uma instrução que torna bastante simples o trabalho do programador e do micro nestas circunstâncias.

Vamos ver como ela pode ser usada. Introduza o seguinte programa:

```
10 CLS : POKE &HFCAB,255
20 PRINT"Pense em um numero inteiro entr
e 1 e 3 ."
30 PRINT"Responda-me com S (sim) ou N (n
ao)."
40 PRINT"O numero e' 1 ?"
60 INPUT R$
70 IF R$="N" THEN GOTO 100
80 PRINT"Consegui acertar na primeira."
90 GOTO 160
100 PRINT"Entao o numero e' 2 ?"
110 INPUT R$
120 IF R$="N" THEN GOTO 150
130 PRINT"Consegui acertar na segunda."
140 GOTO 160
150 PRINT"Certamente o numero e' 3 ?"
160 END
Ok
```

Agora, execute-o!

Sempre que o micro encontra um comando INPUT, interrompe a execução, coloca um ponto de interrogação no vídeo e aguarda a introdução da letra S ou N (sua resposta).

DECISÕES

Quando você introduz uma resposta na variável R$, a linha com a instrução IF... THEN... compara o conteúdo de R$ com a *string* "N". Se a resposta for a letra N, o programa é desviado para a linha em que o micro dá outro *chute.* Caso contrário, ele imprime uma mensagem e vai para a linha 160, onde termina devido ao comando END.

O comando IF /THEN (*SE/ENTÃO*) faz um teste. No exemplo anterior, o comando verifica SE o conteúdo da variável R$ é igual ao conteúdo da *string* "N". Caso o teste seja positivo, o programa segue a partir da linha 100 ou 150. Se o teste for negativo, a linha seguinte é executada.

A estrutura geral da instrução IF/THEN é sempre:

SE (*a condição é verdadeira*) ENTÃO (*execute um comando*)

Em BASIC, isso fica assim:

IF (.....) THEN (.....)

Já vimos que o EXPERT pode armazenar dois tipos de variáveis. Um deles armazena apenas números (variáveis numéricas), e o outro armazena qualquer seqüência de caracteres (variáveis *strings*).

A diferença fundamental entre estes dois tipos está no tratamento que o micro dá a cada um. A distinção entre uma variável numérica e uma variável *string* é a presença ou não do símbolo $ após seu nome.

No primeiro programa que usamos como exemplo neste capítulo, a variável R$ podia armazenar qualquer seqüência de caracteres e, portanto, é uma variável *string.*

Veja mais alguns exemplos de variáveis *strings*:

```
PX$="ALEPH"

GP$="GRADIENTE MSX"

GP$="EXPERT"
```

Vamos analisar o funcionamento de mais um programa. Ele pede que um número seja introduzido e informa se ele é par ou ímpar. Na verdade, o programa verifica apenas se o número é ou não é par, testando se a divisão dele por dois é ou não exata.

```
10 PRINT" ENTRE COM UM NUMERO"
20 INPUT A
30 IF A/2<>INT(A/2) THEN GOTO 60
40 PRINT"PAR"
50 END
60 PRINT"IMPAR"
```

Colocamos o comando END na linha 50 para que a execução não continue na linha 60 quando o número for par. O END serve como uma barreira que não deixa o programa prosseguir. Se quiséssemos, poderíamos também ter colocado uma linha 70 END no programa, porém ela seria desnecessária.

Note que a linha 30 faz uso de uma função que ainda não havíamos utilizado: INT. Ela calcula o valor dentro do parênteses à sua frente e utiliza apenas a parte inteira dele (a rigor, INT calcula o maior número inteiro que é menor ou igual ao número dentro do parênteses). Também na linha 30, encontra-se o sinal <> que significa diferente, isto é, *não igual*. Poderíamos traduzir essa linha mais ou menos assim: SE o valor de A dividido por 2 é diferente da parte inteira dessa divisão ENTÃO vá para linha 60. De uma forma pouco mais sucinta, podemos ainda traduzir: SE A/2 não é inteiro ENTÃO vá para linha 60.

Agora digite RUN e descubra o que acontecerá quando se introduz qualquer número.

Como era de se esperar, as indicações quanto a paridade dos números introduzidos são apresentadas no vídeo.

Tente responder sozinho o que aconteceria se a linha 50 fosse alterada para 50 RUN e a linha 70 RUN fosse acrescentada ao final do programa.

Para verificar sua conclusão, faça com que o programa fique efetivamente assim:

```
10 PRINT" ENTRE COM UM NUMERO"
20 INPUT A
30 IF A/2<>INT(A/2) THEN GOTO 60
40 PRINT"PAR"
50 RUN
60 PRINT"IMPAR"
70 RUN
```

Rode-o e introduza alguns números.

Criamos um laço com o comando RUN mas poderíamos também ter usado GOTO 10 em seu lugar.

COMPLICANDO UM POUCO MAIS

O EXPERT é capaz de tomar decisões muito mais complexas, tais como verificar se um número é maior ou menor que outro, ou ainda, se uma string sucede ou não a outra segundo a ordem alfabética.

O programa que verifica a paridade de um número possui uma falha grave: ele aceita números negativos. Vamos corrigir isso introduzindo a linha:

```
25 IF A<0 THEN RUN
```

Ela fará com que o programa recomece sempre que um número negativo for introduzido. O sinal < significa menor que, de modo que podemos traduzir a linha 25 como: SE o valor de A for menor que 0 ENTÃO recomece novamente. Mais uma vez, poderíamos ter usado GOTO 10 no lugar de RUN.

Se o sinal < for usado numa relação entre *strings*, ele deve ser lido como precede. Observe o programa a seguir:

```
10 PRINT" ENTRE COM UMA PALAVRA"
20 INPUT A$
30 IF A$<"ABACAXI" THEN PRINT"PRECEDE"
40 RUN
```

Sempre que a palavra introduzida através da linha 20 preceder alfabeticamente a palavra ABACAXI, a palavra PRECEDE será impressa no vídeo. Caso contrário, nada será mostrado e o programa simplesmente recomeçará.

Digite e execute esse programa, introduzindo algumas palavras.

UM POUCO DE LÓGICA

Além dos sinais =, <> , < , > , =< e >= , existem algumas palavras que ajudam a testar condições usando critérios lógicos. Veja o programa a seguir:

```
10 PRINT "Entre um numero."
20 INPUT R
30 PRINT "R=";R
40 PRINT "Entre outro numero."
50 INPUT S
60 PRINT "S=";S
70 PRINT "Entre mais um numero."
80 INPUT T
90 PRINT "T=";T
100 IF (R<S) AND (S<T) THEN PRINT "O val
or de R e' menor que o valor de T."
110 IF NOT(R=S) THEN PRINT "O valor de R
 nao e' igual a ao valor de S."
120 IF (R=S) OR (R=T) THEN PRINT "O valo
r de R e' igual ao valor de S ou igual a
o valor de T."
130 RUN
```

As palavras AND, OR e NOT significam respectivamente E, OU e NÃO. Poderíamos traduzir as linhas em que elas são usadas no programa mais ou menos assim:

LINHA 70: SE (R é menor que S) E (S é menor que T) ENTÃO imprima "O VALOR DE R É MENOR QUE O VALOR DE T"

LINHA 80: SE NÃO é verdade que (R é igual a S) ENTÃO imprima "O VALOR DE R NÃO É IGUAL AO VALOR DE S"

LINHA 90: SE (R é igual a S) OU (R é igual a T) ENTÃO imprima "O VALOR DE R É IGUAL AO VALOR DE S OU IGUAL AO VALOR DE T"

Com um pouco de prática, você aprenderá a utilizar facilmente essas palavras do BASIC. Elas são chamadas de operadores lógicos.

Capítulo 8

SORTE OU AZAR?

Muitos problemas resolvidos em computadores envolvem a simulação de situações reais, onde um certo fenômeno depende de fatores imprevisíveis e geralmente aleatórios. Desde os simples joguinhos até os sofisticados programas de aplicação científica, verifica-se a necessidade de algum parâmetro que simule números gerados ao acaso.

Gerar números aleatórios em máquinas precisas como os microcomputadores não é uma tarefa simples. Muitos micros resolvem parcialmente o problema gerando números pseudo-aleatórios, isto é, números produzidos numa seqüência aparentemente desconexa. Num micro desse tipo, um programa que simule um jogo de dados começaria sempre com o mesmo resultado.

O EXPERT, entretanto, pode gerar não apenas números pseudo-aleatórios como também os realmente aleatórios!

Para isso, ele dispõe de algumas funções especiais. Uma delas nós já usamos: a INT. As outras são a RND e a TIME.

Para gerar um verdadeiro número aleatório, o EXPERT utiliza seu relógio interno que mede a passagem de tempo em microssegundos.

O número de microssegundos transcorridos desde o instante em que o micro foi ligado é um parâmetro aleatório e pode ser usado através da função TIME. Para isso, é necessário que um sinal de menos (-) seja colocado antes dessa função. Toda vez que o micro lê uma linha com um -TIME, ele sabe que deve usar essa função para gerar um número aleatório. Veja como podemos usar o RND (abreviatura e contração do inglês RaNDomic number: número aleatório) junto com o -TIME:

```
10 A=RND(-TIME)
20 PRINT A
30 FOR F=1 TO 300
40 NEXT F
50 GOTO 10
```

SORTE OU AZAR?

A linha 10 desse programa atribui um número aleatório para a variável A. A linha 20 imprime esse valor na tela. As linhas 30 e 40 apenas geram um intervalo de tempo, e a linha 50 faz com que tudo recomece a partir da linha 10.

Se você executar várias vezes (com RUN) este programa, verá que a cada execução os números gerados são diferentes. É impossível prever quais serão apresentados no vídeo.

A função RND pode ser usada também para gerar números pseudo-aleatórios. Nesse caso, o -TIME é desnecessário. Veja este outro programa e compare-o com o anterior:

```
10 B=RND(1)
20 PRINT B
30 FOR F=1 TO 300
40 NEXT F
50 GOTO 10
```

Agora, se o programa for repetido várias vezes, a seqüência será sempre a mesma. Os números gerados são pseudo-aleatórios (aparentemente ao acaso).

Vamos ver como a função INT pode ser útil quando queremos sortear números inteiros. Imagine, como exemplo, uma roleta com 100 divisões numeradas de 0 a 99. O programa a seguir simula um sorteio através dessa roleta imaginária.

```
10 C=INT(RND(-TIME)*100)
20 PRINT C
```

O -TIME faz com que a função RND gere números aleatórios entre 0 (inclusive) e 1 (exclusive). Esse número é multiplicado por 100 e, finalmente, a função INT obtém sua parte inteira. O número produzido é um inteiro, entre 0 e 99, e é armazenado na variável C para ser impresso pela linha 20.

UM PROGRAMA ALEATÓRIO

Através dos números aleatórios podemos fazer com que uma ou mais linhas de um programa sejam ou não executadas. O programa apresentado a seguir sorteia uma operação aritmética e dois números para fazer um teste com o usuário. Digite-o e execute-o. Assim ele será facilmente entendido.

Vamos analisar algumas linhas desse programa.

A linha 20 sorteia um número inteiro entre 1 e 4 (verifique como!) e o armazena na variável OP. Cada número corresponde a uma operação aritmética:

1 . . . . . .ADIÇÃO
2 . . . . . .SUBTRAÇÃO
3 . . . . . .MULTIPLICAÇÃO
4 . . . . . .DIVISÃO

As linhas 30 e 40 sorteiam, cada uma, um número inteiro entre 1 e 100. Esses números serão usados junto com a operação sorteada.

As linhas entre 60 e 90 usam o comando GOTO. Nós já o conhecemos e sabemos que ele desvia a execução do programa para a linha especificada à sua frente. Como OP pode valer 1, 2, 3 ou 4, o desvio é feito para a linha 100, 200, 300 ou 400.

```
10 PRINT : PRINT : PRINT "Vou sortear um
a operacao e dois numeros."
20 OP=INT(RND(-TIME)*4)+1
30 N1=INT(RND(-TIME)*100)+1
40 N2=INT(RND(-TIME)*100)+1
50 PRINT "Responda o mais rapido que voc
e puder.","Quanto e'";
60 IF OP=1 THEN GOTO 100
70 IF OP=2 THEN GOTO 200
80 IF OP=3 THEN GOTO 300
90 IF OP=4 THEN GOTO 400
100 PRINT N1;"+";N2;"?"
110 INPUT RE
120 IF RE<>N1+N2 THEN PRINT "Voce errou
!" : PRINT "E'";N1+N2;"."
130 IF RE=N1+N2 THEN PRINT "Voce acertou
!"
150 GOTO 10
200 PRINT N1;"-";N2;"?"
210 INPUT RE
220 IF RE<>N1-N2 THEN PRINT "Voce errou
!" : PRINT "E'";N1-N2;"."
```

```
230 IF RE=N1-N2 THEN PRINT "Voce acertou
!"
250 GOTO 10
300 PRINT N1;"x";N2;"?"
310 INPUT RE
320 IF RE<>N1*N2 THEN PRINT "Voce errou
!" : PRINT "E'";N1*N2;"."
330 IF RE=N1*N2 THEN PRINT "Voce acertou
!"
350 GOTO 10
400 PRINT N1;":";N2;"?"
410 INPUT RE
420 IF RE<>N1/N2 THEN PRINT "Voce errou
!" : PRINT "E'";N1/N2;"."
430 IF RE=N1/N2 THEN PRINT "Voce acertou
!"
450 GOTO 10
Ok
```

Nas linhas entre 100 e 150 é feito um teste com a soma. Nas linhas entre 200 e 250 é feito um teste com a subtração. Nas linhas entre 200 e 350 o teste é com a multiplicação e nas linhas entre 400 e 450 com a divisão.

Note que o programa é executado várias vezes, pois as linhas 150, 250, 350 e 450 contém comandos GOTO 10 que criam um laço.

Se você não conseguiu entender completamente o funcionamento deste programa (o que, no começo, é muito provável) estude-o atentamente quantas vezes forem necessárias para eliminar suas dúvidas.

Não desista!

É fundamental ter paciência e perseverança!

ON ... GOTO

Se você o entendeu bem, está em condições de aprender outras formas de produzir desvios na execução de um programa.

Vamos alterar o programa "QUATRO OPERAÇÕES".

Apague as linhas 60, 70, 80 e 90 e introduza a linha 70 apresentada a seguir:

```
70 ON OP GOTO 100,200,300,400
```

Agora, execute novamente o programa e veja se o resultado geral se alterou.

O programa deve funcionar do mesmo jeito que antes.

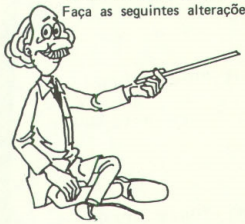
A instrução ON ... GOTO ... na linha 70 é a responsável pelos desvios. Se OP = 1, o desvio é feito para o primeiro número após o GOTO; se OP = 2, o desvio é feito para o segundo número após o GOTO, e assim por diante.

SUB-ROTINAS

Além dessa opção, existe ainda uma outra que é geralmente utilizada quando um grupo de linhas deve ser repetido várias vezes em situações diferentes. Essas linhas a serem executadas várias vezes mas não consecutivamente, são chamadas de SUB-ROTINAS.

Vamos ver como podemos utilizar as sub-rotinas no programa QUATRO OPERAÇÕES.

Faça as seguintes alterações:



```
60 IF OP=1 THEN GOSUB 100
70 IF OP=2 THEN GOSUB 200
80 IF OP=3 THEN GOSUB 300
90 IF OP=4 THEN GOSUB 400
95 GOTO 10
150 RETURN
250 RETURN
350 RETURN
450 RETURN
```

O comando GOSUB nas linhas entre 60 e a 90 faz com que o micro execute uma sub-rotina. Com as últimas alterações, o programa QUATRO OPERAÇÕES ficou com quatro sub-rotinas, uma para cada operação. Se o valor de OP for 1, a sub-rotina que inicia na linha 100 é executada. Ao encontrar o comando RETURN na linha 150, a execução retorna da sub-rotina e quando executa a linha 95 o programa volta à linha 10 e faz um novo sorteio. O mesmo acontece quando as outras sub-rotinas são executadas.

Do mesmo modo que podemos usar ON ... GOTO ... podemos usar também ON ... GOSUB ... . Tente você mesmo fazer a alteração necessária no programa QUATRO OPERAÇÕES, usando ON ... GOSUB ... .

Capítulo 9

EXPERT EM CÁLCULOS

Você pode usar o EXPERT como se ele fosse uma calculadora, apesar de que isso seria um pouco como "matar moscas a tiros de canhão". Para usar o EXPERT "espertamente", porém, é necessário que você conheça as regras básicas da aritmética.

Já vimos como usar as quatro operações fundamentais com o comando PRINT. Além de somar, subtrair, multiplicar e dividir, podemos também fazer exponenciações e radiciações através do sinal ^. Por exemplo, se quisermos calcular quanto é 2 elevado ao cubo ($2^3 = 8$) basta comandar:

```
PRINT 2^3
```

De modo semelhante, se quisermos saber quanto vale a raiz cúbica de 8, devemos usar o sinal ^ na forma:

```
PRINT 8^(1/3)
```

Para o EXPERT, *a* ^ *b* significa *a* elevado a *b* e *a* ^ (1/*b*) significa raiz *b* de *a*.

Uma expressão matemática pode conter as seis operações já comentadas, mais os operadores lógicos (AND, OR e NOT) e os operadores de relação ( ) , (, = ,)= , => e ( )). Há também a possibilidade do uso de parênteses, como são usados normalmente.

A tabela a seguir mostra a ordem de prioridade com que o EXPERT executa as operações.

1.	Parênteses	(e)
2.	Operador de negação	NOT
3.	Multiplicação e divisão	* e /
4.	Soma e subtração	+ e -
5.	Operadores de relação	( ) = (=)= ( )
6.	Operador de conjunção	AND
7.	Operador de disjunção	OR

EXPERT EM CÁLCULOS

Se existem várias operações numa linha de um programa, o EXPERT, ao executá-las, as realiza na ordem de prioridade definida pela tabela anterior. Caso existam duas ou mais operações com mesma prioridade, ele as executa na ordem em que estão escritas, da esquerda para direita.

Por exemplo, se você introduzir:

```
PRINT 60+11-14
```

o resultado será 57, pois 60+11=71 e 71-14=57.

Se introduzirmos:

```
PRINT 60-14+11
```

o resultado será o mesmo 57, pois 60-14=46 e 46+11=57.

Tente descobrir o resultado do comando a seguir.

```
PRINT 2+((3*(3+15/5)-8)-1)/3
```

Se a sua resposta foi cinco, você acertou!

O EXPERT realiza as operações na ordem em que nós mesmos as faríamos. Vejamos:

```
2 + ((3 * (3+15/5) - 8) - 1) /3 =
  2 + ((3 * (3+3) - 8) - 1) /3 =
    2 + ((3 * 6 - 8) - 1) /3 =
      2 + ((18 - 8) - 1) /3 =
        2 + (10 - 1) /3 =
            2 + 9/3 =
              2 + 3 = 5
```

Os operadores de relação e os operadores lógicos são usados normalmente em expressões que vão ser testadas por uma instrução IF... THEN... e, nesse caso, tem a prioridade estabelecida pela tabela. Por exemplo, veja a linha a seguir:

```
100 IF NOT(3=B) AND (PP>4) OR (PP=NMA)
THEN PRINT"DMPHPELA"
```

Nós podemos traduzi-la assim:

LINHA 100: SE (
(NÃO é verdade que (3 = B) E (PP> 4))
OU
(PP = NMA)
) ENTÃO imprima "DMPHPELA"

Portanto, a palavra DMPHPELA será impressa quando a variável B contiver um número diferente de 3 e, simultaneamente, a variável PP contiver um número maior que 4 ou, então, quando a variável PP contiver o mesmo número que a variável NMA.

PROCESSANDO INFORMAÇÕES

Nós descrevemos um PROGRAMA como sendo um conjunto de instruções a serem executadas numa determinada seqüência. As variáveis que um programa necessita para começar o processamento podem ter seus conteúdos introduzidos de diversos modos. Três deles nós já vimos: o INPUT, o LET e a atribuição direta (como ALA = 190462).

Veremos agora como introduzir dados no próprio programa, através do comando READ. Esse comando tem ligado a si duas outras palavras do BASIC: DATA e RESTORE.

Veja o programa a seguir. Vamos analisar como ele funciona.

```
10 DATA EXPERT,MSX,GRADIENTE,ALEPH,BASIC
,MICRO,COMPUTADOR
20 FOR H=1 TO 7
30 READ A$
40 PRINTA$
50 NEXT H
60 RESTORE
70 GOTO 20
```

Na linha 10 estão armazenadas sete (7) palavras. Quando o micro encontra nela a palavra DATA, ele já sabe que ali estão armazenados dados.

A linha 20 inicia um laço que será repetido sete vezes e que termina na linha 50. A variável H serve apenas para controlar o laço.

Na linha 30 está o comando READ A$. Quando ele é executado pela primeira vez, faz com que o micro atribua, como conteúdo da variável A$, a primeira palavra armazenada logo após o DATA na linha 10.

A linha seguinte, com número 40, imprime na tela o conteúdo de A$, ou seja, a palavra EXPERT.

A linha 50 fecha o laço aberto na linha 20 e faz com que a próxima palavra (MSX) seja lida e impressa na tela.

Cada vez que o READ é repetido, uma palavra diferente é atribuída a A$. O laço é repetido sete vezes porque a linha DATA contém apenas sete palavras.

A linha 60 contém um RESTORE. Esse comando faz com que a próxima palavra a ser lida pelo READ seja, novamente, a primeira logo após a instrução DATA na linha 10, isto é, EXPERT.

Finalmente, a linha 70 desvia a execução para a linha 20, e tudo recomeça...

Nesse exemplo, usamos a linha com DATA logo no início do programa, porém ela pode ser colocada em qualquer lugar. Além disso, ao invés de usar apenas uma linha para inserir os dados, poderíamos ter usado sete linhas com um dado em cada uma. Quando há várias linhas com a instrução DATA num programa, o micro as trata como se fossem uma só, desde que contenham o mesmo tipo de dados (só números ou só *strings*).

Poderíamos, também, ter usado outro nome para a variável *string* A$ como, por exemplo, GR$, MA$, ou qualquer outro.

As informações que são armazenadas numa linha com DATA devem ser separadas por vírgulas.

Um programa pode, também, ter vários comandos READ usando a mesma linha DATA. Observe o próximo programa. Ele calcula a média (aritmética) entre 10 números armazenados numa linha com DATA e depois verifica qual o maior e o menor entre eles.

Analise-o você mesmo e justifique cada uma de suas linhas. Tente prever o resultado e depois verifique a previsão, executando-o.

```
10 REM Programa MEDIA
20 FOR F=1 TO 10
30   READ NUM
40   SOMA=SOMA+NUM
50 NEXT F
60 MEDIA=SOMA/10 : MN=MEDIA : MX=MEDIA
70 PRINT"A media e'";MEDIA;"." : RESTORE
80 FOR G=1 TO 10
90   READ DADO
100  IF DADO<MN THEN MN=DADO
110  IF DADO>MX THEN MX=DADO
120 NEXT G
130 PRINT "O maior dos numeros e'";MX;".
","O menor e'";MN;"."
140 DATA 1,2,3,4,5,6,7,8,9,10
Ok
```

OUT OF DATA

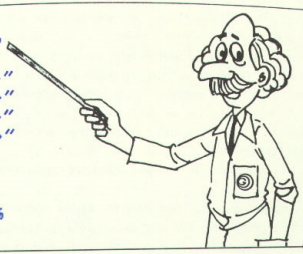
Quando um comando READ é executado e todos os dados de uma linha DATA já foram lidos, o EXPERT apresenta na tela a mensagem OUT OF DATA. Essa mensagem de erro pode ser evitada com um comando RESTORE. O programa a seguir produz essa mensagem.

```
10 DATA GENESIS,APOCALIPSE
20 READ A$
30 READ B$
40 READ C$
50 READ D$
60 PRINT A$,B$,C$,D$
```

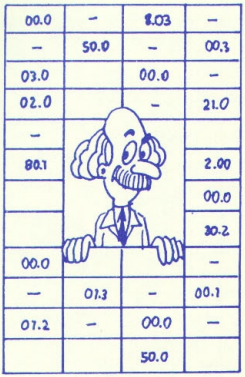
ARMAZENANDO MUITAS STRINGS

Quando o EXPERT é ligado, ele reserva uma certa área (200 bytes) para armazenar variáveis *strings* em sua memória. Nos programas que usam muitas variáveis *strings*, temos que avisar ao micro para que ele reserve mais espaço em sua memória. O comando que permite isso é o CLEAR. Observe o programa a seguir.

```
10 CLEAR 800
20 CLS
30 R$=R$+"()"
40 B$=B$+"()"
50 C$=C$+"()"
60 D$=D$+"()"
70 PRINT R$
80 PRINT B$
90 PRINT C$
100 PRINT D$
110 GOTO 30
```



A linha 10 diz ao micro para reservar 800 casas (bytes) em sua memória para armazenar variáveis *strings*.



Capítulo 10

AS TABELAS NO MICRO

Até agora sempre usamos as variáveis como unidades individuais. Isto significa que temos considerado cada variável como uma unidade distinta. A, B, C$ e D$ são exemplos de variáveis individuais, nas quais nós colocamos números e/ou palavras. Cada variável desse tipo tem que ser definida individualmente.

Em muitas situações, é necessário usar muitas variáveis de modo que elas sejam facilmente acessíveis dentro de uma certa seqüência. Por exemplo, imagine que você queira armazenar uma agenda telefônica na memória do micro. Se a agenda for pequena, um programa como o que é mostrado a seguir pode resolver o problema.

```
10 '   Agenda Pequena
20 LET A$="Leia Organa ..... 222-2222"
30 LET B$="Jabba ........... 233-1774"
40 LET C$="Obi-Wan Kenobi .. 745-8278"
50 LET D$="Han Solo ........ 777-2001"
60 LET E$="Darth Vader ..... 288-5894"
70 LET F$="C3-PO ........... 848-4984"
80 LET G$="R2-D2 ........... 949-0403"
90 PRINT A$,B$,C$,D$,E$,F$,G$
```

E se a agenda tiver mais de cem nomes?

Como podemos resolver o problema?

AS TABELAS NO MICRO

O BASIC permite a criação simultânea de muitas variáveis cujos "nomes" diferem apenas em um ou mais números. Elas são chamadas de VARIÁVEIS INDEXADAS, pois possuem índices de identificação. Nós podemos imaginar uma agenda telefônica como uma tabela de duas colunas. A primeira coluna é a dos nomes e a segunda é a dos telefones.

AGENDA TELEFÔNICA

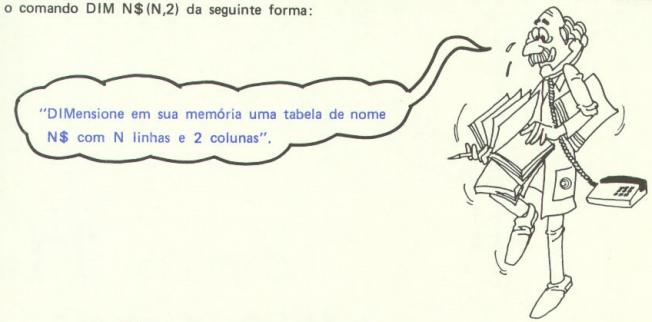
NOMES	TELEFONES
Blaise Pascal	623-1662
Charles Babbage	792-1871
George Boole	815-1864
Hermann Hollerith	860-1929
Norbert Wiener	894-1964
Johannes von Neumann	903-1957

Vamos ver como podemos introduzir essa tabela na memória do EXPERT. Observe atentamente o programa a seguir:

```
10 REM Agenda EXPERTa
20 PRINT:PRINT:PRINT"Consultar (C) ou IN
SERIR (I)"; : POKE &HFCAB,255
30 INPUT S$
40 IF S$="C" THEN GOTO 160
50 IF S$="I" THEN GOTO 70
60 GOTO 30
70 PRINT"Quantos nomes serao inseridos";
80 INPUT N : IF A>0 THEN ERASE N$
90 A=1 : DIM N$(N,2)
100 FOR F=1 TO N
110 PRINT "Entre o nome";F;":"
120 INPUT N$(F,1)
130 PRINT "Entre o telefone:"
140 INPUT N$(F,2)
150 NEXT F : GOTO 10
160 PRINT : PRINT "Nome","Telefone"
170 PRINT
180 FOR F=1 TO N
190 PRINT N$(F,1),N$(F,2)
200 NEXT F
300 GOTO 10
Ok
```

Ao ser executado, esse programa pergunta se o usuário quer consultar a agenda ou introduzir nomes e telefones.

Da linha 10 até a linha 80 você já tem condições de entender sozinho. A linha seguinte, de número 90, é a que nos interessa. Nela existe um comando DIM, que cria uma tabela na memória do computador. A nossa agenda pode então ser armazenada nessa tabela. Podemos "traduzir" o comando DIM N$(N,2) da seguinte forma:



Cada linha da tabela N$ conterá duas posições para armazenamento de informações. A linha 120 permite a intrudução do nome na primeira posição e a linha 140 permite a introdução do telefone na segunda posição. Poderíamos armazenar nossa agenda da seguinte forma:

N$(1,1) = "Blaise Pascal"	N$(1,2) = "623-1662"
N$(2,1) = "Charles Babbage"	N$(2,2) = "792-1871"
N$(3,1) = "George Boole"	N$(3,2) = "815-1864"
N$(4,1) = "Hermann Hollerith"	N$(4,2) = "860-1929"
N$(5,1) = "Norbert Wiener"	N$(5,2) = "894-1964"
N$(6,1) = "Johannes von Neumann"	N$(6,2) = "903-1957"

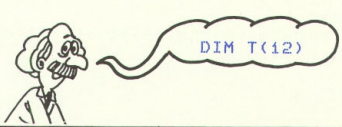
Cada posição da tabela é uma variável indexada (com dois índices!) e seus nomes diferem apenas nos índices.

As tabelas são também conhecidas como MATRIZES ou como ARRAYS e seus elementos podem ter um, dois, três ou mais índices.

OUTRO EXEMPLO

Se você quiser introduzir uma tabela com as taxas mensais de inflação durante um ano, pode usar uma MATRIZ numérica com apenas um índice e criá-la através do comando: DIM T(12).

Cada posição da tabela T será acessada através de um índice e conterá a taxa de inflação num dado mês.



MATRIZ T

T(1)	T(2)	T(3)	T(4)	T(5)	T(6)	T(7)	T(8)	T(9)	T(10)	T(11)	T(12)
9,45	8,53	7,88	9,95	6,33	6,12	6,87	8,04	9,15	11,43	12,59	14,20

Capítulo 11

SEQÜÊNCIAS DE CARACTERES

Uma *string* é uma seqüência de caracteres, geralmente formando uma palavra ou uma frase. Existem comandos do BASIC que permitem a alteração da ordem dos caracteres dentro de uma dada *string:* o LEFT$, o RIGHT$, o MID$ e o LEN.

Vamos aprender como utilizá-los, usando para isso o programa listado a seguir.

```
10 REM OGIRANDOGIRANDOGIRANDO...
20 CLS
30 PRINT"INTRODUZA UMA PALAVRA"
40 INPUT Q$
50 P$="    "
60 FOR R=1 TO LEN(Q$)
70 P$=P$+" "+MID$(Q$,R,1)
80 NEXT R
90 LOCATE 10,12
100 PRINT P$
110 P$=RIGHT$(P$,LEN(P$)-1)+LEFT$(P$,1)
120 FOR F=1 TO 50
130 NEXT F
140 GOTO 90
```

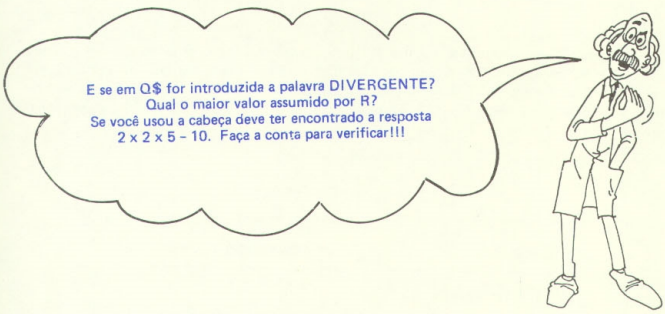
Nesse programa há, ainda, um outro comando que nós não conhecíamos: o LOCATE. Ele serve para localizar uma posição da tela onde se quer imprimir alguma coisa e nós o estudaremos mais adiante.

Vamos analisar o programa listado, linha por linha, para entender o seu funcionamento e prever qual o resultado produzido quando ele for executado.

LINHA 10: Apenas contém uma mensagem (o nome do programa).
LINHA 20: Limpa a tela.
LINHA 30: Imprime uma mensagem na tela pedindo a introdução de uma palavra através do teclado.
LINHA 40: Permite a introdução de uma *string* através do teclado e a armazena na variável Q$.
LINHA 50: Atribui três espaços como conteúdo à variável P$.
LINHA 60: Abre um laço que será repetido tantas vezes quantas forem os caracteres contidos em Q$. A instrução LEN calcula quantos caracteres contêm a variável *string* cujo nome é colocado dentro dos parênteses à sua frente. Por exemplo, se em Q$ for introduzida a palavra GRADIENTE, a variável de controle R assumirá valores de 1 até 9, pois serão 9 os caracteres de Q$ e conseqüentemente LEN (Q$) = 9.

Q$ = " G R A D I E N T E "
1 2 3 4 5 6 7 8 9

LEN (Q$) = 9



LINHA 70: Esta linha altera o conteúdo da variável P$. Do mesmo modo que podemos atualizar uma variável numérica (por exemplo, comandando G = G + 1) podemos também atualizar uma variável *string*. Na linha 70 o sinal de soma (*ü*) significa junção. Se a variável P$ contiver a palavra ROTACIONAL e comandarmos P$ = P$ + "CAMPO", seu novo conteúdo será ROTACIONALCAMPO.

"ROTACIONAL" + "CAMPO" = "ROTACIONALCAMPO"

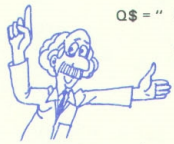
COMANDO	CONTEÚDO DE P$
P$ = "ROTACIONAL"	ROTACIONAL
P$ = P$ + "CAMPO"	ROTACIONALCAMPO

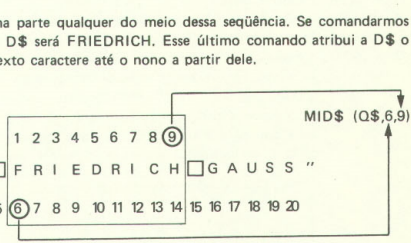
SEQÜÊNCIAS DE CARACTERES

Temos, ainda, que aprender como funciona a instrução MID$ para que possamos entender a linha 70. Imagine que a variável Q$ contenha a seqüência de caracteres CARL FRIEDRICH GAUSS. Cada caractere possui uma ordem na seqüência.

```
Q$ = " C A R L □ F R I E D R I C H □ G A U S S "
     ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑
     1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
```

A função MID$ obtém uma parte qualquer do meio dessa seqüência. Se comandarmos D$ = MID$ (Q$,6,9), o conteúdo de D$ será FRIEDRICH. Esse último comando atribui a D$ o conteúdo da variável Q$ desde o seu sexto caractere até o nono a partir dele.





Agora podemos entender a linha 70. Imagine que na linha 40 tenha sido introduzida a palavra STOKES na variável Q$. O valor de LEN (Q$) será 6 e o laço que inicia na linha 60 será repetido 6 vezes. Quando o micro executa pela primeira vez a linha 70, o conteúdo de P$ será de três espaços. Após a execução da linha 70, P$ passa a conter "□□□" + "□" + "S", ou seja, quatro espaços mais uma letra S.

"□□□" + "□" + "S" = "□□□□S"

COMANDO	CONTEÚDO DE P$
P$ = "□□□"	□□□
P$ = P$ + "□" + MID$ (Q$,R,1)	□□□□S

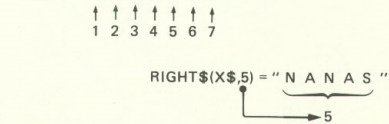
Na segunda execução da linha 70, P$ = "□□□□S" e R = 2. Portanto, após a segunda execução, a variável P$ conterá a seqüência □□□□S□T. Após a terceira execução, o conteúdo de P$ será □□□□S□T□O. Após a quarta execução, P$ será□□□□S□T□O□K; depois da quinta, P$ conterá□□□□S□T□O□K□E e, finalmente, após a sexta execução, P$ será□□□□S□T□O□K□E□S.

LINHA 80: Fecha o laço aberto na linha 60.

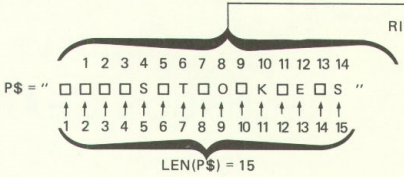
LINHA 90: Fixa a posição (coluna,linha) em que será executada a próxima impressão com PRINT na tela. O comando LOCATE 10,12 faz com que o próximo comando PRINT imprima a partir da coluna 10 e da linha 12 da tela. Lembre-se que a tela de texto tem, normalmente, 39 colunas e 24 linhas.

LINHA 100: Imprime o conteúdo de P$ na posição fixada pelo comando LOCATE na linha 90.

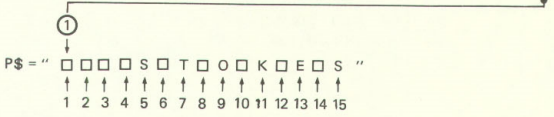
LINHA 110: Atualiza o conteúdo de P$. As funções RIGHT$ e LEFT$ são semelhantes a MID$. RIGHT$(X$,N), por exemplo, contém os N caracteres do lado direito da *string* X$.



LEFT$, analogamente, obtém caracteres do lado esquerdo de uma *string*. Se a *string* P$ contiver a seqüência□□□□S□T□O□K□E□S, obviamente, LEN (P$) será 15 e LEN (P$)-1 será 14. RIGHT$(P$,LEN(P$)-1) é equivalente aos 14 caracteres da direita de P$, ou seja: □□□S□T□O□K□E□S.



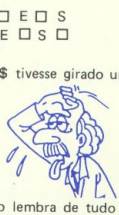
LEFT$(P$,1) é o primeiro caractere da esquerda de P$, ou seja:



Após a primeira execução da linha 110, P$ será □□□S□T□O□K□E□S□.

* antes da primeira execução: □□□□ S □ T □ O □ K □ E □ S
* após a primeira execução: □□□S □ T □ O □ K □ E □ S □

Como se pode observar, tudo se passa como se a *string* armazenada em P$ tivesse girado uma posição para esquerda!



LINHA 120: Abre um laço de temporização.
LINHA 130: Fecha o laço de temporização.
LINHA 140: Retorna a execução para linha 90.

Este capítulo é como uma longa *string*. Provavelmente você não lembra de tudo o que leu e, por isso, é recomendável olhar novamente o programa que usamos como exemplo no início deste capítulo e não seguir adiante até entendê-lo. Na verdade ele é bem simples e apenas faz com que uma palavra fique girando na tela.

Capítulo 12
RECURSOS GRÁFICOS

Vamos começar a estudar os recursos gráficos do EXPERT analisando o programa a seguir:

```
10 SCREEN 2
20 FOR C=2 TO 15
30 PAINT(128,80),C
40 NEXT C
```

A primeira linha que o micro executa é a de número 10. Nela há um comando SCREEN 2 que seleciona qual das teclas disponíveis o EXPERT vai usar. A tela escolhida através de SCREEN 2 é a de alta resolução, dividida em 256 colunas e 192 linhas. Lembre-se: as colunas são numeradas de 0 a 255 da esquerda para a direita e as linhas são numeradas de 0 a 191 de cima para baixo.

A linha 20 abre um laço que será repetido 14 vezes e cuja variável de controle, C, assume valores inteiros entre 2 e 15.

A linha 30 contém um comando PAINT (128,80),C que faz com que a tela seja pintada com uma certa cor. Poderíamos traduzí-lo assim: "Pinte a região da tela ao redor do ponto localizado na coluna 128 e na linha 80 com a cor cujo código é C".

A linha 40 apenas fecha o laço aberto na linha 20.

Cada vez que a linha 30 é executada, o valor de C é diferente e, portanto, a tela é colorida com uma cor diferente. O ponto de coordenadas 128,80 fica aproximadamente no centro da tela. Como não há nenhuma linha que o cerque por todos os lados, o micro con sidera que a região ao seu redor é a mais extensa possível, ou seja, toda a tela. Vamos compreender essas informações mais claramente com auxílio de outro comando: o CIRCLE.

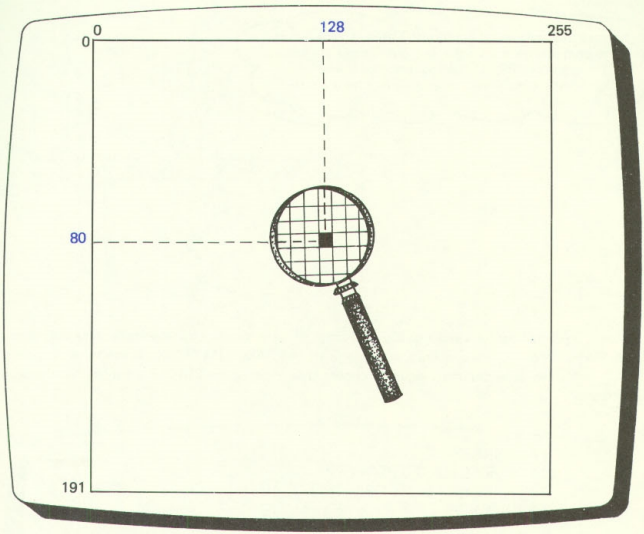


Figura 12.1

Observe o programa a seguir:

```
10 SCREEN 2
20 FOR C=1 TO 15
30 CIRCLE(128,80),50,C
40 PAINT(128,80),C
50 NEXT C
```

As linha 10, 20, 40 e 50 você já sabe como funcionam.

O comando CIRCLE (128,80),50,C manda o micro traçar uma circunferência ao redor do ponto de coordenadas (COLUNA=128, LINHA=80), com raio igual a 50 pontos e com a cor cujo código é C. Agora, quando o comando PAINT é executado, existe uma linha (a circunferência) que serve como fronteira envolvendo o ponto (128,80). O micro considera que a região a ser pintada é a que fica dentro da circunferência. Note que usamos a mesma cor no comando CIRCLE e no comando PAINT.

Sempre que estivermos usando a SCREEN 2, devemos fazer com que a cor da linha da fronteira seja a mesma usada por PAINT. Se esquecermos disso, será como se não existisse a fronteira!

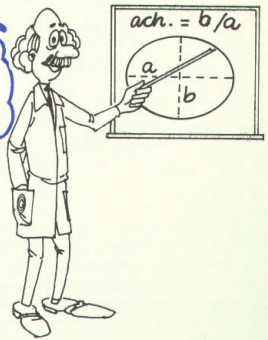
Poderíamos ter usado qualquer ponto do interior da circunferência para definir o comando PAINT. Por exemplo, a linha 40 poderia ser: 40 PAINT (117,63),C. O efeito seria o mesmo.

Vamos aprender mais algumas opções que o comando CIRCLE permite. Veja o programa adiante:

```
10 SCREEN 2
20 FOR C=2 TO 15
30 CIRCLE(128,80),25*C-40,C,,,3
40 NEXT C
```

Ao ser executado, esse programa desenha 14 elípses concêntricas na tela.

Uma elipse é uma espécie de circunferência achatada (ou se você achar melhor, ovalada), Em toda elipse existem dois eixos ou diâmetros que a definem completamente. Um deles é o maior eixo que pode ser desenhado no interior da elipse e o outro é o menor eixo. O achatamento de uma elipse é a razão entre seu eixo menor e seu eixo maior. Quando os dois eixos forem iguais, o achatamento vale 1 e a elipse se transforma numa circunferência!



A forma mais geral de se usar o CIRCLE é:

CIRCLE (coluna,linha),raio,cor,ang. inicial,ang. final, achatamento

Nós já sabemos usar os primeiros 4 números contidos nesse comando (coluna,linha,raio e cor). O último número (achatamento) faz com que a circunferência se transforme numa elipse. O EXPERT considera como achatamento a razão entre o eixo vertical e o eixo horizontal da elipse na tela.

Agora, vamos analisar o programa que desenha elipses.

A linha 10 seleciona a SCREEN 2, isto é, a tela de alta resolução.

A linha 20 inicia um laço que será repetido 14 vezes e cuja variável de controle C assume valores inteiros entre 2 e 15.

A linha 30 faz com que seja desenhada uma elipse com as seguintes características na tela:

1 – centro na posição	coluna=128 e linha=80
2 – raio maior	25*C-40
3 – cor	C
4 – achatamento	3

Consequentemente, o raio maior e a cor da elipse dependem do valor de C.

A linha 40 apenas fecha o laço aberto na linha 20.

Tente você mesmo alterar várias vezes os números usados na linha 30 e analise o que acontece em cada caso. Aqui estão algumas sugestões:

```
☆ 30 CIRCLE(128,80),25*C-40,C,,,1/3
☆ 30 CIRCLE(128,80),15*C-30,C,,,1/4
☆ 30 CIRCLE(128,80),5*C,C,,,C/30
```

Nos dois programas que usamos, a imagem gerada desaparecia logo após a execução da última linha. Em capítulos anteriores nós já aprendemos como fazer para que a imagem permaneça no vídeo. Veja este outro programa e tente entendê-lo:

```
10 SCREEN 2
20 FOR F=55 TO 200 STEP 4
30 CIRCLE(F,80),50,12
40 NEXT F
50 GOTO 50
```

Agora, após a confecção da figura na tela, a linha 50 faz com que o programa continue indefinidamente em execução. Para interrompê-lo deve-se digitar CONTROL+STOP.

Vamos agora tentar entender os dois outros parâmetros (ang. inicial e ang. final) que podem ser usados no comando CIRCLE. Limpe a tela e digite mais este exemplo:

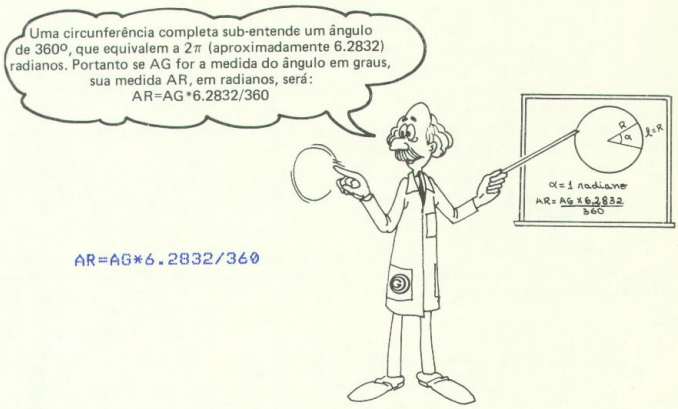
```
10 SCREEN 2
20 PI=3.14159
30 DP=2*PI
40 FOR F=1 TO 10
50 CIRCLE(128,80),8*F,4,0,DP*F/10,1
60 NEXT F
70 GOTO 70
```

RECURSOS GRÁFICOS

A linha 50 desenha uma circunferência com as seguintes características:

1 – centro na posição	coluna=128 e linha=80
2 – raio	8*F (variável)
3 – cor	4 (azul escuro)
4 – ângulo inicial	0
5 – ângulo final	DP*F/10 (variável)
6 – achatamento	1

O EXPERT sempre trabalha com ângulos em radianos. Se um programa usa ângulos em graus, deve-se transformá-los em radianos para usá-los com o CIRCLE.



Para terminar com a função CIRCLE, apresentamos mais um exemplo que gera figuras parecidas com as que surgem na superfície da água quando duas gotas a atingem simultaneamente.

```
10 REM GOTAS
20 SCREEN 2
30 C=2+14*RND(-TIME)
40 C1=256*RND(-TIME)
50 L1=192*RND(-TIME)
60 C2=256*RND(-TIME)
70 L2=192*RND(-TIME)
80 FOR F=0 TO 300 STEP 4
90 CIRCLE(C1,L1),F,C
100 CIRCLE(C2,L2),F,C
110 NEXT F
120 FOR F=1 TO 500
130 NEXT F
140 GOTO 10
```

LINHAS E QUADRADOS

Vamos, agora, aprender como desenhar rapidamente linhas e quadrados na tela usando o comando LINE, que é tão eficiente e versátil quanto o CIRCLE.

Observe o seguinte programa:

```
10 SCREEN 2
20 LINE(10,15)-(114,145)
30 GOTO 30
```

Ao ser executado, ele desenha uma linha desde o ponto situado na coluna 10 e linha 15 até o ponto situado na coluna 114 e linha 145.

Do mesmo modo que o CIRCLE, o LINE também permite a redefinição da cor usada para desenhar. O programa a seguir faz a mesma coisa que o anterior, mas o faz várias vezes seguidas sendo que em cada uma delas, usa uma cor aleatória. Vamos analisá-lo:

```
10 SCREEN 2
20 C=2+14*RND(-TIME)
30 LINE(10,15)-(114,145),C
40 GOTO 20
```

A linha 10 define a tela a ser usada (nesse caso, a tela de alta resolução). A linha 20 sorteia um código de cor entre 2 e 15 e o armazena na variável C. A linha 30 traça uma linha ligando os pontos situados nas coordenadas especificadas dentro dos parênteses e com a cor definida por C. A linha 40, finalmente, reinicia tudo a partir da linha 20.

Vamos, agora, construir um quadro.

Acrescente uma vírgula e a letra B ao fim da linha 30, deixando-a assim:

```
30 LINE(10,15)-(114,145),C,B
```

Agora, ao executá-la, o micro entende que deve desenhar um quadro entre os pontos colocados dentro dos parênteses (figura 12.2).

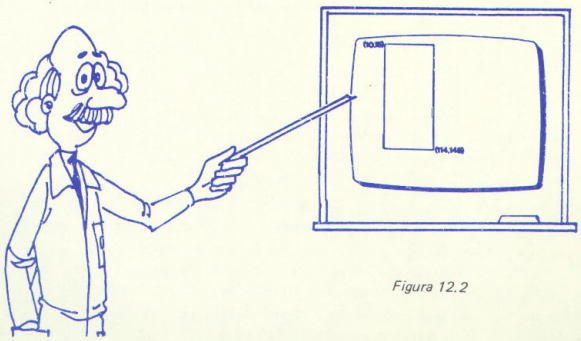


Figura 12.2

Os dois pontos usados ficam sendo dois vértices do quadro desenhado.

Podemos também colorir a parte interna do quadro simplesmente acrescentando uma letra F logo após a letra B. Mais uma vez altere a linha 30 e deixe-a assim:

```
30 LINE(10,15)-(114,145),C,BF
```

Agora toda a região dentro do quadro é pintada com a mesma cor do contorno. Tente entender o que produz este outro programa:

```
10 SCREEN 2
20 FOR X1=50 TO 205 STEP 4
30 X2=255-X1
40 C=2+14*RND(-TIME)
50 Y1=X1*191/255
60 Y2=X2*191/255
70 LINE(X1,Y1)-(X2,Y2),C,BF
80 NEXT X1
90 GOTO 20
```

DESENHANDO QUALQUER COISA

O EXPERT possui muitos outros recursos gráficos incorporados no que chamamos de MACRO LINGUAGEM GRÁFICA. A porta de acesso a essa sub-linguagem do BASIC MSX é o comando DRAW. Sua sintaxe é sempre DRAW *string*, onde *string* pode ser uma constante ou uma variável contendo códigos gráficos especiais.

Observe o programa adiante. Ele utiliza quase todos os recursos do DRAW na linha 50 para produzir uma bela figura na tela.

```
10 REM DRAWLIPSE 1
20 SCREEN 2
30 FOR A=0 TO 6.28 STEP .1
40 PSET(128+50*COS(A),85+20*SIN(A)),5
50 DRAW "nu50ne50nr50nf50nd50ng50nl50nh5
0"
60 NEXT A
70 GOTO 70
Ok
```

Nesse exemplo, o comando DRAW está no interior de um laço e por isso é repetido várias vezes. Não se preocupe em entender o que está por trás da linha 40. Ela apenas altera a posição em que o desenho, a ser feito pelo DRAW, irá começar.

Os códigos gráficos, utilizados pelo DRAW, são:

Código Sn = Define a escala em que será feito o desenho. O número *n* representa a quantidade unitária de pontos desenhados e pode estar entre 0 e 255. Seu valor inicial é 4.

Código An = Gira o sistema de coordenadas de 90º em 90º. O número *n* pode estar entre 0 e 3 e seu valor inicial é o 0 (sistema de coordenadas normal). Se *n*=1, o giro é de 90º; se n=2, o giro é de 180º e, se *n*=3, o giro é de 270º.

Código Cn = Define a cor com que o desenho será traçado. O número *n* pode estar entre 0 e 15 e seu valor inicial é 15 (cor branca).

Código M:x,:y = Desenha uma linha desde o último ponto *plotado* (posição atual) até a posição definida por *x* e *y*. O número *x* indica o incremento na coluna e pode estar entre 0 e 255 e o número *y* indica o incremento na linha e pode estar entre 0 e 191.

Código Mx,y = Desenha uma linha desde o último ponto *plotado* (posição atual) até o ponto cuja coluna é dada pelo número *x* (entre 0 e 255) e cuja linha é dada pelo número *y* (entre 0 e 191).

Código Un = Traça uma linha de comprimento *n* vezes maior que o valor unitário (definido por Sn) da posição atual para cima. O valor inicial de *n* é 1.

Código Dn = Traça uma linha de comprimento *n* vezes maior que o valor unitário (definido por Sn) da posição atual para baixo. O valor inicial de *n* é 1.

Código Rn = Traça uma linha de comprimento *n* vezes maior que o valor unitário (definido por Sn) da posição atual para direita. O valor inicial de *n* é 1.

Código Ln = Traça uma linha de comprimento *n* vezes maior que o valor unitário (definido por Sn) da posipão atual para esquerda. O valor inicial de *n* é 1.

Código En = Traça uma linha desde a posição atual até a posição situada *n* vezes o valor unitário acima de *n* vezes o valor unitário à direita dela. O valor inicial do número *n* é 1.

Código Fn = Traça uma linha desde a posição atual até a posição situada *n* vezes o valor unitário abaixo e *n* vezes o valor unitário à direita dela. O valor inicial do número *n* é 1.

Código Gn = Traça uma linha desde a posição atual até a posição situada *n* vezes o valor unitário abaixo e *n* vezes o valor unitário à esquerda dela. O valor inicial do número *n* é 1.

Código Hn = Traça uma linha desde a posição atual até a posição situada *n* vezes o valor unitário acima e *n* vezes o valor unitário à esquerda dela. O valor inicial do número *n* é 1.

A posição atual é redefinida após a execução de qualquer um desses códigos, com exceção de S, A e C. Antes dos códigos que alteram o valor da posição atual, pode-se utilizar um destes outros dois códigos: B ou N.

Código B = Faz com que a posição atual seja redefinida pelo código à sua frente sem que a linha seja desenhada.

Código N = Faz com que a linha especificada pelo código à sua frente seja desenhada sem que a posição atual seja redefinida.

Os códigos gráficos podem também ser inseridos numa variável *string* para depois serem utilizados por DRAW. Para isso basta utilizar o código X.

Código Xn$ = Faz com que os códigos gráficos contidos numa variável *string* (representada por n$) sejam executados.

Agora, tente fazer seus próprios programas usando DRAW. Depois de algumas tentativas você estará desenhando qualquer figura com facilidade.

FANTASMAS, DUENDES, MÁSCARAS, etc . . .

Além dos recursos já apresentados, o EXPERT possui ainda algo realmente fascinante: os *SPRITES*. Podemos imaginar os *sprites* como máscaras para serem colocadas sobre a tela. Essas máscaras podem ser movimentadas livremente e em diferentes profundidades. Pode-se usar simultaneamente 32 camadas distintas sobre a tela. Cada máscara é um *sprite* que pode ser colocado numa camada diferente, cada uma com um número entre 0 e 31. A camada 0 é a mais próxima do observador e a 31 é a mais distante. Numa paisagem, por exemplo, poderíamos representar uma árvore distante na camada 31, uma pessoa bem próxima na camada 0 e uma coruja voando para a árvore na camada 16 (figura 12.3).

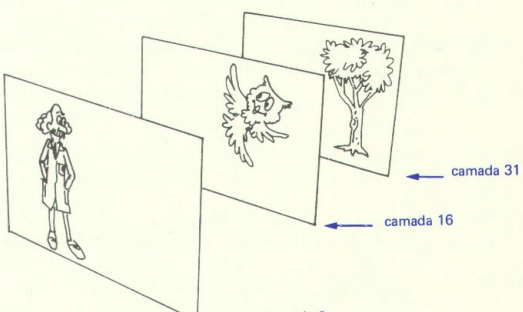


Figura 12.3

Para entender como os *sprites* devem ser usados, observe o programa a seguir. Ele cria 5 corujinhas e as movimenta na região central da tela.

Vamos analisá-lo rapidamente.

O padrão a ser desenhado na tela é definido pelo usuário nas linhas DATA (de 220 a 290). Os dígitos [1] representam o desenho e os dígitos [0] representam espaços vazios.

O laço entre as linhas 30 e 60 lê os valores contidos nas linhas DATA e os armazena numa variável *string* de nome S$.

A linha 70 define um *sprite*. A variável SPRITE$(1) é reservada para armazenar *sprites* e não deve ser usada para armazenar outro tipo de informação. O número dentro dos parênteses (no caso, o número é 1) é o número do *sprite* armazenado. Ele será necessário para imprimir o *sprite* na tela.

Das demais linhas, só nos interessam a 120, a 130, a 140, a 150 e a 160. Elas são as responsáveis pela impressão dos *sprites*. O comando básico em todas elas é o mesmo: PUT SPRITE. O número logo a seguir é o do plano de profundidade. Depois, dentro dos parênteses, estão as coordenadas onde será impresso o vértice superior esquerdo da figura (*sprite*). O próximo número é o código da cor com que a figura será feita e, finalmente, o último número é o número do *sprite* (no caso, 1).

Vamos ver o que produzirá a linha 120.

Ela vai imprimir um *sprite* no primeiro plano da tela (o mais próximo do observador), sendo que seu vértice esquerdo superior ficará na coluna F e na linha 85. Sua cor será verde (código 2) e seu número é 1.

A forma de se introduzir o *sprite* na variável reservada SPRITE$ não precisa ser necessariamente a que nós usamos para fazer as corujas, mas ele é o de visualização mais fácil.

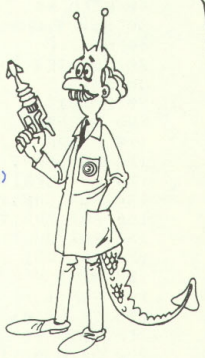
```
10 REM CORUJAS
20 SCREEN 2
30 FOR F=1 TO 8
40 READ K$
50 S$=S$+CHR$(VAL("&B"+K$))
60 NEXT F
70 SPRITE$(1)=S$
80 I=1
90 F=101
100 G=255-F
110 H=128
120 PUT SPRITE0,(F,85),2,1
130 PUT SPRITE1,(G,85),6,1
140 PUT SPRITE2,(H,85),10,1
150 PUT SPRITE3,(H,F-30),7,1
160 PUT SPRITE4,(H,G-30),13,1
170 FOR R=1 TO 20
180 NEXT R
190 F=F+I
200 IF F=101 OR F=155 THEN I=-I
210 GOTO 100
220 DATA 01111110
```

```
230 DATA 10000001
240 DATA 10100101
250 DATA 11011011
260 DATA 11111111
270 DATA 11000011
280 DATA 01100110
290 DATA 10100101
```

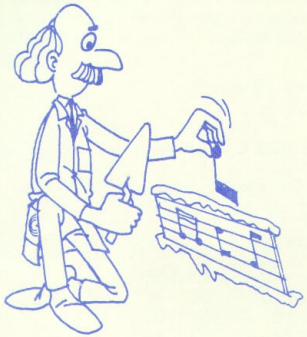
RECURSOS GRÁFICOS

A seguir, apresentamos um outro programa onde o *sprite* é introduzido de outro modo. Tente compreendê-lo.

```
10 REM UFOS
20 SCREEN 2,3
30 FOR V=0 TO 60
40 X=INT (RND(1)*255)
50 Y=INT (RND(1)*192)
60 CIRCLE (X,Y),0,15
70 NEXT V
80 FOR I=0 TO 31
90 READ A$
100 B$=B$+CHR$(VAL("&H"+A$))
110 NEXT
120 SPRITE$(0)=B$
130 X=100 : Y=100
140 S=INT (RND(1)*80)
150 D=INT(RND(1)*4)
160 IF D=0 THEN VX=0:VY=-1
170 IF D=1 THEN VX=1:VY=0
180 IF D=2 THEN VX=0:VY=1
190 IF D=3 THEN VX=-1:VY=0
200 FOR I=0 TO S
210 PUT SPRITE 0,(X,Y),10,0
220 X=X+VX : Y=Y+VY
```



```
230 IF X>240 OR X<0 THEN VX=-VX
240 IF Y>175 OR Y<0 THEN VY=-VY
250 NEXT I
260 GOTO 140
270 DATA 00,00,03,07,0D,1F,FF,AA
280 DATA FF,1F,0D,17,23,40,00,00
290 DATA 00,00,C0,E0,B0,F8,FF,55
300 DATA FF,F8,B0,E8,C4,02,00,00
310 GOTO 220
```



Capítulo 13

CONSTRUINDO SONS

O EXPERT possui um potente sintetizador musical facilmente utilizável através de comandos em BASIC. Este sintetizador pode gerar desde suaves melodias até ruídos e explosões.

O principal comando para produção de sons musicais é o PLAY. A forma de utilizá-lo é ilustrada pelo exemplo a seguir:

```
10 PLAY"CDEFGAB"
```

Esse comando reproduz a escala musical do Ocidente.

Aqueles que conhecem um pouco de notação musical já devem ter notado que o EXPERT utiliza o sistema cifrado, onde cada nota é representada por uma letra entre A e G (figura 13.1).

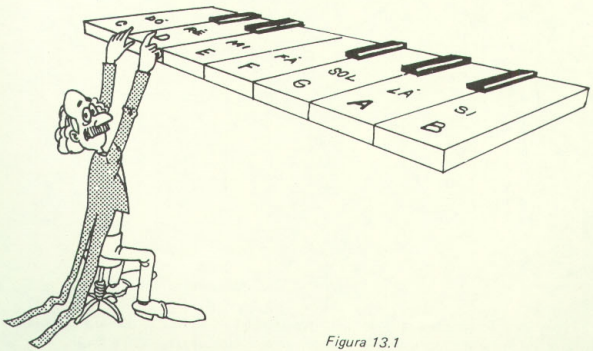


Figura 13.1

Podemos também produzir sustenidos e bemóis simplesmente acrescentando à frente da letra um destes sinais: + ou # (para sustenidos) e – (para bemóis).

Os dois comandos adiante geram uma seqüência de notas correspondentes às cinco teclas pretas do piano.

```
PLAY"C+D+F+G+A+"
PLAY"D-E-G-A-B-"
```

GRAVES, MÉDIOS E AGUDOS

Observe o programa a seguir. Ele utiliza muitos outros recursos do PLAY.

```
10 PLAY"O7L1AR32L8AR1","O7L1CR32L8CR1"
20 PLAY"T240L6V12","T240L2V9"
30 PLAY"R8O6GAB07DCCEB","O4GO5GE"
40 PLAY"DGF#GDO6BGAB","O4BO5EO4E"
50 PLAY"O7CDEDCO6BABG","O4ABO5C"
60 PLAY"F#GADF#AO7CO6BA","O5DF#D"
70 PLAY"BGABO7DCCED","GGC"
80 PLAY"DF#GDO6BGAB","O4BO5ED"
90 PLAY"EO7DCO6BAGDGF#G2","CC#DG"
```

Vamos aprender para que serve cada um deles.

A duração de cada nota pode ser especificada usando-se a letra L seguida de um número inteiro entre 1 e 64. A duração inicial dos sons gerados por PLAY corresponde a L4. O L8 na linha 10 faz com que a duração das notas passe a ser a metade da duração inicial. Veja na figura 13.2 a relação entre o número à frente de L e a duração.

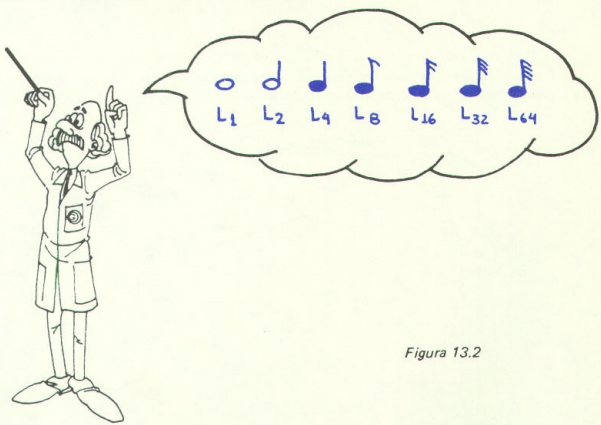


Figura 13.2

A letra T especifica o andamento (velocidade) da música. O número à frente do T pode ser qualquer inteiro entre 32 e 255 e o andamento inicial corresponde a T120.

A letra O define a oitava musical que será usada. O EXPERT dispõe de oito oitavas sendo a mais grave acessada por O1 e a mais aguda por O8.

Podemos também executar pausas (ausência de som) através da letra R e de um número entre 1 e 64. A correspondência entre os números e a duração das pausas é análoga à da duração das notas (definidas pela letra L).

O EXPERT possui três canais de som independentes e pode gerar acordes com até três notas. A vírgula, usada em todas as linhas do programa exemplo, faz com que sejam usados dois canais. O que está à esquerda dela sai através de um, e o que está à direita sai através de outro.

A letra V controla o volume com que o som é gerado. O número à frente dessa letra pode estar entre 0 e 15, sendo seu valor inicial igual a 8.

Esses são os principais recursos do PLAY. Existem ainda alguns outros que preferimos omitir para não confundi-lo, contudo, eles podem ser encontrados no livro LINGUAGEM BASIC MSX desta mesma editora.

Um outro comando BASIC relacionado com a geração de sons é o SOUND. Nós não entraremos em detalhes quanto ao seu funcionamento, entretanto, é interessante saber que através dele você pode gerar barulhos como o de um trem, de uma moto, de um helicóptero, de vento, do mar, etc....

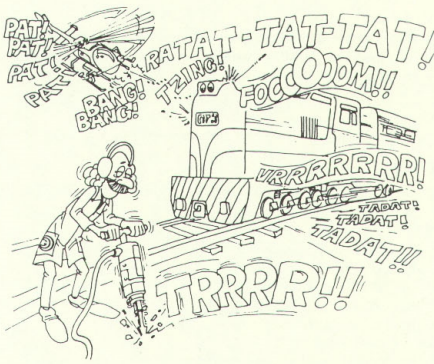


Figura 13.3

A título de exemplo apresentamos três programinhas que o utilizam. Estude-os e depois modifique-os.

```
10 FOR I=0 TO 13
20   READ S
30   SOUND I,S
40 NEXT I
50 DATA 0,0,0,0,24,0,22,3,2,2,16,90,2,12
60 DATA 0,0,0,0,0,0,21,247,16,0,0,100,60
,0
Ok
```

```
10 SOUND 8,16
20 SOUND 7,(1+2+4+16+32)
30 SOUND 6,5
40 SOUND 13,10
50 SOUND 12,180
60 SOUND 11,0
```

```
10 FOR A=0 TO 13
20 READ B
30 SOUND A,B
40 NEXT
50  DATA 0,0,0,0,0,0,20,247,16,0,0,100,
60,0
```

Para finalizar, vamos digitar um programa mais extenso (de Mussorgsky), onde são utilizados mais recursos, disponíveis no MSX. Vale a pena o esforço para encerrarmos estes primeiros passos com chave de ouro.

```
10 REM PROMENADE
20 CLS: PRINT "QUADROS EM EXPOSIÇÃO"
30 READ A$,B$,C$
40 IF A$="" THEN END
50 PLAY A$,B$,C$
60 GOTO 30
70 DATA V13,V10,V10
80 DATA O4L4GFA+O5C8F8D
90 DATA RRRRR
100 DATA RRRRR
110 DATA C8F8DO4B-O5CO4GF
120 DATA RRRRRR
130 DATA RRRRRR
140 DATA O4L4GFA+O5C8F8D
150 DATA O4L4DCDAA
160 DATA O3L4B-AB-O4CF
170 DATA C8F8DO4B-O5CO4GF
180 DATA O4L4AB-GGCC
190 DATA O4L4CFDEO3GA
200 DATA FGDF8G8C
210 DATA RRRRR
220 DATA RRRRR
230 DATA G8A8FO5FDCO4B-8F
240 DATA RRFFFF
```

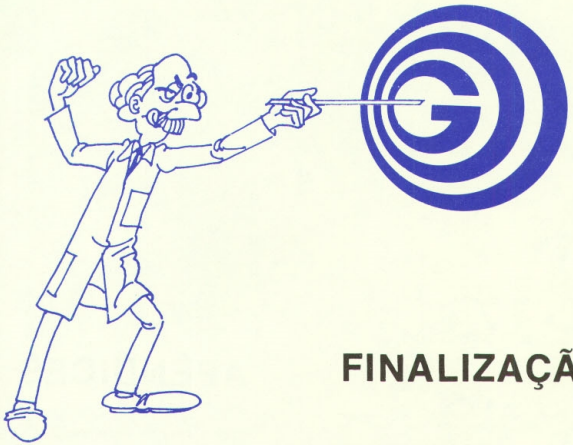
```
250 DATA RR03L4FB-GF
260 DATA 04L4FGDF8G8E-
270 DATA RRRRR
280 DATA RRRRR
290 DATA 04L4B-805C804A-05A-FE-8D-804A-
300 DATA RRA-A-A-A-
310 DATA RR03L4A-04D-03B-A-
320 DATA 04L4A-B-A-B-805C8E-804B-8A-
330 DATA 03L4A-B-A-B-804C8E-803B-8A-
340 DATA 02L4G-2FG-G-03A-
350 DATA 05L8D-E-FA-G-FE-G-FD-E-4
360 DATA 04L8A-05CD-04A-05E-D-C04G-05D-0
4D-05C4
370 DATA 03L8FE-D-4E-FA-4B-4A-4
380 DATA 04L4A-B-A-L8B-05CE-04B-
390 DATA 03L4A-B-A-L8B-04CE-03B-
400 DATA 02G-2L4FG-G-8R8
410 DATA 05L4CDCL8DFGDL4C
420 DATA 02B-2L4AB-B-03B-
430 DATA 02B-2L4AB-B-03B-
440 DATA 05L8FGA06C05B-AGB-AFG4
450 DATA 05L8CEFR8GFER8FR8E4
460 DATA 03L8AGF4GA04C4D4C4
470 DATA 05L4A8E8FADAD
480 DATA 05L4CDC04B-05C04B-
```

Cuidado! Não confunda
a letra `B` com o número `8`;
a letra `O` com o número `0`;
a letra `l` com o número `1`;
senão você obterá a mensagem:

```
Illegal function call in 50
```

```
490 DATA 03L4AB-FGFG
500 DATA 05L4F8C8DFDF8C8D
510 DATA 04AB-AB-AB-
520 DATA 03L4DGDGDG
530 DATA 05L4C04AB-05C04AB-805D8
540 DATA 04G8E8FFG8E8FF
550 DATA 03L4CFGCFG
560 DATA 05L4C04A05CFL8E-DC04B-
570 DATA 04GFGRL8B-R8AF
580 DATA 03L4C02F03C02FGA8B-8
590 DATA 05L4CDFG8B-8FG
600 DATA 04L4FF05CE-8R804FG
610 DATA 02L4AB-AGFG
620 DATA 05L4FL8E-DC04B-L405CDF
630 DATA 04L4FL8B-R8AFL4FF05C
640 DATA 02L4FGA8B-8AB-A
650 DATA 05L4G8B-8FGF04GF
660 DATA 05E-8R804FGF03GF
670 DATA 02L4GFGF03GF
680 DATA 05L4G8B-8FGF04GF
690 DATA 05E8R804FGFE-C
700 DATA 03L4C02FGF03E-F
710 DATA 04L4B-05C8F8DC8F8D04B-
720 DATA 04FA8R8B-A8R8B-F
```

```
730 DATA 03DC02B-AG03G
740 DATA 05L4C04GFGFB-
750 DATA 04GECDCD
760 DATA 03L4CEF02B-AG
770 DATA 05L4C8F8D04B-05E-C04B-
780 DATA 04A8R8B-G05C04AF
790 DATA 02L4FB-03GCFB-
800 DATA ""
810 DATA ""
820 DATA ""
```



FINALIZAÇÃO

Parabéns !!

Ao chegar a esta página, se você seguiu corretamente as instruções de cada capítulo, o BASIC MSX já não lhe é totalmente incompreensível e você deve estar em condições de arquitetar seus próprios programas.

Certamente há ainda muitos pontos não muito claros: alguns comandos do vasto BASIC MSX foram apenas citados rapidamente e outros foram omitidos para não sobrecarregá-lo de informações num primeiro contacto com o Expert. Todos os detalhes sobre os outros recursos desta poderosa linguagem podem ser encontrados no livro "LINGUAGEM BASIC MSX", listados em ordem alfabética, com exemplos e com informações sobre sua sintaxe.

No "COLEÇÃO DE PROGRAMAS PARA MSX Vol. I", você encontrará uma grande quantidade de programas que utilizam os recursos de som, cores e ação do MSX. O volume I foi concebido para principiantes e contém todas as informações sobre a concepção e construção dos programas.

No "COLEÇÃO DE PROGRAMAS PARA MSX Vol. II", você é levado a montar, passo a passo, um jogo de ação e aprenderá como construir programas aplicativos (Matemática Financeira, Estatística, etc) e utilitários para o seu MSX.

No "JOGOS DE HABILIDADE PARA MSX" estão listados jogos incríveis, a maioria dos quais usam a poderosíssima Linguagem de Máquina do Z-80 para explorar todos os recursos desta máquina maravilhosa.

Para os programadores mais experientes, aconselhamos o APROFUNDANDO-SE NO MSX, onde são discutidos com detalhes os usos dos vários elementos da arquitetura do MSX: PPI, ROM, RAM, VDP, PSG, variáveis do sistema, rotinas do BIOS, chaveamento de SLOT's, etc.

APÊNDICES

APÊNDICE I

OS ERROS MAIS FREQUENTES

Mesmo o programador mais experiente está sujeito a cometer erros de digitação ou de lógica. Conseqüentemente, se você está se iniciando no fascinante mundo da computação, não se preocupe com os muitos erros que, certamente, serão cometidos. A seguir, comentamos alguns erros mais freqüentes e o que fazer para evitá-los.

Erros de Lógica

Quando o computador se defronta com um erro de lógica contido num programa, ele pára sua execução e mostra na tela uma mensagem de erro e a linha em que ele parou. Comande LIST (ou SHIFT + F4) para ver o programa todo. Se você comandar LIST. (ou seja, LIST seguido de um ponto), apenas a linha em questão aparecerá.

Veja o que aconteceu e tente corrigir o problema. No apêndice IX você poderá encontrar um sumário das mensagens de erro e suas explicações. Isso vai lhe ajudar na identificação do engano ocorrido.

Não desanime na procura do problema: o tempo gasto para isso vale mais do que horas de aulas de computação e, portanto, não é desperdiçado.

Cuidado, porém, com uma coisa importante: nem sempre o erro está na linha em que foi acusado!

Por exemplo, no programa PROMENADE da página 76, aconteceu para muitos usuários uma interrupção na linha 50:

```
50 PLAY A$,B$,C$
```

Na realidade ela não tem nada de errado e não adianta tentar redigitá-la, pois o erro continuará!

O que devemos procurar é onde as variáveis A$, B$ e C$ foram definidas. Olhando o programa vemos que isso ocorre na linha 30, através do comando READ. Como ele utiliza as linhas DATA, é nelas que devemos procurar o erro!

Erros de Digitação

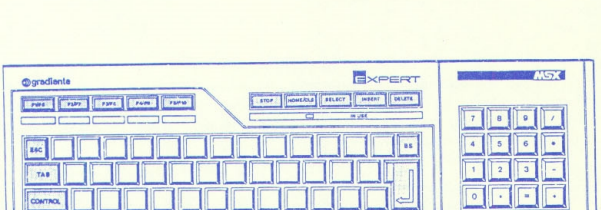
O erro mais comum, especialmente no caso de programadores principiantes, é o de esquecer a tecla RETURN (↵) no fim de cada linha lógica, antes da introdução da linha seguinte.

Outro erro é a confusão dos símbolos:

O (letra maiúscula) com 0 (número zero)
B (letra maiúscula) com 8 (número oito)
1 (número um) com ¡ (letra minúscula)
l (letra minúscula) com I (letra maiúscula)
l (letra minúscula) com 1 (número um)
' (apóstrofo) com / (barra de divisão)
\ (divisão inteira) com \ (caractere gráfico)

APÊNDICE II

AS TECLAS ESPECIAIS



SHIFT

É uma tecla que não funciona sozinha. Na verdade, ela é utilizada para que possamos obter letras maiúsculas ou outros caracteres, como: !, ", $, %, +, (,), &, <, >, etc... Ela é muito parecida com a tecla de maiúsculas de uma máquina de escrever.

CAPS LOCK

Pressionando esta tecla, você deverá notar que as letras, na sua tela, mudam de tamanho. Acionando-a pela primeira vez, as letras passam a ser escritas em maiúsculas. Se você tornar a pressioná-las, as letras passam novamente a minúsculas. Funciona como a trava de maiúsculas de uma máquina de escrever

LGRA

Esta tecla é usada para escolher os símbolos gráficos que correspondem a cada tecla. Se você pressionar LGRA junto com uma tecla qualquer, o símbolo a ela correspondente, aparecerá na tela.

CONTROL

A tecla CONTROL (às vezes abreviada para CTRL) é usada para acessar funções especiais de algumas teclas, como CONTROL + STOP por exemplo.

INSERT

Esta tecla será usada quando você desejar acrescentar caracteres dentro de uma linha. Mova o cursor para a posição onde você deseja acrescentar algo, pressione INSERT, e o texto que você digitar, será inserido. Ela é desativada por algumas outras teclas, ou pressionando-a novamente.

DELETE

Pressione DELETE para eliminar qualquer caractere sobre o qual esteja o cursor.

BS

Sempre que você errar, ao escrever, use a tecla BS (BACK SPACE). Por exemplo: você, acidentalmente, escreveu "msx" ao invés de "MSX". Basta, então, pressionar BS três vezes e reescrever as letras corretas. Portanto, BS permite corrigir o(s) caracter(es) errado(s) escrito(s) à esquerda do cursor.

TAB

Esta tecla tem por finalidade mover o cursor para uma posição específica na linha, que corresponde ao início do próximo grupo de oito colunas. Esta tecla é parecida com o tabulador de uma máquina de escrever.

HOME/CLS

Pressionando esta tecla o cursor se posicionará no canto superior esquerdo da tela, pois você estará usando a função HOME. Caso você pressione esta tecla junto com a tecla SHIFT, o cursor irá posicionar-se no canto superior esquerdo após ter limpado a tela, pois assim você estará usando a função CLS (CLear Screen).

STOP

Esta tecla interrompe e retoma a execução de qualquer comando, instrução ou programa. Para que, após a interrupção, o comando retorne ao usuário, deve-se pressionar CONTROL + STOP (simultaneamente).

RETURN

Esta tecla introduz o que foi digitado na tela para dentro do computador, isto é, para sua memória.

F1 . . . F10

Estas teclas são as Teclas Funcionais e em cada uma delas está marcada uma letra F (de FUNCTION). Elas são um dispositivo armazenador, pois permitem o acesso a um comando pressionando-se somente uma ou duas teclas, ao invés de escrever letra por letra. As funções das teclas F1 até F5 são acionadas diretamente através da simples digitação de cada uma delas. Já as funções das teclas de F6 até F10 são acionadas com a digitação simultânea da tecla SHIFT. Ao ser ligado, o EXPERT contém as seguintes funções nessas teclas:

F1 COLOR : Este comando é usado para definir as cores da borda, do fundo e do primeiro plano da tela. Necessita de argumentos e de RETURN.

F2 AUTO : É usado para gerar automaticamente a numeração das linhas de um programa durante a sua digitação. Necessita de RETURN.

F3 GOTO : GOTO permite um desvio incondicional na execução de um programa. Necessita de argumentos e de RETURN.

F4 LIST : Este comando instrui seu MSX para listar na tela, todas as linhas de um programa. Necessita de RETURN.

F5 RUN : O RUN faz com que o programa que está na memória do micro seja executado. Não necessita de RETURN.

F6 COLOR 15,1,1 : Esta tecla faz com que a borda e o fundo da tela sejam pretos e as letras brancas. Não precisa de RETURN.

AS TECLAS ESPECIAIS

F7 CLOAD : CLOAD carrega, para memória do micro, dados gravados em fita cassete por um comando CSAVE.

F8 CONT : Este comando ordena ao computador, que CONTinue a executar qualquer processamento que tenha sido interrompido.

F9 LIST : Idem a F4, porém sem necessidade de RETURN.

F10 RUN : Idem a F5, porém com necessidade do RETURN.

O MSX normalmente expõe as teclas de funções entre F1 e F5 na última linha e sempre que você pressionar a tecla SHIFT as funções das teclas entre F6 e F10 serão expostas. Qualquer destas funções pré-definidas podem ser rapidamente reprogramadas para sua própria conveniência, substituindo-as por funções que você use mais freqüentemente. O comando para redefinir a função de uma dessas teclas é: KEY *número,"função"*. Através do *número* você define qual a tecla (de 1 a 10) que será reprogramada e a *função* é o que ela passará a conter.

SELECT

Esta tecla não é usada para programação BASIC e serve somente para o acesso a programas de processamento de texto, ou na entrada de dados de um pacote de software.

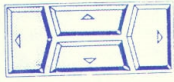
ESC

A tecla ESC (de ESCape) muitas vezes é usada em aplicações de software. Ela é uma função usual que interrompe a operação do programa, ou continua a operação seguinte à que havia sido interrompida.

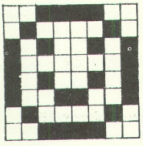
RGRA

A tecla RGRA é usada para selecionar outros caracteres que correspondem às teclas. Se você pressionar RGRA e, enquanto a mantém para baixo, teclar uma letra qualquer, o caracter diferente, correspondente a esta tecla, será mostrado na tela.

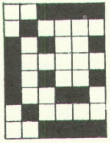
SETAS



Movem o cursor para qualquer posição do vídeo.



SCREEN 1



SCREEN 0

figura II.1

Para que os caracteres gráficos dos teclados especiais apareçam por inteiro no vídeo, é necessário que se esteja trabalhando com a SCREEN 1. Na SCREEN 0 eles aparecem "cortados" do lado direito.

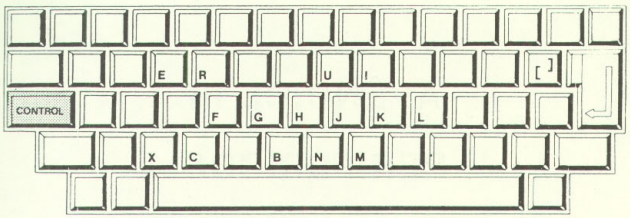


figura II.2

Além destas teclas, o EXPERT possui algumas outras que se tornam especiais quando digitadas simultaneamente com a tecla CONTROL. Veja na tabela abaixo a correspondência entre essas teclas e suas funções.

Combinações de Teclas	FUNÇÃO	Teclas Especiais
CONTROL + B	Move o cursor até o começo da palavra anterior	
CONTROL + C	Dá um BREAK na espera do INPUT	
CONTROL + E	Apaga do cursor ao fim da linha BASIC	
CONTROL + F	Move o cursor até o começo da palavra seguinte	
CONTROL + G	Dá um "bip"	
CONTROL + H	Volta o cursor, apagando	BS
CONTROL + I	Move o cursor até a próxima tabulação	TAB
CONTROL + J	Coloca o cursor na próxima linha (line feed)	
CONTROL + K	Move o cursor até o canto superior esquerdo	HOME
CONTROL + L	Limpa a tela colocando o cursor em HOME	CLS
CONTROL + M	Insere a linha BASIC na memória	
CONTROL + N	Leva o cursor ao fim da linha	
CONTROL + R	Põe o cursor no modo inserção	INSERT
CONTROL + U	Apaga a linha inteira	
CONTROL + X		SELECT
CONTROL + [		ESC

Pressionando simultaneamente as teclas CONTROL + SHIFT + STOP, obtemos um RESET por software, ou seja, o Expert reinicializa as variáveis do sistema, deixando-as no estado em que se encontravam ao ligar o micro.

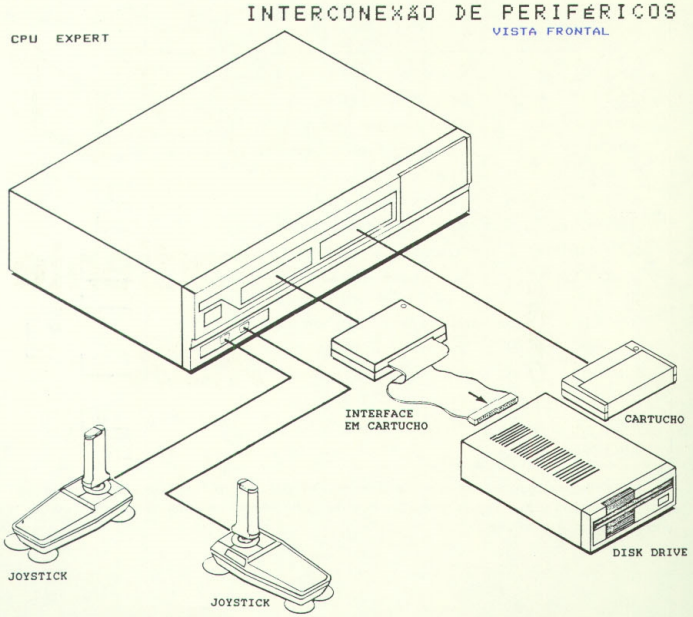
APÊNDICE III

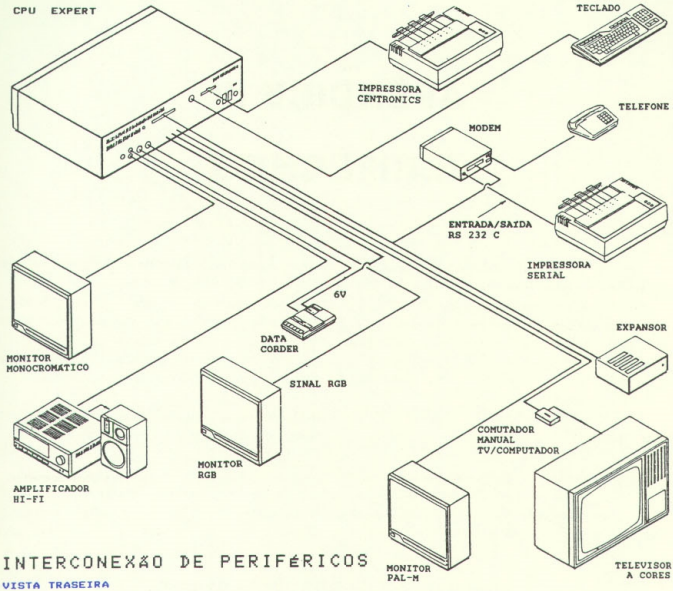
PERIFÉRICOS

O EXPERT pode ser ligado a muitos outros aparelhos, os periféricos, controlando-os ou trocando dados com eles.

A versatilidade do padrão MSX é muito grande e permite o crescimento do sistema a partir das unidades fundamentais: o teclado e o console.

Algumas das possibilidades de conexão de periféricos estão indicadas nas figuras III.1 (frontal) e III.2 (posterior).





INTERCONEXÃO DE PERIFÉRICOS

VISTA TRASEIRA

gradiente

MSX

APÊNDICE IV

CARTUCHOS

Na parte frontal do EXPERT existem dois encaixes (*SLOTS*) onde podem ser inseridos cartuchos. Os cartuchos podem ser do tipo: RAM, ROM, INTERFACE, e outros.

CARTUCHO RAM . . . . . .expansão de memória a ser gravada pelo usuário.

CARTUCHO ROM . . . . . .memória previamente gravada na fábrica, com algum programa. Não pode ser alterada.

CARTUCHO INTERFACE. . . . . .utilizado para interligar periféricos que necessitam de interface (por exemplo, um *DISK-DRIVE*).

Geralmente, cada cartucho é acompanhado de um pequeno manual com especificações de seu funcionamento. É recomendável a leitura prévia desses manuais para uma perfeita utilização das funções de cada cartucho.

CUIDADOS AO INSERIR UM CARTUCHO

1. Quando você for inserir ou retirar um cartucho, desligue o micro.

2. Após a leitura do manual do cartucho, verifique o lado correto para a inserção e o *SLOT* a ser utilizado. Salvo especificações em contrário, qualquer *SLOT* pode ser usado.

3. Verificar se o cartucho está conectado firmemente e, depois disto, ligar o micro.

O cartucho é um componente de precisão, portanto, evite abrí-lo, tocar com os dedos ou molhar a parte metálica de encaixe, pois qualquer descuido poderá causar defeitos.

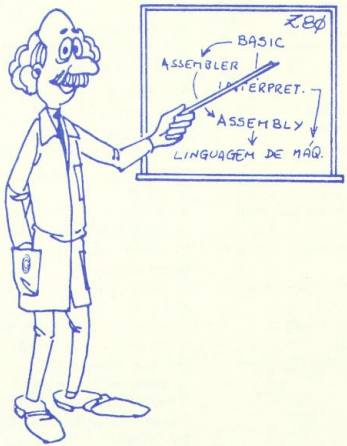
APÊNDICE V

ESPECIFICAÇÕES TÉCNICAS

Microprocessador		Z80A (3,58MHz)
Memória residente	ROM do sistema	32 Kbytes (BASIC-MSX)
	RAM	80K Bytes - 64K Bytes usuário - 16K Bytes vídeo
Linguagem de Programação		BASIC-MSX e Linguagem de Máquina Z80
Vídeo	CDP	TMS-9128NL - Processador de Vídeo
	Texto 1	32 colunas x 24 linhas (condição inicial 29 colunas)
	Texto 2	40 colunas x 24 linhas (condição inicial 40 colunas)
	Gráficos	256 x 192 pontos (horizontal x vertical)
	Sprite	32 níveis
	Cor	16 cores
Som	PSG	AY-3-8910-A - Sintetizador de som programável
	Características	08 oitavas, 03 vozes, canal adicional de ruído
Teclado	Disposição das teclas	89 teclas para pontuação, acentuação, caracteres do Português, gráficos, 10 funções programáveis
	Modalidades	06 modalidades (níveis), 256 endereçamentos de caracteres residentes
Interface residente	Monitor de Som	Alto falante interno com controle de volume externo
	Gravador cassete/ Data Corder	Processo FSK 1200/2400 baud
	Impressora	Padrão tipo Centronics
Conexões externas	Comunicações	Padrão RS232 - Modem padrão Telebrás - Videotexto (opcionais)
	Interface/Cartuchos	02 slots frontais p/ cartuchos padrão MSX (memória virtual, interfaces, aplicativos, games, etc.)
	Barramento de expansão	01 conector, 50 pinos para expandir o sistema, conectar periféricos, etc.
	Vídeo Monocromático	01 conector tipo RCA p/ monitor de vídeo (1,2 VPP/ 75 ohms)
	RGB Analógico	01 conector circular 08 vias (sinal RGB 1 VPP/75 ohms; sinal sincronismo 0,7 VPP/75)
	TV/Monitor a cores	Conector RCA VIDEO COLOR - Saída p/ monitor de vídeo colorido PAL-M Conector RCA VIDEO RF-OUT - Saída p/ antena de TV, através de cabo RF ligado a comutador manual TV/COMPUTADOR. Chave p/ seleção do canal (3 ou 4)

ESPECIFICAÇÕES TÉCNICAS

	Saída som	01 conector tipo RCA - saída misturador (03 canais) interno
	Gravador cassete/data corder	01 conectador circular 05 vias sinais + controle
	Entrada-saída de controles/joystick	02 conectores frontais tipo D9 (09 pinos),01 p/ cada controlador e/ou joystick
	Impressora	01 conector tipo finger duplo 26 pinos
	Tomadas suplementares	02 tomadas suplementares p/ alimentação de periféricos (controladas pela chave liga/desliga da CPU) carga máx. 100 VA p/ tomada. Conector e cabo alimentador de 6 V p/ DATA CORDER.
Alimentação	Tensão da rede	AC 120V/240V 60 Hz (chave p/ comutação da tensão da rede)
	Consumo (somente CPU)	Máximo 30 va
	Consumo (c/ todos periféricos)	Máximo 42 VA
Acessórios	Cabos	01 cabo RF 75 ohms tipo RCA-RCA 01 cabo 05 vias - P1, P2, P2 (REM, AUX, EAR) 01 cabo P4/P4 - Alimentação do DATA CORDER
	Etiquetas	01 conj. de etiquetas de funções p/ titulação pelo usuáio das funções programáveis
	Fusíveis suplementares	02 fusíveis: 01 p/ 120V. 01 p/ 240V
	Livros	02 livros: 01 DOMINANDO O EXPERT, 01 LINGUAGEM BASIC MSX. 01 Resumo de Operações do EXPERT.



APÊNDICE VI

MAPA DA MEMÓRIA

O EXPERT possui 32 Kbytes de memória ROM (pré-gravada na fábrica, apenas para leitura) e 80 Kbytes de memória RAM (para escrita e leitura). O microprocessador Z80-A pode acessar diretamente 64 Kbytes de memória RAM e é em parte dessa área que o usuário pode trabalhar. Outros 16 Kbytes (VRAM) são acessados pelo microprocessador TMS-9128NL, específico para controlar o vídeo.

Tanto a ROM como a RAM são divididas em páginas para agilizar o acesso aos seus endereços. Cada página de memória tem 16 Kbytes.

A numeração dos endereços da ROM é sempre a mesma, independentemente de se estar ou não usando expansões, interfaces ou cartuchos. Ela se sobrepõe à parte da numeração da RAM e é por isso que apenas 28,8 Kbytes desta, podem ser acessados diretamente pelo BASIC.

O MSX possui quatro *slots,* sendo dois internos (0 e 2) e dois externos (1 e 3). Os dois *slots* internos são ocupados pela ROM (*slot* 0) e pela RAM (*slot* 2). Os *slots* externos podem ser conectados a cartuchos, interfaces, etc...

O *slot* 1 tem prioridade sobre o *slot* 3, isto é, se existir um cartucho ligado em cada um deles, prevalecerá a operação do que está no *slot* 1. A conexão do *slot* 1 é a marcada com CARTRIDGE A. O *slot* 3 possui duas conexões: uma dianteira marcada com CARTRIDGE B e outra traseira marcada com BUS EXPANSION. Salvo indicação em contrário, eles não devem ser utilizados simultaneamente.

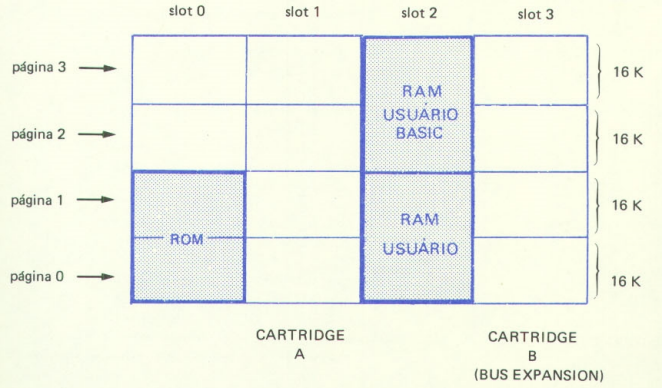


figura VI.1 (mapa da memória)

MAPA DA MEMÓRIA

A RAM acessível pelo usuário que programa em BASIC é constituída pelas duas páginas superiores do *slot* 2. As duas páginas inferiores podem ser acessadas mudando-se o status do PPI (veja APROFUNDANDO-SE NO EXPERT, desta mesma editora, para maiores detalhes).

O esquema de distribuição da RAM (usuário BASIC) está indicado na figura VI.2.

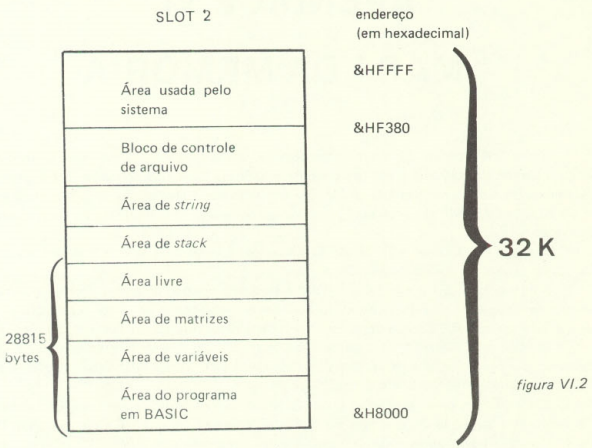


figura VI.2

A seguir, uma explicação rápida sobre cada uma das regiões indicadas na fig. VI.2:

ÁREA DO PROGRAMA: área onde serão armazenados programas com seus respectivos números de linha.

ÁREA DE VARIÁVEIS: armazena dados numéricos e os *"pointers"* para os dados alfanuméricos.

ÁREA LIVRE: é a área não utilizada.

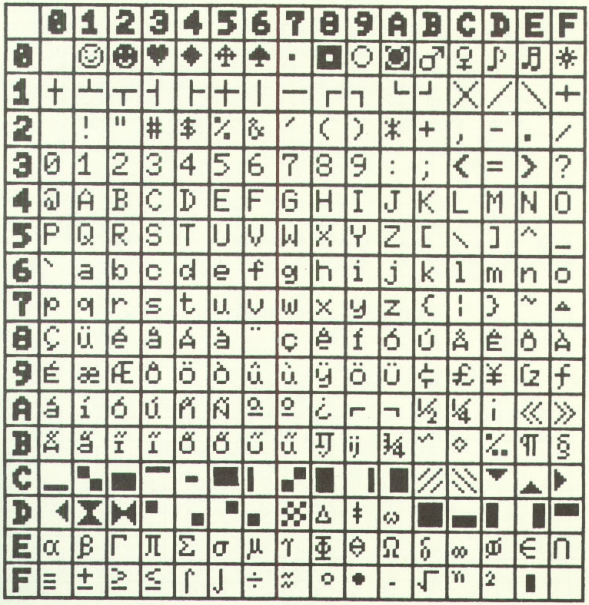
ÁREA DE STACK: área utilizada para guardar endereços de retorno quando executadas as instruções FOR-NEXT ou GOSUB-RETURN.

ÁREA DE STRING: área utilizada para armazenar strings que englobem variáveis alfabéticas e conjunto de variáveis. O tamanho é definido pela instrução CLEAR. Caso não haja definição por tal instrução, será reservada uma área de 200 bytes.

BLOCO DE CONTROLE DE ARQUIVOS: área utilizada para entrada/saída de arquivos. O tamanho é determinado pela instrução MAXFILES.

APÊNDICE VII
CARACTERES ASCII

A tabela a seguir fornece o código hexadecimal dos 256 caracteres armazenados na memória ROM do seu EXPERT. Para obter um determinado código você deve ler primeiro a linha e depois a coluna. Por exemplo, o caractere L corresponde ao código ASCII 4C (em hexadecimal).



APÊNDICE VIII
IMPRESSORA

Exitem duas tabelas de caracteres que podem ser selecionadas ao se usar uma impressora: a da ABNT (que já está ativada ao ligarmos o Expert) e a do MSX. Para selecioná-las, basta alterar o conteúdo da posição &HF417 da memória ou o último atributo do comando SCREEN, conforme o quadro a seguir:

Padrão ABNT	Padrão MSX
POKE &HF417,0	POKE &HF417,1
SCREEN ,,,,0	SCREEN ,,,,1

Na tabela de conversão MSX-ABNT a seguir estão relacionados os códigos (em hexadecimal) dos caracteres listados. Para qualquer caractere de código maior que 127 (&H7F) que não conste desta tabela, será enviado para a impressora um espaço em branco, de código 32 (&H20).

TABELA DE CONVERSÃO MSX – ABNT

MSX	ABNT		MSX	ABNT		MSX	ABNT		MSX	ABNT	
AD	A1	¡	AC	BC	¼	89	CD	í	B1	E3	ã
BF	A7	º	AB	BD	½	A5	D1	Ñ	87	E7	ç
A6	AA	ª	BA	BE	¾	8A	D3	ó	82	E9	é
AE	AB	«	A8	BF	¿	8E	D4	ô	88	EA	ê
F8	B0	°	8F	C0	À	B4	D5	õ	A1	ED	í
F1	B1	±	84	C1	Á	8B	DA	ú	A4	F1	ñ
FD	B2	²	8C	C2	Â	9A	DC	ü	A2	F3	ó
E6	B5	µ	B0	C3	Ã	E1	DF	ß	93	F4	ô
BE	B6	¶	80	C7	Ç	85	E0	à	B5	F5	õ
A7	BA	º	90	C9	É	A0	E1	á	A3	FA	ú
AF	BB	»	8D	CA	ê	83	E2	â	81	FC	ü

APÊNDICE IX

CÓDIGOS DE ERROS

Sumário dos códigos de erros e suas mensagens.

CÓDIGO	MENSAGEM
1	NEXT without FOR

NEXT sem FOR

– Comando NEXT utilizado sem uma instrução FOR correspondente.
– Este erro também pode ocorrer se a instrução NEXT (variável) for invertida em um laço múltiplo.

2	Syntax error

Erro de sintaxe

– Ocorre quando uma palavra do vocabulário BASIC é escrita de forma incorreta ou quando se faz uso indevido da pontuação.

3	RETURN without GOSUB

RETURN sem GOSUB

– Uma instrução RETURN foi encontrada sem que um GOSUB fosse executado antes.

4	Out of DATA

Insuficiência de dados

– Uma instrução READ foi executada sem encontrar nenhum dado para ler.
– A instrução DATA pode ter sido omitida ou todos os dados já foram lidos. Neste último caso, RESTORE resolve o problema.

5	Illegal function call

Função ilegal

– Tentativa de executar uma operação usando um parâmetro ilegal.

– Um erro FC (illegal Function Call) também pode ocorrer como resultado de:
1. Dimensão negativa para matriz.
2. Argumento negativo ou nulo em logarítmos (LOG).
3. Raíz quadrada de um número negativo.
4. Argumento inconveniente para MID$, LEFT$, RIGHT$, INP, OUT, PEEK, POKE, TAB, SPC, STRING$, SPACE$, INSTR$, ou ON . . . . . GOTO.

---

6 Overflow

Sobrecarga

– A magnitude de um número é muito grande para o computador.

---

7 Out of memory

Insuficiência de memória

– Toda memória disponível foi utilizada ou reservada.
– O programa é muito grande, tem arquivos demais, possui muitos laços, muitas variáveis ou expressões muito complexas.

---

8 Undefined line number

Número de linha não definido

– Uma linha contendo um GOTO, GOSUB, IF . . . THEN . . . ELSE, está se referindo a outra linha que é inexistente.

---

9 Subscript out of range

Subtexto fora de faixa

– Tentativa de usar um elemento que está se referindo a um:
* subtexto que está fora de suas dimensões, ou
* número errado de subtextos.

---

10 Redimensioned array

Redimensionamento

– Dois termos DIM foram usados para dimensionar a mesma matriz, ou
– O termo DIM, ultrapassou a dimensão 10, que é estabelecida pelo computador.

---

11 Division by zero

Divisão por zero

– Uma divisão por zero, foi encontrada em uma expressão, ou

– O resultado da entidade zero produziu um domínio negativo.

12 Illegal direct

Modo direto usado ilegalmente

– Um comando que é usado no modo indireto foi introduzido no modo direto.

13 Type mismatch

Atribuição ilegal

– Para um nome de uma variável *string* foi atribuído um valor, ou vice-versa.
– Para uma função, a qual aguarda um argumento numérico, foi dado um argumento *string*, ou vice-versa.

14 Out of string space

Fora do espaço *string*

– O espaço reservado para as variáveis *strings* foi excedido na memória.

15 String too long

String **muito longa**

– Tentativa de criar uma *string* com mais de 255 caracteres.

16 String formula too complex

Fórmula de *string* **muito complexa**

– Uma expressão *string* está muito longa ou muito complexa. A expressão deve ser dividida em expressões menores.

17 Can't continue

Impossibilidade de continuar

– Foi feita uma tentativa de continuar um programa, o qual:
1. Parou devido a um erro;
2. Tenta ser modificado durante uma parada na execução;
3. Ou, simplesmente não existe continuação.

18 Undefined user function

Uso de função não definida

– A função FN foi chamada antes de ser definida pela instrução DEF FN.

19 Device I/O error

Erro no dispositivo de I/O

– Ocorreu um erro de I/O (Input/Output) no cassete, na impressora ou na operação de CRT; ou seja, há um erro na entrada ou saída de dados. Este erro é fatal e isso significa que o BASIC não pode encontrá-lo.

20 Verify error

Erro de verificação

– O programa que está sendo executado não está coerente com o programa gravado no cassete.

21 No RESUME

Falta RESUME

– Um erro encontrado na rotina foi introduzido porém não contém o termo RESUME.

22 RESUME without error

RESUME sem erro

– Um termo RESUME foi encontrado antes de um erro colocado na rotina.

23 Unprintable error

Erro não imprimível

– Não há mensagem de erro disponível para a condição existente. Isto geralmente acontece quando uma mensagem ERROR aparece, para um código de erro inválido ou indefinido.

24 Missing operand

Falta de operando

– Tentativa de operação sem o fornecimento de um dos operandos necessários.

25 Line buffer overflow

Linha do *buffer* em *overflow*

– Uma linha introduzida possui caracteres demais.

26-49 Unprintable errors

Erros não imprimíveis

– Estes códigos não tem definição. Devem ser reservados para futuras expansões do BASIC.

50 FIELD overflow

***Overflow* em FIELD**

– Tentativa de inserir um FIELD num espaço muito pequeno de *bytes*, que foi especificado no *length* do registro contido no termo OPEN de um arquivo aleatório, ou
– O fim de um *buffer* FIELD foi encontrado no Modo Seqüencial de I/O, sendo que deveria ser para um arquivo aleatório.

51 Internal error

Erro interno

– Mau funcionamento interno.
(Seu computador está com problemas!)

52 Bad file number

Problemas com número de arquivo

– Há um termo, ou comando, referente a um número de arquivo, o qual não está aberto (OPEN). Ou então, ele está fora do alcance dos números de arquivo especificados pelo termo MAXFILE.

53 File not found

Arquivo não encontrado

– Há um termo LOAD, KILL ou OPEN referente a um arquivo, o qual não existe na memória.

54 File already open

Arquivo já aberto

– Uma saída seqüencial, no modo OPEN, foi o resultado de arquivo que já havia sido aberto ou, então, um termo KILL abrindo o mesmo tipo de arquivo.

55 INPUT past end

INPUT depois do final

– Um termo INPUT foi executado depois de todos os dados de um arquivo (ou arquivo nulo) serem lidos. Evita-se este erro usando a função EOF para encontrar o final do arquivo.

56 Bad file name

Nome incorreto para arquivo

– Uma forma ilegal foi usada para o nome do arquivo com: LOAD, SAVE, KILL, NAME, etc.

57 Direct statement in file

Termo no modo direto foi encontrado no arquivo

– Um termo direto foi encontrado carregando um arquivo no formato ASCII. O LOAD foi terminado.

58 Sequential I/O only

Somente entrada e saída seqüencial

– Um termo de acesso randômico está distribuindo o arquivo de forma seqüencial.

59 File not found

Arquivo não aberto

– O arquivo especificado em um termo PRINT#, INPUT#, etc., não havia sido aberto.

60-255 Unprintable error

Erro não imprimível

– Estes códigos não possuem definição.
Costuma-se usá-los para definir códigos pessoais de erro.

APÊNDICE X
GLOSSÁRIO

Neste glossário você irá encontrar algumas palavras de uso corrente entre os programadores no Brasil. Muitas delas são de origem inglesa, outras são termos técnicos ou de uso restrito a uma certa área de atividade e nem sempre podem ser encontradas em dicionários.

A maioria dessas palavras foram usadas com intuito de familiarizar o leitor com termos que ele irá encontrar freqüentemente em textos ligados à área da informática. Apesar de suscintas, as explicações são suficientes para uma primeira leitura.

ÁLGEBRA BOOLEANA — Álgebra usada para matematizar operações lógicas. Foi formalizada pela primeira vez pelo matemático inglês George Boole.

ALFANUMÉRICO — Termo genérico que designa todos os caracteres que representam letras ou números.

ALEATÓRIO — Ao acaso. Todo fenômeno que não tem suas causas determináveis. Ver RANDÔMICO.

ARRAY — É uma variável indexada (matriz ou tabela).

ASCII — *American Standard Code for Interchange Information.* É um código padrão utilizado na maioria dos microcomputadores.

ASSEMBLER — Programa que permite a programação em linguagem *Assembly*.

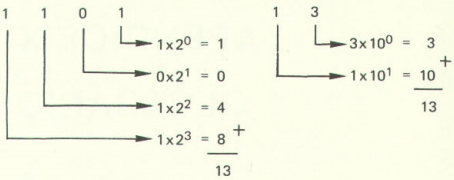
ASSEMBLY — É a linguagem mais fundamental depois da própria linguagem da máquina. É extremamente complexa a programação direta com códigos de máquina e para evitá-la associou-se um código mnemônico à cada instrução de máquina. O *Assembly* é o conjunto desses códigos mnemônicos.

BASIC — É uma das mais difundidas linguagens de programação (além de ser uma das mais simples!). Existem diversas versões do BASIC sendo que uma das mais completas é a usada pelo padrão MSX.

BAUD — Unidade de velocidade de transmissão de informação. A menor unidade de informação possível é o bit. Um baud é um bit por segundo. Por exemplo, se um sistema transmite dados com a velocidade de 100 bauds, significa que ele transmite 100 bits em cada segundo.

GLOSSÁRIO

BINÁRIO — Sistema de numeração que usa apenas os algarismos 0 e 1. Por exemplo, o número 13 escrito no sistema binário será 1101.



BIT — É a unidade mais fundamental de informação.

BUFFER — Área de memória onde são temporariamente armazenados dados.

BUG — Termo que designa um problema num programa. Poderia ser traduzido como "grilo", ou ainda "galho".

BUS — O mesmo que transmissor. Emprega-se esse termo geralmente para designar um conjunto de terminais que servem para transmissão ou recepção de informações. Barra de transmissão ou recepção de informação.

BYTE — É uma unidade lógica de informação. Inicialmente designava um conjunto arbitrário de bits, porém, devido à padronização imposta pelo uso, designa um conjunto de 8 (oito) bits.

CARREGAR — Transferir informações de uma memória (geralmente externa: gravador, *floppy-disk*, etc...) para a memória RAM.

CHIP — É um componente eletrônico no interior do qual existem circuitos complexos destinados à execução de operações específicas.

CLOCK — É o gerador da freqüência de operação do microprocessador, ou seja, do número de instruções elementares que ele pode realizar em cada segundo.

COMANDO — Ordem a ser executada imediatamente.

COMPILADOR — Programa que traduz conjuntos completos de instruções, de uma linguagem para outra.

CPU — Unidade Central de Processamento. É a parte mais fundamental de um computador. Nos micros geralmente ela é constituída por um único *chip*. É a CPU quem gerencia todo o funcionamento do computador, distribuindo tarefas às partes secundárias.

CRT — *Catode Ray Tube* — Tubo de raios catódicos ou TV. É a designação da tela em modo texto como dispositivo.

CURSOR — Indicador da posição em que será escrito o próximo caractere digitado.

DATA — Dados (em latim). É a instrução BASIC que permite armazenar dados em linhas de programas.

DATA BUS — Barra de endereçamento de dados.

DEBUG — Eliminação de erros de um programa.

DELETAR — Eliminar caracteres à direita do cursor ou linhas de um programa.

DISK — Disco onde são gravadas informações através de processos eletromagnéticos.

DOS — *Disk Operation Sistem.* Sistema de operação de discos.

DRIVER — É um termo usado com vários significados. Pode ser imaginado como o gerente de uma certa atividade. Por exemplo, DISK DRIVER é o controlador (gerente) de discos flexíveis.

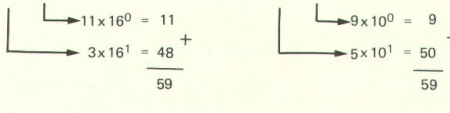
DUMP — Termo que designa a verificação e apresentação através de um meio de saída do conteúdo de todas as variáveis ou arquivos de um programa.

EDITOR — Rotina destinada a facilitar a escrita numa linguagem e eventuais correções da mesma.

FILE — Arquivo. Informações armazenadas em memória externa auxiliar (geralmente, *floppy-disk*).

HARDWARE — É a parte eletrônica de um sistema. Pode ser entendido como arquitetura de construção da máquina.

HEXADECIMAL — Sistema de numeração que usa dezesseis dígitos para representar os números. Os dígitos são: 0,,1,2,3,4,5,6,7,8,9,A,B,C,D,E e F. O número decimal 59 equivale a 3B em hexadecimal.



INPUT — É a entrada de um sistema.

INTERFACE — Circuito que permite a interligação e comunicação entre dois elementos de um sistema.

INTERPRETADOR — Programa que traduz instrução por instrução de uma linguagem para outra.

INSTRUÇÃO — É uma ordem para ser executada num determinado momento dentro de uma dada seqüência.

I/O — Input/Output. Entrada e saída de um sistema.

JOYSTICK — Alavanca de controle manual geralmente utilizada em jogos de ação.

KBYTE — 1024 bytes.

LOCAÇÃO — Posição provisória de um programa ou de dados na memória do computador.

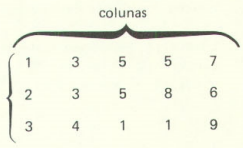
LOAD — Comando que carrega informações de uma memória externa auxiliar para a memória RAM.

LOOP — Laço. Parte de um programa que é executada várias vezes seguidas.

LSI — *Large Scale Integration.* Integração em larga escala. São circuitos integrados (*chip's*) equivalentes a milhares de componentes discretos.

LINGUAGEM DE MÁQUINA — É a linguagem que o microprocessador entende. Suas instruções são seqüências de pulsos elétricos e podem ser imaginadas como seqüências de 0's (ausência de sinal) e 1's (presença de sinal). A informação é codificada, portanto, no sistema binário.

MATRIZ — O significado mais usual de matriz é o matemático. Uma matriz é uma tabela em que cada elemento é distinguido dos demais por índices (ou coordenadas). O número de índices necessário para identificar um elemento de uma matriz é a sua dimensão. Por exemplo, a dimensão da matriz a seguir é dois, pois precisamos de 2 números para identificar cada um de seus elementos. O único número 8 dessa matriz está na posição (2,4), linha 2 e coluna 4. Veja também ARRAY.



MEMÓRIA — É qualquer elemento armazenador de informações.

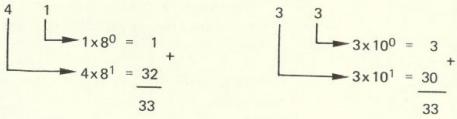
MENU — Apresentação de opções a serem escolhidas pelo usuário de um programa.

MICROPROCESSADOR — É a parte mais fundamental de um computador. É ele quem gerencia o fluxo de informação através da máquina. Veja CPU.

MNEMÔNICOS — São códigos da linguagem ASSEMBLY.

MODEM — *MODulator DEModulator.* Dispositivo que transforma dados em sinais que podem ser transmitidos ou recebidos por linha telefônica. Equipamento que interliga um microcomputador a um sistema telefônico.

OCTAL — Sistema de numeração baseado no número 8. O número decimal 33 equivale ao número octal 41.



OFF-SET — Deslocamento que deve ser acrescentado a um endereço base para obter outro endereço.

ON-LINE — Modo de operação de um sistema em que a comunicação da máquina com o usuário é direta, imediata e, geralmente através de vídeo.

OVERFLOW — Sobrecarga.

OUTPUT — É a saída de um sistema.

PADDLE — Controle manual geralmente empregado para jogos. Difere do *joystick* por ser analógico.

PERIFÉRICO — É qualquer elemento que pode ser acrescentado a um sistema, tornando-se parte dele.

PIXEL — Elemento fundamental de impressão no vídeo.

PLOTTER — Periférico destinado a confecção de gráficos e outros tipos de desenhos em papel.

POINTER — Indicador de posição. Ponteiro.

PRINTER — Impressora. Dispositivo que imprime dados enviados pelo computador em papel.

RAM — *Random Access Memory*. Memória de acesso aleatório. É a memória alterável onde são armazenados os programas e dados.

RANDÔMICO — O mesmo que aleatório. Ao acaso.

ROM — É a memória permanente (que não se perde quando o micro é desligado) onde está armazenada a linguagem BASIC e outras informações indispensáveis à operação.

REGISTRO — Elemento de memória interno de um microprocessador. Um registro só é acessado pelo próprio *chip* a que ele pertence.

RETURN — Retorno. É a tecla que envia as informações introduzidas através do teclado para a memória RAM do micro ou que ordena a execução de um comando. É também uma palavra do vocabulário BASIC usada no final de todas as subrotinas de um programa e que devolve o processamento ao programa principal.

RF — Rádio Freqüência. Sinal de freqüência modulada que chega à antena receptora de um aparelho de TV.

GLOSSÁRIO

RGB — Três sinais já decodificados a partir do sinal de vídeo-composto. Um é o sinal para cor vermelha (*Red*), outro para cor verde (*Green*) e outro para a cor azul (*Blue*).

ROTINA — É uma parte lógica de um programa que pode ser distinguida individualmente.

SCREEN — Tela.

SOFTWARE — Designação genérica de programa. Tudo que é relacionado com a programação faz parte do *software*. Um programa, é um *software*.

SPRITE — Máscara que pode ser sobreposta à tela. Figura definida através de *software* que pode ser movimentada e sobreposta a outras figuras na tela.

SUB-ROTINA — É uma parte de um programa que pode ser executada várias vezes em situações distintas.

SINTAXE — Forma correta de escrever.

SISTEMA OPERACIONAL — Sistema lógico de gerenciamento de uma máquina. É o responsável pela operação automática da máquina, independentemente do operador.

SLOT — É um encaixe para conexão de periféricos diversos, *drivers*, cartuchos, expansões, etc.

STRING — Nome que se dá a qualquer seqüência de caracteres introduzida no micro. Quando são usadas após comandos (como PRINT, LET, INPUT, etc...) devem estar entre aspas.

TERMINAL — Meio físico ou lógico através do qual saem ou entram informações num sistema.

TABELA VERDADE — Tabela usada na análise de operadores lógicos e que os determinam completamente. Por exemplo, a tabela do operador XOR é a seguinte:

E2 \ E1	0	1
0	0	1
1	1	0

TOUCH PAD — Painel sensível a toque. (veja PADDLE).

UTILITÁRIOS — São programas que agem sobre outros programas facilitando ao usuário a edição dos mesmos.

VARIÁVEL — É um elemento armazenador de dados.

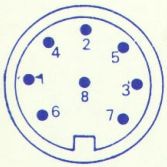
VÍDEO-COMPOSTO — Sinal da moduladora da RF. A partir deste sinal são obtidos os sinais RGB.

APÊNDICE XI

PINAGEM

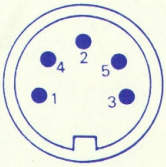
Pinagem RGB

1	Compose sync
2	terra
3	R
4	B
5	G
6	+Vcc
7	Y
8	áudio



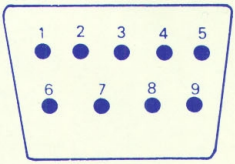
Pinagem do cassete

1	REM
2	Blindagem de dados
3	REM
4	Entrada de dados
5	Saída de dados



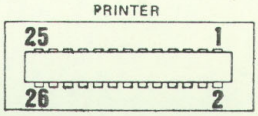
Pinagem Joystick

1	frente
2	trás
3	esquerda
4	direita
5	+5V
6	disparador 1
7	disparador 2
8	saída
9	(GND) terra



Pinagem da printer

2 a 24	Terra
19, 23, 25, 26	Sem conexão
1	STROBE
3	D0
5	D1
7	D2
9	D3
11	D4
13	D5
15	D6
17	D7
21	BUSY



Se você quiser obter mais informações sobre LITERATURA específica para o MSX escreva para a ALEPH Publicações e Assessoria Pedagógica Ltda. C.P.: 20.707 - São Paulo -SP

Editora Aleph

Gráfica Palas Athena
Associação "Palas Athena" do Brasil
Rua José Bento, 384
Fone: 279-6288 — CEP 01523
Cambuci — São Paulo

Editora Aleph

gradiente

CAIXA POSTAL 30.318 - SÃO PAULO - SP
PRODUZIDO NA ZONA FRANCA DE MANAUS
PRINTED IN BRAZIL
901-01940-000 06.2648.001

AS CORES DO EXPERT