Nélson Casari

MSX

PRÁTICA E DOMÍNIO

- Basic MSX
- Programas Comentados
- Arquivos em Fita
- Gráficos
- Desenhos
- Som

atlas

MSX

PRÁTICA E DOMÍNIO

EDITORA ATLAS S.A.

Rua Conselheiro Nébias, 1384 (Campos Elísios)
Caixa Postal 7186 — Tel.: (011) 221-9144 (PABX)
01203 São Paulo (SP)

NÉLSON CASARI

MSX

PRÁTICA E DOMÍNIO

- Basic MSX
- Programas Comentados
- Arquivos em Fita
- Gráficos
- Desenhos
- Som

3ª EDIÇÃO

SÃO PAULO
EDITORA ATLAS S.A. – 1988

Rua Conselheiro Nébias, 1384 (Campos Elísios)
Caixa Postal 7186 — Tel.: (011) 221-9144 (PABX)
01203 São Paulo (SP)

1. ed. — Agosto — 1986; 2. ed. 1986; 3. ed. 1988

ISBN 85-224-0180-2

Impresso no Brasil/**Printed in Brazil**

Depósito legal na Biblioteca Nacional, conforme Decreto nº 1.825, de 20 de dezembro de 1907.

Capa
Paulo Ferreira Leite

Dados de Catalogação na Publicação (CIP) Internacional
(Câmara Brasileira do Livro, SP, Brasil)

Casari, Nélson, 1931-
C33m MSX : prática e domínio / Nélson Casari. -- São Paulo : Atlas, 1988.

Bibliografia.
ISBN 85-224-0180-2

1. BASIC (Linguagem de programação para computadores) 2. MSX (Computadores) 3. MSX (Computadores) — Programação I. Título.

CDD-001.64
-001.642
-001.6424

86-1338

Índices para catálogo sistemático:

1. MSX : Computadores : Processamento de dados 001.64
2. MSX : Computadores : Programação : Processamento de dados 001.642
3. MSX BASIC : Linguagem de programação : Computadores : Processamento de dados 001.6424

Sumário

(*) = Programas comentados.

- Dedico este livro àqueles que o lerem para estudo, pois foi para e por eles que me enclausurei durante vários meses para produzi-lo.

- Agradeço ao Editor e seus Colaboradores, pois é por intermédio deles que meu modesto trabalho chega ao leitor interessado em aprender sempre algo mais.

- Agradeço a todos aqueles cujos esforços permitiram a realização deste livro e que doaram - muitas vezes anonimamente - seus conhecimentos e experiência, para benefício e proveito de muitos.

- Agradeço muito à minha família - em especial a meus filhos - em cujo seio encontrei o aconchego necessário para o trabalho cansativo.

- E por último - mas em primeira consideração - agradeço profundamente a Deus, pois sinto que é Ele quem me concede forças para tentar ser útil.

O autor

Introdução

Este livro parte da premissa de que, sendo proprietário de um computador da linha MSX, o leitor já está razoavelmente informado sobre o que seja essa pequena maravilha que a moderna tecnologia eletrônica coloca à nossa disposição para as mais variadas finalidades.

Assim, não vamos repetir os dados introdutórios constantes dos manuais de usuário e de BASIC, que são fornecidos juntamente com o aparelho e que, certamente, já terão sido, ou serão, examinados e estudados pelo leitor. Não vamos perder-nos também em considerações universais sobre computação e informática, sobejamente difundidas por todas as formas de comunicação.

Vamos, sim, entrar diretamente na questão que mais preocupa o novo proprietário ou usuário: lidar com a máquina o máximo possível, a fim de conseguir dominá-la no menor espaço de tempo e poder tirar o melhor proveito de suas enormes potencialidades.

A fim de poupar ao usuário o trabalho de ter de buscar em diversas fontes as informações de que necessita para uso de seu computador, procuramos reunir neste livro todos os dados considerados indispensáveis, tendo tido o cuidado, inclusive, de repassar todos os comandos, instruções e funções do MSX BASIC. Por essa razão, muitas informações aqui contidas são encontradas também nos manuais do usuário e de BASIC que acompanham o aparelho.

Tais manuais, entretanto, nem sempre podem ou conseguem tratar e transmitir com bastante clareza todos os dados que contêm, de maneira que esperamos estar oferecendo um complemento ou uma alternativa de esclarecimentos para as questões que eventualmente permaneçam em dúvida.

O livro está dividido em capítulos que abordam de maneira prática e direta os recursos básicos e avançados de programação. Todos os capítulos apresentam programas explicados ou comentados, visando propiciar exemplos efetivos de aplicação. O Capítulo 15 - PROCESSAMENTO DE ARQUIVOS - apresenta programas aplicativos que poderão encontrar utilidade prática imediata.

Ao longo dos anos, e cada dia mais, a experiência tem-nos convencido de que a melhor e mais eficiente forma de consolidar estudos e conhecimentos é através da prática. Assim, recomendamos ao leitor que digite e ponha em prática todos os programas aqui apresentados, analisando-os e experimentando alterar, tanto quanto possível, suas instruções mais importantes em cada estágio, a fim de melhor compreender sua estruturação.

O AUTOR

Familiarização com o teclado e o monitor de TV 1

1.1 TECLADO DATILOGRÁFICO

O teclado de um computador do padrão MSX é operado como o de uma máquina de escrever elétrica ou eletrônica. É um teclado similar ao tipo QWERTY, com algumas teclas e caracteres a mais.

Os MSX nacionais apresentam algumas pequenas diferenças nos teclados, particularmente com relação aos sinais de acentuação de palavras da língua portuguesa.

TECLA SHIFT

Para digitar letras maiúsculas ou os caracteres impressos na metade superior de algumas teclas, pressione SHIFT simultaneamente com a tecla correspondente.

TECLA CAPS

Se quiser escrever tudo em maiúsculas, no EXPERT, pressione uma vez a tecla CAPS. Para voltar ao normal, pressione-a novamente. A tecla CAPS funciona como a tecla fixadora de maiúsculas numa máquina de escrever, mas apenas para as letras do alfabeto. Os números e os caracteres impressos na parte superior das teclas de dupla função continuam sendo obtidos com o uso simultâneo da tecla SHIFT.

No HOTBIT o modo inicial de escrever é em maiúsculas. Logo após o computador ser ligado, a tecla CAPS é acionada automaticamente, como pode ser constatado pelo acendimento da pequena luz verde junto à mesma. Deve ser pressionada uma vez para escrita em letras minúsculas e outra vez para voltar ao modo normal.

TECLAS COM SETAS

Para retroceder o carro ou cilindro - representado pelo pequeno quadrado branco que aparece no vídeo - use a tecla ←.

Para avançar o carro ou cilindro, use a tecla →.

Para voltar a uma ou mais linhas acima: ↑.

Para pular ou descer uma ou mais linhas: ↓.

As teclas com setas são denominadas "teclas do cursor".

BARRA ESPAÇADORA

Para dar espaço(s) use a barra espaçadora, localizada como a de uma máquina de escrever.

TECLA ◄◄ ou BS

Para retroceder o carro ou cilindro, apagando letras ou caracteres digitados a mais ou por engano, use a tecla ◄◄ do HOTBIT ou BS do EXPERT.

CURSOR

O que chamamos de carro ou cilindro é, em verdade, o CURSOR, representado por um pequeno quadrado branco, como já foi dito, que indica o local de vídeo onde será escrito o próximo caractere digitado.

Escreva agora algo, uma carta, por exemplo, para se acostumar com o teclado. Faça-o como se estivesse treinando datilografia. Com auxílio das teclas ← → ↑ ↓ coloque o cursor no local em que dese-

ja iniciar o texto. O vídeo substitui o papel.

Se não tiver outra inspiração no momento, copie esta página.

1.2 TV COMO MONITOR DE VIDEO DO MSX

A tela de TV, ou um monitor de vídeo mesmo, é o meio de comunicação entre computador e usuário. Assim, todas as instruções ou comandos que fazemos ao computador através do teclado são monitorados no vídeo, de maneira que podemos acompanhá-los ou conferi-los. E também é através do vídeo que nos inteiramos de todas as ações do computador e resultados delas.

O vídeo é operado pelo MSX em dois modos:

MODO TEXTO e MODO GRÁFICO

Na primeira parte deste livro será utilizado apenas o MODO TEXTO, que se subdivide em:

MODO TEXTO 1, operando a tela de vídeo em 24 linhas de até 40 colunas cada uma, e

MODO TEXTO 2, operando a tela de vídeo em 24 linhas de até 32 colunas cada uma.

Ao ser ligado, o computador opera automaticamente no modo TEXTO 1, em 39 colunas de impressão. Para operar em 40 colunas ou outra largura, deverá ser usada a instrução WIDTH. Consulte-a através do índice remissivo.

INSTRUÇÃO SCREEN

Define o modo de utilização de tela.
SCREEN 0 define o modo TEXTO 1.
SCREEN 1 define o modo TEXTO 2.
A instrução SCREEN define também:
- dimensão de "sprite" (0 a 3. Ver Capítulo 18 a respeito)
- estalido de uso do teclado (0: sem. 1: com estalido)

- velocidade de gravação de dados em fita cassete (1: 1200 bauds. 2: 2400 bauds)
- padrão de impressora usada (0: padrão MSX. 1: padrão ABICOMP)

Tais parâmetros são definidos na ordem exposta e separados por vírgulas. Exemplo:

SCREEN 1,2,0,2,0

define modo de texto 1, tamanho de "sprites" 2, uso do teclado sem estalido, velocidade de gravação de dados em fita cassete de 2400 bauds, impressão por impressora padrão MSX.

Quando o computador é ligado, são adotados automaticamente os seguintes parâmetros:

SCREEN 0,0,1,1,1

Quando definido ou alterado por uma instrução, um parâmetro é mantido até posterior alteração ou até que o computador seja desligado ou "resetado", isto é, rearmado.

Para definir ou alterar um parâmetro, não é preciso citar os demais, bastando colocar a vírgula que os sucede. Exemplo:

SCREEN,,0

para eliminar o estalido de uso do teclado,

SCREEN,,,2

para determinar velocidade de gravação em fita cassete de 2400 bauds.

LIMPANDO O VÍDEO

Se quiser eliminar os escritos que aparecem na parte superior do vídeo logo após o computador ser ligado, empregue um dos seguintes recursos:

- pressione simultaneamente as teclas SHIFT e CLS/HOME; ou
- pressione simultaneamente as teclas CTRL e L; ou use a instru

ção comentada a seguir.

INSTRUÇÃO CLS

A instrução CLS limpa o vídeo, deixando apenas as palavras correspondentes às teclas de função.

Com qualquer dos recursos citados o cursor é deslocado para o canto superior esquerdo do vídeo, automaticamerte, após estar limpo, com exceção da vigésima quarta linha, onde permanecem as seguintes palavras:

COLOR, AUTO, GOTO, LIST e RUN.

São as palavras correspondentes às teclas de função. Falaremos sobre elas mais adiante. No momento desejamos apagá-las, para deixar o vídeo completamente limpo. Para tanto, deve ser usada a

INSTRUÇÃO KEY OFF

A instrução KEY OFF elimina do vídeo as aludidas palavras correpondentes às teclas de função programável.

Digite KEY OFF ou key off e pressione a tecla RETURN.

Apague em seguida as palavras KEY OFF através de qualquer dos recursos citados.

Antes de prosseguir, pratique um pouco mais com o teclado, digitando o que lhe ocorrer na mente. Quando terminar, releia o que escreveu e verifique se está tudo certo.

1.3 EDIÇÃO E CORREÇÕES

Se tiver que corrigir algum caractere digitado errado, posicione o cursor sobre ele, através das teclas do cursor, e digite o caractere certo. A substituição processar-se-á automaticamente.

Esqueceu de digitar alguma letra ou palavra? Leve o cursor até

onde deverá ser feita a inserção e pressione a

TECLA INS

A tecla INS permite fazer inserções de espaços, letras, palavras e até frases em textos ou linhas de programas.

Note que, na modalidade de inserção, após a tecla INS ter sido pressionada, o cursor fica reduzido à metade vertical de seu tamanho.

Depois de feita a inserção, o cursor pode ser removido do local e movido normalmente através das teclas próprias. Qualquer delas reverte o cursor à sua modalidade normal.

Se for feita uma inserção em linha de instrução, deverá ser usada em seguida a tecla RETURN, para validade da mesma.

Se, em vez de letra ou caractere, foi esquecido um espaço em branco ou mais de um, repita o procedimento de localização, use a tecla INS e a barra espaçadora para criar o espaço necessário. Remova o cursor do local em seguida, usando as teclas próprias ou RETURN.

TECLA DEL

Elimina letras, caracteres ou espaços do local em que estiver posicionado o cursor. Basta pressioná-la.

1.4 OUTROS CARACTERES E SÍMBOLOS

Observe no vídeo os caracteres e símbolos que não estão impressos nas teclas:

TECLA GRAPH

Pressione a tecla GRAPH simultaneamente com cada uma das teclas do teclado.

TECLA CODE

Pressione simultaneamente cada uma das teclas do teclado com a tecla CODE. Não estranhe se algumas teclas, ao serem pressionadas, não apresentarem nada no vídeo, nesta modalidade.

TECLAS GRAPH + SHIFT

Pressione simultaneamente ambas com cada uma das teclas do teclado. Nesta modalidade, também, algumas teclas não têm função. Não estranhe, portanto, se nem todas apresentarem algo no vídeo quando forem pressionadas.

Antes de prosseguir, limpe o vídeo para poder perceber a função da tecla que será introduzida a seguir. Pressione SHIFT e CLS/HOME simultaneamente.

TECLA TAB

Desloca o cursor com intervalo de oito espaços, a partir da coluna 0, a primeira à esquerda do vídeo. Pressione-a algumas vezes para observar bem sua atuação. A tecla TAB pode ser muito útil na digitação de tabelas, por exemplo.

1.5 TECLAS DE FUNÇÃO PROGRAMÁVEL

Quando começamos a praticar com o teclado, fizemos o comando direto KEY OFF, que eliminou da última linha do vídeo as palavras:

COLOR AUTO GOTO LIST RUN

Essas palavras são impressas automaticamente na posição de vídeo referida, assim que o computador é ligado. Caso o computador tenha permanecido ligado desde que foi feito referido comando, elas ainda estarão ausentes do vídeo. Façamo-las reaparecer.

Para tanto, servir-nos-emos da instrução apresentada a seguir:

INSTRUÇÃO KEY ON

Produz na vigésima quarta linha de vídeo, nos modos TEXTO 1 e 2, as palavras correspondentes às teclas de função programável.

Digite KEY ON ou key on e pressione RETURN. Em seguida, limpe o vídeo com CLS e RETURN.

Pressione agora, uma de cada vez, as teclas grandes F1, F2, F3, F4, F5, situadas na parte superior do teclado.

Note que as palavras aparecerão na parte superior do vídeo e na mesma ordem em que aparecem na parte inferior e, também, na mesma ordem das teclas, pois a cada uma corresponde o comando representado pela palavra respectiva.

(A notação ERRO DE SINTAXE (no HOTBIT) ou SYNTAX ERROR (no EXPERT), surgida no vídeo quando foi pressionada a tecla F5 (=RUN), deve-se ao fato de que, ao pressionar as precedentes consecutivamente, introduzimos os respectivos comandos um após outro, evidentemente com erro de sintaxe, fato que originou a emissão da citada notação.)

Mantenha agora apenas a tecla SHIFT pressionada durante alguns instantes e observe as palavras que aparecem na última linha do vídeo:

COLOR 1 CLOAD" CONT LIST. RUN

Tais palavras, que se referem também às teclas F quando pressionadas simultaneamente com a tecla SHIFT, correspondem às posições F6, F7, F8, F9, F10, respectivamente.

Vejamos como funcionam os comandos das teclas de função programável. Antes, porém, limpe o vídeo pressionando CTRL + L.

COLOR

Define as cores a serem utilizadas no vídeo.

Pressione a tecla F1. Digite em seguida 1,15. Pressione RETURN.

Faça outras experiências com combinações de cores diferentes, lembrando que o primeiro parâmetro se refere à cor do primeiro plano e o segundo à de fundo. Pode ser usado ainda um terceiro parâmetro, mas isto será visto mais adiante.

AUTO

Gera automaticamente números de linhas de programas.

Pressione a tecla F2 e em seguida RETURN. Surgirá no vídeo a palavra AUTO e, na linha seguinte à da mesma, o número 10, automaticamente, como numeração inicial de linha de programa.

Digite logo após o número 10:

PRINT "ESTA E A LINHA NÚMERO 10"

Pressione a tecla RETURN, após o que surgirá automaticamente no vídeo o número 20. Digite após o mesmo:

PRINT "ESTA E A LINHA 20"

Pressione em seguida a tecla RETURN e repita o procedimento até o número 50.

Quando aparecer o número 60, depois de pressionada a tecla RETURN, interrompa o processo de numeração automática de linhas de programa, pressionando CTRL + STOP. Deixe as linhas digitadas no vídeo.

GOTO

Dirige a execução de um programa para a linha de instrução cujo número especifica.

Pressione a tecla F3. Surgirá no vídeo a palavra GOTO. Digite logo após o número 10 e pressione a tecla RETURN. Repare que as cinco linhas digitadas pouco antes aparecem agora no vídeo com outro aspecto, isto é, sem números, sem a palavra PRINT e sem aspas, porque, na forma em que foram escritas, constituem linhas de instruções de programa e foram executadas quando, através de GOTO, a ação do computador se dirigiu para elas.

Limpe o vídeo (CTRL + L), pressione novamente a tecla F3, digite o número 20 após a palavra GOTO que surgirá no vídeo e pressione a tecla RETURN.

Repita o procedimento até a linha 50, observando sempre o efei-

to da instrução GOTO usada em comando direto.

Limpe o vídeo, digitando agora CLS e pressionando em seguida a tecla RETURN.

LIST

Lista no vídeo um programa, parcial ou total.

Pressione a tecla F4 e em seguida a tecla RETURN. Limpe de novo o vídeo.

Repita o procedimento com F4 e RETURN.

As linhas que aparecem "listadas" no vídeo, que são as digitadas para experiência do comando AUTO (tecla F2), permanecem na memória do computador porque constituem um programa, embora pequeno. Por essa razão, ao ser feito o comando LIST, o computador apresenta no vídeo a listagem do programa que se encontra armazenado em sua memória.

RUN

Roda um programa, o que significa que executa as instruções contidas nele.

Pressione a tecla F5. Não é necessário pressionar RETURN para execução de sua função.

O pequeno programa existente na memória do computador será executado novamente. Repita o procedimento e note que o programa é executado a partir do primeiro espaço livre no vídeo.

Limpe o vídeo mais uma vez.

COLOR 15,4,4 ou COLOR 15,1,1

Define as cores a serem utilizadas na tela, em conformidade com os parâmetros de seu conteúdo.

Pressione simultaneamente as teclas SHIFT e F6. Não é necessá-

rio pressionar em seguida a tecla RETURN. As cores definidas no comando serão as cores da tela de vídeo, ato contínuo.

CLOAD"

Permite carregar ou passar para a memória do computador um programa em BASIC previamente gravado em fita cassete.

Depois de pressionada a tecla F7, o nome do programa deverá ser digitado, se o mesmo tiver de ser "procurado" na fita pelo computador. Se o programa a ser carregado for o primeiro a ser reproduzido da fita pelo gravador, não há necessidade de digitar o nome do mesmo. Bastará deletar o sinal de aspas seguinte a CLOAD. Para o processo de carregamento ser iniciado deverá ser pressionada a tecla RETURN em seguida.

Naturalmente, o gravador deverá estar conectado ao computador adequadamente. O Capítulo 14 trata desse assunto.

CONT

Permite a continuidade de execução de um programa interrompido pelo comando conjunto CTRL + STOP, ou por uma das instruções STOP ou END de linhas de instruções de um programa. Não necessita da complementação de RETURN para sua função ser desempenhada. Está programado na tecla F8.

Limpe novamente o vídeo.

Pressione F4 para verificar se o computador ainda retém na memória "aquele mesmo" pequeno programa. Em caso positivo, vamos ter de bani-lo da memória do computador, para continuar nossa experiência com as teclas de função programável. Para tanto, faça o seguinte:

Digite NEW e pressione em seguida RETURN.

Constate a seguir, pressionando F4 e RETURN, que o pequeno programa sumiu da memória do computador.

Vamos, pois, continuar com a experiência da tecla F8, digitando o "programinha" que segue:

```
10 PRINT "PRIMEIRA ETAPA"
20 STOP
30 PRINT "SEGUNDA ETAPA"
40 END
50 PRINT "TERCEIRA ETAPA"
60 PRINT "FIM"
```

Pressione agora a tecla F5. O programa será executado até a linha 20 e aparecerá no vídeo a notação PAREI EM 20.

Pressione a tecla F8 (simultaneamente com a tecla SHIFT, não se esqueça) e o programa será executado a partir do ponto em que parou, linha 20, até a linha 40, onde parará sem apresentar nenhuma notação porque encontra uma instrução END (= fim).

Pressione novamente F8 + SHIFT e note que as linhas 50 e 60 serão executadas, não obstante se encontrarem após a instrução END, porque o comando CONT dará continuidade ao programa.

Se for feito um comando CONT que não possa ser executado, o computador escreverá no vídeo NÃO CONTINUO! (HOTBIT) ou CAN'T CONTINUE (EXPERT).

LIST.

Exibe no vídeo a última linha digitada para um programa.

Note que não se trata de LIST apenas, mas de LIST. (com ponto após a palavra), no HOTBIT.

Pressione F9 + SHIFT e constate que aparecerá no vídeo a última linha digitada para o programa feito para exercitar o comando CONT.

Repare que não falamos <u>última linha do programa</u>, mas <u>última linha digitada</u>.

Se o programa tiver centenas de linhas de instruções, e a última digitada foi a de número 20, por exemplo, esta é que será apresentada no vídeo pelo comando da tecla F9.

No EXPERT, o comando LIST da tecla F9 é o mesmo da tecla F4.

RUN

Trata-se de RUN da tecla F10 e não da tecla F5.

Pressione F10 + SHIFT e note que a diferença do comando desta tecla em relação ao da tecla F5, no HOTBIT, é que, antes de começar a execução do programa, o computador executa automaticamente no vídeo um CLS, passando o programa a ser executado a partir da primeira linha da tela, enquanto o comando RUN da tecla F5 executa o programa a partir do primeiro espaço livre.

No EXPERT o comando da tecla F10 é o mesmo da tecla F5.

PROGRAMAÇÃO DAS TECLAS DE FUNÇÃO PROGRAMAVEL

Em conformidade com sua denominação, as teclas F1 a F10 podem ter suas funções programadas pelo usuário.

Assim, podemos atribuir a qualquer delas comandos, instruções ou funções que nos sejam mais convenientes, para melhor utilização do computador.

COMANDO KEY

Define comandos, instruções ou funções para as teclas de função programável.

Vamos atribuir à tecla F2, a título de exemplo, uma instrução como PRINT.

Digite: KEY 2, "PRINT"

Pressione RETURN em seguida. Pronto! Se não quiser mais digitar letra por letra a instrução PRINT toda vez que tiver de usá-la, bastará pressionar a tecla F2.

Ainda a título de experiência, vamos atribuir à tecla F8 a instrução CLEAR:

Digite: KEY 8, "CLEAR"

Pressione RETURN em seguida. Agora a tecla F8 está programada para introduzir a instrução CLEAR toda vez que for pressionada.

Como pudemos ver, uma das grandes conveniências destas teclas é a de não precisarmos digitar letra por letra as palavras, funções, instruções ou comandos programados para as mesmas.

As teclas de função programável podem receber até 15 caracteres de programa, o que é bastante cômodo quando temos de usar com acentuada frequência palavras ou instruções razoavelmente longas durante a digitação de programas, como, por exemplo:

```
PRINT USING"##"
LOCATEA,3:PRINT
PRINT STRING$(
```

COMANDO KEY LIST

Exibe no vídeo a listagem dos conteúdos das dez teclas de função programável. Verifique com o comando direto:

```
KEY LIST  /  RETURN
```

Modalidades de uso 2

2.1 MODALIDADE DIRETA SEM COMANDOS

Com exceção dos dois programas pequenos introduzidos no computador para experiências com as teclas de função programável, até agora o computador funcionou praticamente como uma máquina de escrever, tendo sido usada a tela de TV em vez do papel para escrita, mas não memorizou o que foi escrito na modalidade em que o usamos até este ponto, isto é, na modalidade direta sem comandos.

Apagar o que estiver no vídeo nessa modalidade equivale a destruir ou jogar fora o papel em que foi datilografado um texto numa máquina de escrever.

2.2 MODALIDADE DIRETA COM COMANDOS

Ainda com a finalidade de adquirir prática no manejo do teclado, vamos agora usar o computador na modalidade direta com comandos.

Nessa modalidade, o computador executa prontamente uma "ordem" recebida, mas não a "guarda" na memória para outras execuções. Ele não memoriza a ordem recebida.

A ordem pode ser um comando, uma instrução ou uma função. Dada uma instrução como a que segue, por exemplo:

```
LET A = 1986
```

e pressionada a tecla RETURN em seguida, o computador registrará na variável numérica A o número 1986 e manterá esse registro enquanto não for alterado ou não for desligado o computador, mas não memorizará a instrução feita no modo direto.

Certifique-se disso com a seguinte experiência:

Digite: PRINT A

Pressione RETURN. Notará que será exibido no vídeo o número registrado na variável A.

Apague agora esse número da memória do computador, utilizando a instrução CLEAR, que limpa apenas a área das variáveis sem destruir instruções ou programa porventura existentes na memória:

Digite: CLEAR

Pressione RETURN.

Digite: PRINT A

Pressione RETURN e note que A agora é igual a 0.

Experimente também fazer os comandos diretos LIST ou RUN a fim de verificar se a instrução LET A = 1986 ficou memorizada.

Constatará que não, pois instruções ou comandos diretos são executados pelo computador e "descartados" em seguida. Mais adiante veremos que, para que tal não aconteça, é necessário caracterizar uma instrução como linha de programa.

NOTAÇÃO OK

Pudemos verificar que, após ter executado um comando, o computador sempre apresenta no vídeo a notação OK.

Essa é a maneira de o computador informar que está pronto para receber outro comando.

TECLA RETURN

Pudemos verificar também e percebemos que um comando direto só é executado após ser pressionada a tecla RETURN.

Faça algumas experiência mais:

Digite: PRINT "Treinando no teclado."

Não pressione RETURN.

Digite: CLS

Pressione RETURN e note que a instrução PRINT não foi executada, enquanto a instrução CLS foi prontamente executada.

Digite: PRINT "Imprimindo ou escrevendo no vídeo."

Pressione RETURN e note que a instrução PRINT é prontamente executada.

Pressione novamente RETURN e note que nada é executado agora.

Faça ainda a seguinte experiência:

Com auxílio das teclas do cursor, leve este até a letra final da última frase impressa no vídeo - não a que está entre aspas após a instrução PRINT, mas a que foi escrita no vídeo pela execução da instrução - e apague-a totalmente, usando a tecla com duas setinhas ou BS (= "back space").

Em seguida, coloque o cursor em qualquer parte da linha de instrução - PRINT "Imprimindo ou escrevendo no vídeo." - e pressione a tecla RETURN.

Note que, nessa condição, a instrução também é prontamente executada, pois o computador assume a linha como um novo comando ou uma nova instrução a ser executada. Essa versatilidade é cômoda quando desejamos repetir comandos com instruções longas: levamos o cursor até alguma parte delas e pressionamos simplesmente a tecla RETURN.

Mas, ainda assim, comandos ou instruções nessas condições, isto é, sem configuração de programa, não são memorizados pelo computador para repetições eventuais.

2.3 MODALIDADE INDIRETA COM COMANDOS E INSTRUÇÕES

Se quisermos escrever ou produzir algo para ser repetido diversas vezes ou freqüentemente, será necessário "manter" a instrução correspondente na memória do computador. E isto só pode ser feito através de um programa, sendo necessário escrevê-lo e inseri-lo na memória do computador, o que significa programar o computador para finalidades determinadas, como veremos no capítulo seguinte.

Programação do computador 3

Programar o computador significa introduzir em sua memória uma série de instruções e comandos para que ele os execute seqüencialmente, cumprindo alguma tarefa específica. Por exemplo: realizar determinados cálculos, imprimir um texto, elaborar um gráfico ou um desenho, tocar uma música etc.

Para tanto, é necessário que as instruções e os comandos sejam digitados no padrão sintático da linguagem do microcomputador e dispostos em linhas numeradas na ordem em que devem ser executadas.

A fim de que a execução seja perfeita e apresente os resultados objetivados com precisão, não basta atender apenas a esses requesitos. É preciso que o programa não contenha erros de lógica, cuja detecção não é feita pelo computador, que está preparado somente para apontar erros de sintaxe.

Os recursos de programação postos à nossa disposição pelo BASIC do MSX são excepcionalmente valiosos e variados e a melhor forma de conhecê-los, entender sua finalidade e alcance e fixá-los em nossa memória é através da prática pelo uso.

Vamos, pois, começar a programar, com o objetivo de fazer o uso maior possível de todos os comandos, instruções e funções disponíveis no MSX. Antes, porém, visando estabelecer um quadro básico completo sobre o assunto, examinemos rapidamente algumas técnicas e recursos adotados em programação:

3.1 PROGRAMAÇÃO ESTRUTURADA

Um programa deve ser elaborado de maneira que funcione com eficiência, ser de entendimento fácil quanto possível a qualquer usuário e oferecer boas possibilidades de alterações ou adaptações.

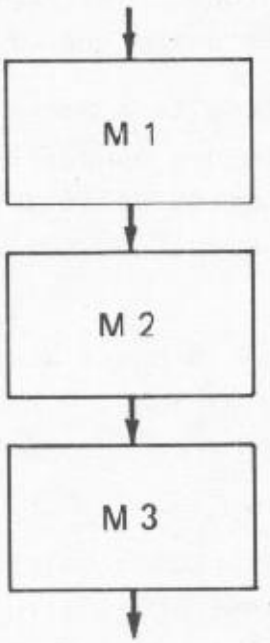
Essa é a filosofia básica da chamada PROGRAMAÇÃO ESTRUTURADA, uma das técnicas de elaboração de programas mais racionais e utilizadas na atualidade.

Resumidamente, os princípios dessa técnica de programação preconizam ser possível elaborar qualquer programa utilizando apenas combinações de três tipos de estruturas básicas, sem perder de vista concisão e eficiência de execução.

As chamadas estruturas básicas são as seguintes:

ESTRUTURA SEQUENCIAL ou DE SEQUENCIA SIMPLES:

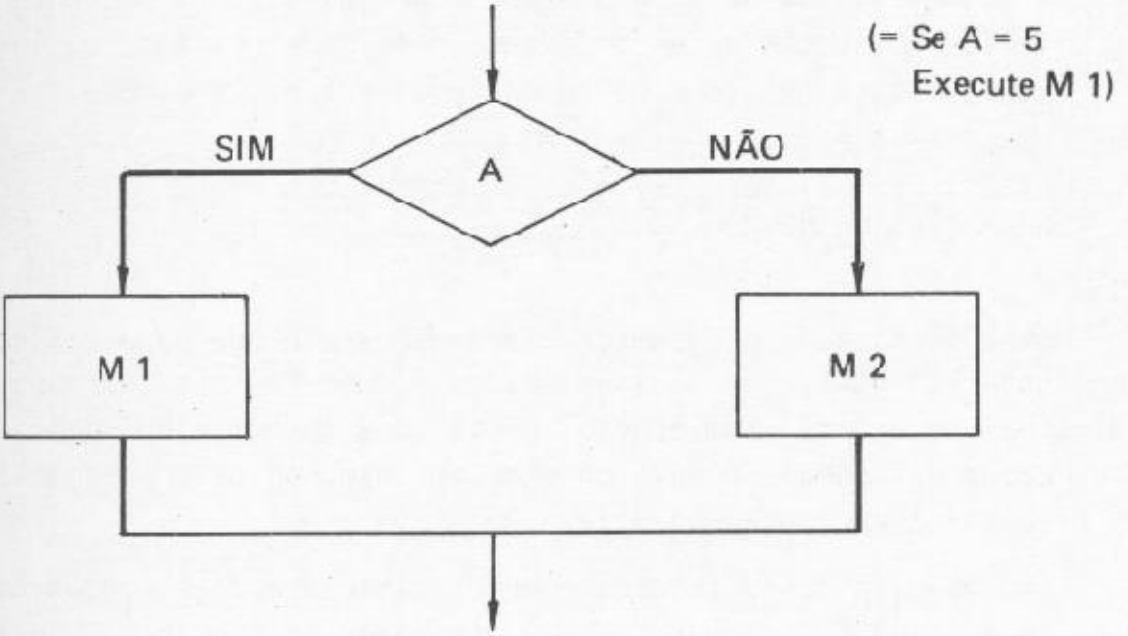
O programa é elaborado em módulos, que são executados consecutivamente, sem desvios. O fluxograma de um programa assim será:



(= Execute M 1
Execute M 2
Execute M 3)

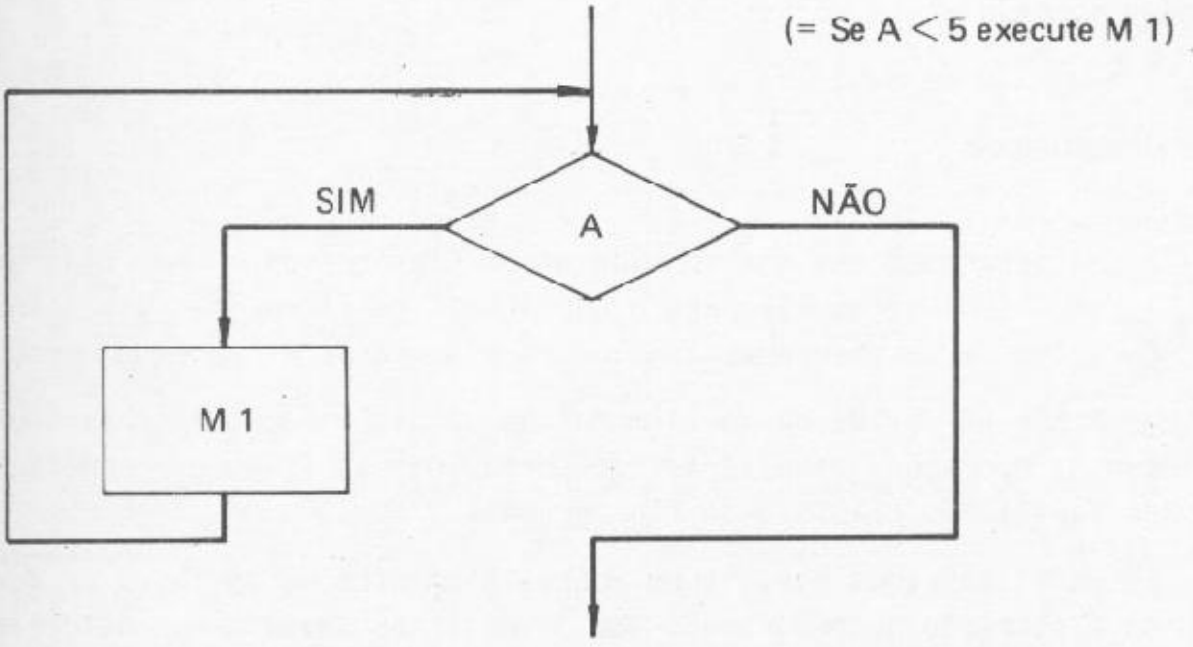
ESTRUTURA CONDICIONADA ou DE COMANDOS SELECIONADOS POR IF/THEN/ELSE:

O programa é elaborado em módulos cuja execução depende de condições existentes como falsas ou verdadeiras após a instrução IF. Se fluxograma típico:



ESTRUTURA DE "LAÇO" (LOOP) ou DE CICLO REPETITIVO:

Qualquer módulo do programa poderá ser repetido n número de vezes até que determinada condição seja satisfeita. Seu fluxograma:



Observação: Tendo em vista os princípios preconizados por essa técnica, a instrução GOTO deve ter seu uso minimizado em PROGRAMAÇÃO ESTRUTURADA. Essa restrição é devida à versatilidade da instrução GOTO que, segundo certas correntes defensoras dessa modalidade de programação, é prejudicial à estruturação límpida e fluente de um programa, pois pode desviar o fluxo de execução para qualquer outra parte do próprio programa.

3.2 PROGRAMAÇÃO TOP DOWN

Outra técnica de programação largamente adotada na atualidade é a denominada TOP DOWN, que podemos traduzir como "de cima para baixo", a qual consiste em criar um módulo inicial que gerencie ou supervisione a execução dos demais módulos que devem integrar o programa, sendo estes elaborados em função daquele.

Como exemplo dessa técnica podemos citar a criação de um "menu" como parte inicial ou "roteiro" de um programa, em função do qual são elaborados os demais módulos. Quando da execução do programa, o "menu" não só controla os diversos módulos, mas também funciona como agente de integração dos mesmos.

A nosso ver, a utilização dessas duas técnicas, PROGRAMAÇÃO ESTRUTURADA e TOP DOWN, bem como sua conjugação quando possível, constituem a base da boa programação de microcomputadores, devendo ser estudadas com seriedade, através de obras específicas, por aqueles que pretendem aprofundar-se no assunto.

3.3 FLUXOGRAMAS

Há programadores que consideram os fluxogramas - ou diagramas de blocos - indispensáveis para esquematizar graficamente a sequência de operações de um programa, de maneira a se tornar a base do mesmo.

E não há dúvida de que, em muitos casos, o raciocínio para estabelecer as operações necessárias de forma lógica, fluente e concisa pode ser facilitado pelo uso de fluxogramas.

Os fluxogramas constituem apenas elementos de apoio ou de auxílio na elaboração de programas. Não podem e não devem ser considerados

como elementos básicos de programas nem como guias absolutos ou indispensáveis em sua elaboração.

A adoção de seu uso deve depender do grau de auxílio que podem prestar ao programador em face da complexidade dos programas que deve elaborar.

Depois do advento da linguagem BASIC, de extrema inteligibilidade e muita semelhança com o linguajar humano, simplificando bastante a estruturação de comandos e instruções, a utilização de microcomputadores tornou-se muito mais fácil e simples, permitindo que problemas bastante complexos sejam programados sem grandes dificuldades e, na maioria dos casos, apenas com auxílio de um roteiro preestabelecido.

Finalmente, deve ser dito que, sejam quais forem as técnicas e os métodos adotados e as magnitudes de certos recursos, pouco valerão se não forem utilizados com habilidade, bom senso e, especialmente, razoável criatividade.

3.4 COMEÇANDO A PROGRAMAR

Como vimos, um programa constitui um conjunto de linhas de instruções numeradas, que devem ser introduzidas na memória do computador. A inserção é feita pressionando-se a tecla RETURN (comando RETORNO DE LINHA) após cada linha ter sido digitada. Se formos repetir essa lembrança a cada linha digitada, vamos tornar enfadonha a tarefa. Assim, somente quando se tornar realmente necessária a lembrança, como nos casos de comandos diretos, por exemplo, usaremos / RETURN (uma barra e a palavra).

Devemos lembrar, por fim, que os conceitos de LINHA DE INSTRUÇÃO ou LINHA DE PROGRAMA e de LINHA DE VÍDEO ou LINHA DE IMPRESSÃO são diferentes.

Uma LINHA DE INSTRUÇÃO ou LINHA DE PROGRAMA no MSX pode conter até 254 espaços, enquanto uma LINHA DE VIDEO ou LINHA DE IMPRESSÃO pode conter até 40 espaços no modo TEXTO 1 e até 32 espaços no modo TEXTO 2. LINHA DE IMPRESSÃO EM PAPEL depende das características da impressora.

Uma linha de programa pode conter diversas instruções diferentes, desde que separadas por : (dois-pontos). Os 254 espaços menciona-

dos incluem os ocupados pelo número da linha, espaços "em branco" e as próprias instruções.

3.5 PROGRAMA INICIAL

```
10 CLS
20 PRINT TAB(4)"Este programa é apenas
de texto."
30 PRINT
40 PRINT TAB(4)"Um programa se caracter
iza por um"
50 PRINT"conjunto de instruções na form
a deste."
60 PRINT
70 PRINT TAB(4)"Se não contiver erros o
 programa é"
80 PRINT"executado até o fim."
90 PRINT,,TAB(4)"Se contiver erro de si
ntaxe em al-"
100 PRINT"guma linha, a execução é inte
rrompida"
110 PRINT"e o computador informa onde s
e locali-"
120 PRINT"za o mesmo, para correção."
130 END
```

"Rode" o programa, isto é, faça o computador executá-lo.

Digite RUN ou run / RETURN ou simplesmente pressione a tecla F5.

A execução do programa reproduzirá no vídeo o texto contido em suas linhas de instruções, de acordo com a estética determinada pelas mesmas. Note as separações silábicas e de palavras de uma mesma frase, feitas a partir das próprias instruções a fim de que não fiquem truncadas após a execução. Se deixarmos a cargo do computador essa tarefa, a separação será feita sempre quando a palavra ou frase atingir o fim da linha em que estiver sendo escrita no vídeo.

Deve ser lembrado que, ao ser ligado, o computador opera automaticamente no modo TEXTO 1 (SCREEN 0), em 39 colunas de impressão. Para operar em 40 colunas, que é a largura máxima desse modo, deve ser fei-

to o comando direto SCREEN 0: WIDTH 40 / RETURN.

INSTRUÇÃO PRINT

Faz aparecer ou "imprime" no vídeo o que estiver definido por ela entre aspas. Se a definição for uma operação aritmética a ser efetuada, as aspas devem ser eliminadas. Exemplos:

```
10 PRINT "OPERAÇÃO ARITMÉTICA 5 x 4 : 6"
20 PRINT 5*4/6
```

PRINT (com vírgula após)

Faz com que o início de impressão seja na 14ª coluna da tela de vídeo, a contar da esquerda para a direita. Exemplo:

```
10 PRINT, "ENTENDIDO"
```

PRINT,, (com duas vírgulas após)

Desloca o início de impressão para uma segunda linha, deixando a primeira em branco. A linha 90 do programa retroapresentado aplica o recurso. Se não tiver uma instrução complementar (como TAB(4) no mesmo programa), a impressão será iniciada na coluna 0.

PRINT,,, (com três vírgulas após)

Desloca o início de impressão para a coluna 14 da segunda linha.

PRINT,,,, (com quatro vírgulas após)

Desloca o início de impressão para uma terceira linha, coluna 0, saltando duas linhas, que ficam "em branco".

PRINT (sem complemento)

"Imprime" no vídeo uma linha "em branco". As linhas 30 e 60 do programa citado são exemplos.

? (sinal de interrogação) poderá substituir a palavra da instru-

ção PRINT em comandos diretos e em linhas de instruções. Exemplos:

```
? "OPERAÇÃO ARITMÉTICA 5 x 125 = "; 5*125
10 ? "BRASIL, PATRIA GENEROSA"
```

No caso de linha de instruções, o computador substitui automaticamente o sinal de interrogação pela palavra PRINT, após receber o comando LIST.

INSTRUÇÃO LPRINT

Funciona como PRINT, mas apenas para a impressora.

INSTRUÇÃO LOCATE

Determina a posição do cursor na tela de vídeo para execução de instruções como PRINT, LIST e INPUT. Só é válida nos modos TEXTO 1 e 2. Sua sintaxe é:

```
LOCATE x,y
```

sendo x a coordenada referente a coluna e y a coordenada que se refere a linha.

FUNÇÃO TAB

É complementar de PRINT e determina a coluna para início de impressão no vídeo. Usada com LPRINT, determina a coluna para início de impressão por impressora. Sua sintaxe é:

```
PRINT TAB(n)
```

sendo n o número da coluna.

No programa inicial apresentado, a função TAB é responsável pelos espaços em branco no início de cada parágrafo.

FUNÇÃO SPC

Imprime espaços na tela de vídeo ou no papel, neste caso através de uma impressora. Funciona apenas com PRINT e LPRINT.

Enquanto TAB "pula" espaços em branco a partir da coluna 0, SPC "imprime" espaços em branco a partir do último caractere impresso no vídeo ou através de impressora. Sua sintaxe é:

PRINT SPC(n)

sendo n o número de espaços.

Continuando o exercício de prática, limpe agora o vídeo e pressione em seguida a tecla F5.

O texto que aparece novamente no vídeo é produto da execução do programa que está na memória do computador. Se este não for desligado, poderá ser executado indefinidamente.

Quando o computador é desligado, tudo que estiver armazenado na memória RAM é destruído ou apagado.

Se se deseja preservar ou guardar um programa, será necessário guardá-lo numa memória auxiliar ou secundária. Fitas cassete e discos magnéticos são memórias auxiliares ou secundárias em que podem ser armazenados ou guardados programas. Leia o Capítulo 14 a esse respeito.

Para apagar um programa da memória do computador, não é preciso desligá-lo. Isto pode ser feito pelo comando introduzido em seguida:

COMANDO NEW

Limpa a memória do computador, zerando inclusive as variáveis. Deve ser usado sempre antes da digitação de um novo programa.

Digite: NEW

Pressione a tecla RETURN.

Constate que o comando NEW foi executado:

Pressione a tecla F5.

Digite: LIST

Pressione a tecla RETURN.

Pudemos verificar que a memória do computador está "limpa".

Limpe agora o vídeo. Vamos digitar um programa de linha única.

3.6 PROGRAMA DE LINHA ÚNICA

Digite:
```
10 CLS: PRINT "QUE PROGRAMA!": FOR I=1 TO
   200: NEXT I: GOTO 10
```

Pressione a tecla F5 e observe no vídeo o pequeno programa em plena execução! Não obstante o mesmo conter uma única linha, há nela um conjunto de instruções que fazem o computador executá-lo em conformidade com o pretendido em sua elaboração.

TECLA STOP

Pressionada simultaneamente com a tecla CTRL, susta a execução de um programa em andamento, dando um "stop" ou "break" no mesmo.

Dê um "stop" na execução do programinha de linha única, pois, se não o fizer, ele só parará quando o computador for desligado.

A tecla STOP não deve ser confundida com a instrução STOP, que será introduzida logo mais.

3.7 SEGUNDO PROGRAMA

Suponha que temos de determinar a área de um quadrado de lado = 5. Aplicamos a fórmula adequada (a=lxl) e fazemos um comando direto no computador:

```
PRINT 5*5 / RETURN
```

resultando 25. Não foi preciso fazer um programa para obter esse resultado. O computador desempenhou a função de calculador, isto é, funcionou como uma simples máquina calculadora.

Todavia, se tivermos de calcular as áreas de vários terrenos - digamos 10 - e tê-las todas juntas para comparação, por exemplo, a solução racional que se impõe é programar o computador para essa finali-

dade. Vamos fazê-lo:

Identificaremos os terrenos por números e adotaremos três variáveis para representar suas medidas: A para área, F para frente e L para lado.

O programa poderá ser escrito assim:

3.8 PROGRAMA "AREA DE TERRENOS"

```
10 CLS:T=1:PRINT "QUANTOS TERRENOS?"
20 INPUT Q:DIM A(Q)
30 PRINT TAB(10) "TERRENO Nº";T:PRINT
40 INPUT "FRENTE:";F
50 INPUT "LADO:";L
60 A(T)=F*L
70 T=T+1:PRINT:IF T=Q+1 THEN 90
80 GOTO 30
90 CLS:FOR C=1 TO Q
100 PRINT,,"TERRENO Nº";C;"=";A(C);"MET
ROS QUADRADOS."
110 NEXT C
120 END
```

Ao ser rodado o programa, o computador solicitará, através do vídeo, a quantidade dos terrenos a calcular e as medidas de cada um, dados que serão fornecidos pelo usuário através do teclado. Ato imediato, será exibida no vídeo a lista com as áreas calculadas.

PONTO-E-VIRGULA COMO INSTRUÇÃO

Usados após a definição da instrução PRINT, determinam que a seguinte execução de uma instrução PRINT seja após eles, como nas linhas 30, 40, 50 e 100 do programa apresentado. A instrução não precisa ser repetida após os mesmos, pois é subentendida pelo computador.

3.9 INSTRUÇÕES PRINT USING E LPRINT USING

PRINT USING complementa a instrução PRINT quanto à formatação de dados a serem impressos no vídeo, e LPRINT USING através da impressora.

Sua utilização amplia os recursos de programação, como veremos a seguir:

3.10 PROGRAMA "LISTA DE PREÇOS"

```
10 CLS: INPUT "QUANTOS ARTIGOS (MÁXIMO
10)";Q
20 FOR C=1 TO Q
30 PRINT
40 INPUT "DENOMINAÇÃO DO ÍTEM";D$(C)
50 PRINT
60 INPUT "PREÇO EM CZ$";P(C):REM Digite
 preços com várias casas decimais depoi
s do ponto.
70 NEXT C
80 CLS: LOCATE 12,3: PRINT "LISTA DE PR
EÇOS"
90 PRINT: PRINT
100 FOR C=1 TO Q
110 PRINT D$(C);: PRINT TAB(30) USING "
#####,.##";P(C)
120 NEXT C
130 PRINT,,TAB(6)"PREÇOS FIRMES EM CRUZ
ADOS"
```

Rode o programa, reparando na uniformidade e alinhamento perfeito dos preços no vídeo, graças à instrução PRINT USING da linha 110.

3.11 INSTRUÇÃO PRINT USING FORMATANDO "STRINGS"

A instrução PRINT USING pode formatar inclusive "strings", valendo-se dos seguintes sinais:

"!" (ponto de exclamação entre aspas). Será impressa somente a primeira letra da "string". Exemplo:

```
10 A$ = "MSX"
20 PRINTUSING "!"; A$
```

"\ \" (duas barras invertidas, separadas por n quantidade de espaços entre ambas). A "string" será impressa com número de caracteres igual aos espaços entre barras + 2. Teste o exemplo seguinte:

```
10 C$ = "CARNAVALESCO"
20 PRINTUSING "\     \"; C$
```

"&" (e comercial). Faz imprimir dentro de uma "string" outra ou outras "strings" representas por "&". Exemplo:

```
10 J$ = "JOÃO": M$ = "MARIA"
20 PRINTUSING "& E & SE AMAM"; J$, M$
```

3.12 INSTRUÇÃO PRINT USING FORMATANDO EXPRESSÕES NUMERICAS

São tantos os recursos para formatação de expressões numéricas através da instrução PRINT USING que julgamos conveniente apresentá-los na forma de uma tabela, a fim de não nos estendermos em demasia:

PRINT USING "#"; - Indica a quantidade de cifras que serão impressas e o formato. Exemplos de formatos possíveis: "##", "###" (ou mais cerquilhas), "##.##", "####.##", "########,.##".

Se a expressão tiver menos dígitos do que os indicados na instrução, a impressão será feita com espaços em branco à esquerda. Se a expressão tiver mais espaços do que os indicados, a expressão será impressa com o sinal "%" precedendo-a.
No último exemplo, o sinal de vírgula faz com que seja acrescentada uma vírgula a cada três dígitos à esquerda da pontuação decimal.

PRINT USING "+##"; (ou "#.##+") - Fará imprimir a expressão com sinal + ou - na posição que for indicada na instrução e conforme seja seu valor negativo ou positivo.

PRINT USING "##.##-"; - Fará imprimir os números com sinal - (negativo) em seguida, se a expressão for negativa, e

apenas com um espaço se a expressão for positiva.

PRINT USING "**#####.##"; - Preenche com asteriscos os espaços em branco na frente da expressão.

PRINT USING "$$#####.##"; - Imprime a expressão precedida de $.

PRINT USING "**$#####.##"; - Imprime a expressão precedida de $ e acrescenta asteriscos nos espaços não ocupados.

PRINT USING "##.##^^^^"; - Imprime a expressão em formato exponencial.

Todas as formatações indicadas podem ser combinadas e a quantidade de cerquilhas poderá variar à vontade do usuário.

A instrução pode ainda ser usada na seguinte configuração:

PRINTUSING "PREÇO UNITARIO: CZ$###.##"; 543.247

originando a seguinte impressão: PREÇO UNITÁRIO: CZ$ 543.25.

INSTRUÇÃO LPRINT USING

Todas as formatações indicadas para PRINT USING são válidas para LPRINT USING.

INSTRUÇÃO END

Finaliza a execução de um programa, fazendo-o retornar ao nível de comandos. Fecha arquivos eventualmente abertos, mas não altera o conteúdo de variáveis.

O uso no fim de um programa é opcional, mas necessário se existirem linhas de instruções DATA e sub-rotinas após a mesma.

INSTRUÇÃO STOP

Interrompe a execução de um programa na linha de instruções em que se achar, retornando-o ao nível de comandos. Ao ser executada, o

computador emite a seguinte notação no vídeo:

PAREI EM (nº de linha) (HOTBIT), BREAK IN (nº de linha) (EXPERT).

Ao contrário de END, STOP não fecha arquivos eventualmente abertos. Também não altera o conteúdo de variáveis.

COMANDO CONT

Um programa interrompido por END ou por STOP tem prosseguimento através do comando CONT (tecla F8).

3.13 PROGRAMA "ENDSTOP"

```
10 CLS: DEFINT A,B
20 A=RND(-TIME)*10
30 B=RND(-TIME)*-20
40 PRINT A,B
50 PRINT,,"PRESSIONE AS TECLAS SHIFT +
F8"
60 STOP
70 PRINT A,B
80 PRINT,,"PRESSIONE AS TECLAS SHIFT +
F8"
90 END
100 GOTO 20
```

Rode o programa algumas vezes, observando a ação de END, STOP e CONT.

Troque em seguida as instruções das linhas 60 e 90 e rode novamente o programa. Verifique se ocorre alguma diferença em sua execução e constate a atuação de END, STOP e CONT.

Observação: um programa cuja execução foi interrompida por CTRL + STOP, ou ainda END ou STOP em linhas de programação, pode receber os comandos LIST e PRINT e ter, inclusive, os valores de suas variáveis alterados, continuando sua execução, a partir do ponto de parada, através do comando CONT.

Essa versatilidade de CONT permite colocar a instrução STOP em lugares estratégicos do programa, a fim de poder verificar ou controlar o seu desempenho ou estudar a atuação de determinadas instruções.

Se forem efetuadas alterações nas instruções do programa durante a sua paralisação, porém, o comando CONT não será válido para continuidade de execução.

3.14 INSERINDO NOVAS INSTRUÇÕES NUM PROGRAMA

Linhas de instruções novas ou adicionais podem ser inseridas em qualquer ponto de um programa já pronto que esteja na memória do computador. Essa é uma das razões por que os programas são elaborados normalmente com numeração espaçada.

Quando a numeração é consecutiva e novas instruções devem ser intercaladas, o programa deve ser renumerado através do comando RENUM, a fim de que seja criado o espaço necessário.

O comando RENUM é estudado no capítulo seguinte.

Controles de listagem, de execução e sua análise 4

AUTO - LIST - LLIST - RENUM - DELETE - REM - RUN

Os comandos AUTO, LIST e RUN já foram referidos sumariamente no Capítulo 1 - FAMILIARIZAÇÃO COM O TECLADO. Todavia, dada sua versatilidade, julgamos oportuno repassá-los agora, a fim de conhecer-lhes todos os recursos.

COMANDO AUTO - TECLA DE FUNÇÃO F1

Auto gera automaticamente números de linhas de programas. O número inicial e o incremento são indicados no comando. Exemplo:

AUTO 50,5 / RETURN

iniciará a numeração a partir do número 50 e incrementará cada número subseqüente de linha em 5, cada vez que for pressionada a tecla RETURN.

Se não forem especificados os parâmetros citados, o computador assumirá automaticamente 10,10 como valores iniciais.

Se for adotada numeração AUTOmática quando outras linhas de programas estiverem na memória do computador e houver coincidência de números, aparecerá um asterisco após o número, indicando o fato. Se for pressionada a tecla RETURN imediatamente, a linha já existente será mantida e será gerado um número subseqüente para uma nova instrução.

Para interromper a numeração automática, basta pressionar simultaneamente as teclas CTRL e STOP ao ser gerado um número.

COMANDO LIST - TECLA DE FUNÇÃO F4

Na modalidade direta, sem complementos:

LIST / RETURN

exibe no vídeo a listagem completa do programa existente na memória do computador.

Se a listagem não couber na tela de vídeo, passará "rolando" pelo mesmo e parará no final.

A tecla STOP, pressionada, fará a listagem parar no porto desejado. Pressionada novamente, fará a listagem prosseguir.

LIST 100 - Exibe no vídeo apenas a linha 100 do programa.

LIST -100 - Exibe no vídeo todas as linhas existentes até a de número 100.

LIST 50-150 - Exibe todas as linhas existentes entre os números 50 e 150, inclusive.

LIST 300- - Exibe as linhas a partir do número 300 até a última do programa.

COMANDO LLIST

Atua como LIST, passando a listagem para uma impressora conectada ao microcomputador.

COMANDO RENUM

Renumera as linhas de um programa, inclusive as de desvios GOTO e GOSUB.

O comando deve especificar o novo número a ser adotado, o antigo e o incremento a ser dado, nessa ordem. Exemplo:

RENUM 50,100,10 / RETURN

renumera um programa iniciando com 50 (novo número), a partir de 100

(número antigo do programa existente na memória do computador), com incremento de 10 em cada linha.

Se não forem especificados esses parâmetros no comando, o computador adotará automaticamente 10,10 como NOVO NÚMERO INICIAL e INCREMENTO, renumerando o programa a partir da primeira linha de instruções.

COMANDO DELETE

Elimina da listagem de um programa a linha ou linhas especificadas no comando. Exemplo:

DELETE 70 / RETURN

elimina a linha 70 do programa.

DELETE 100-190 / RETURN

elimina as linhas numeradas de 100 a 190 no programa.

DELETE -200 / RETURN

elimina as linhas desde a inicial até a de número 200.

DELETE. / RETURN

elimina a última linha do programa.

INSTRUÇÃO REM

Permite introduzir comentários na listagem de um programa a fim de elucidar trechos do mesmo.

Não é executável pelo computador e não interfere, pois, na execução do programa. Se for incluída no início de uma linha de instrução, não deverá conter outras instruções. Poderá ser incluída como instrução final de uma linha que contenha diversas outras instruções.

COMANDO RUN

Executa ou "roda" um programa existente na memória do computador. Se especificar um número de linha, o programa será executado a partir dessa linha.

Se não especificar um número de linha, o programa será executado desde o início.

RUN não deverá ser usado se o programa contiver variáveis com valores ou dados atribuídos, pois, antes de executar o programa, procede a um CLEAR na área de variáveis, zerando-as ou apagando seu conteúdo. Deve ser substituído por GOTO |nº de linha| nesse caso.

RUN poderá ser usado para transferir de uma fita cassete para a memória do computador, e fazê-lo rodar automaticamente depois de carregado, um programa gravado no formato ASC II através do comando SAVE. Sua formatação será, nesse caso:

```
RUN "CAS:"
```

para carregar o primeiro programa que encontrar na fita (no formato citado) e executá-lo a partir da primeira linha. Ou será:

```
RUN"CAS:FIPES": 250
```

para carregar um programa de nome indicado e executá-lo a partir da linha especificada.

Leia o Capítulo 14 - GRAVAÇÃO E CARREGAMENTO DE PROGRAMAS EM FITA CASSETE para obter mais detalhes sobre comandos específicos.

COMANDOS TRON e TROFF

O comando TRON exibe no vídeo os números de linhas de um programa na ordem em que são executadas, para análise passo a passo da sua execução. Os números são exibidos entre colchetes.

TRON só é válido nos modos TEXTO 1 e 2 e é de extrema utilidade para depurar programas e entender o "mecanismo" de certas instruções.

TROFF desativa ou desabilita TRON.

Lidando com variáveis 5

5.1 DEFININDO E ATRIBUINDO VALORES E DADOS AS VARIAVEIS

Há vários modos de definir variáveis e atribuir valores e dados a elas.

1º MODO - DIRETO OU IMEDIATO: Através da instrução LET, na forma:

LET A = 10 ou A = 10

para variáveis numéricas. A palavra LET pode ser suprimida, sendo subentendida pelo computador, como no segundo exemplo.

LET A$ = "BRASIL" ou A$ = "BRASIL"

para variáveis alfanuméricas.

As variáveis numéricas aceitam apenas números ou valores como atribuições.

As variáveis alfanuméricas aceitam números e/ou dados, sempre entre aspas e são distinguidas pelo sinal $ (cifrão). Suas atribuições são definidas como "strings" e os números tratados como tais.

2º MODO - INDIRETO: Através da instrução INPUT, que requer a atribuição para a variável por ela definida apenas quando é acionada por um comando direto ou pela execução de uma linha de instrução de um programa. Ao ser acionada por execução de programa, o computador coloca no vídeo uma interrogação, indicando estar aguardando atribuição pa

ra a variável, e só prossegue com a execução do programa depois de a mesma ter sido digitada.

3º MODO - REMOTO OU POR TABELA: Através da instrução READ. Este modo será estudado no Capítulo 9 - DATA, READ, RESTORE.

EXEMPLOS DO MODO DIRETO OU IMEDIATO:

```
LET B = 5  /  LET BA = 1822  /  LET C1 = -7
K = 5  / Q7 = B+C  /  I = C+(RND(-TIME)*10)
```

para variáveis numéricas.

```
LET A$ = "NOVA"  /  LET B$ = "  "  /  C$ = "EM"
D$ = "REPÚBLICA"  /  E$ = "ANDAMENTO" / F$ = "1986"
```

para variáveis alfanuméricas.

Teste os exemplos apresentados com comandos diretos:

```
PRINT B, BA, I  /  PRINT C1, K, Q7
PRINT A$+B$+D$  /  PRINT C$;B$;E$;B$;C$;B$;F$
```

Experimente também INPUT em comando direto:

```
INPUT A, A$  /  RETURN
```

Após surgir o sinal de interrogação, digite um número qualquer e pressione RETURN. Surgirão então dois sinais de interrogação, indicando estar o computador aguardando entrada de dados para a segunda variável definida por INPUT, em razão de terem sido declaradas duas variáveis por uma mesma instrução. Digite os dados para a segunda variável e teste em seguida as atribuições feitas com comandos diretos.

Exemplos do MODO INDIRETO serão apresentados e comentados em programas seqüentes.

5.2 APLICAÇÕES EM PROGRAMAS COMENTADOS

5.3 PROGRAMA "O MSX ADIVINHA SUA IDADE"

```
10 CLS: PRINT,,TAB(8)"O MSX ADIVINHA SU
A IDADE"
20 PRINT,,
30 DEFINT A-Z
40 INPUT "QUAL É O SEU NOME ";N$
50 PRINT,,N$;", DIGITE SUA IDADE:";: IN
PUT I
60 A$="ACERTEI": T$="TENTATIVAS.": C=1
70 A=RND(+TIME)*100+1: PRINT,,SPC(18)A
80 C=C+1: IF C=50 OR C=100 THEN 140
90 IF A=I THEN 100 ELSE 70
100 PRINT,,SPC(5)"VOCE TEM";I;"ANOS. ME
US PARABÉNS!"
110 PRINT,,SPC(8)A$;" EM";C;T$
120 PRINT,,TAB(5)"Para nova adivinhação
, tecla F5."
130 END
140 PRINT,,SPC(4)"Está difícil, mas vou
 continuar."
150 FOR T=1 TO 500: NEXT T: GOTO 70
```

INSTRUÇÃO DEFINT

Define variáveis numéricas como inteiras, o que significa que todas que se enquadrarem nessa categoria serão processadas com desprezo de eventuais frações decimais.

COMENTARIOS SOBRE O PROGRAMA 5.3

LINHA 30: a instrução DEFINT define variáveis de A a Z como inteiras para processamento pelo programa.

LINHA 40: uma forma diferente de introduzir e definir uma variável alfanumérica, em vez de 40 PRINT "QUAL É SUA IDADE?": INPUT N$ /.

LINHA 50: a atribuição feita à variável N$ é usada como parte da pergunta introduzida pela instrução. É definida a variável I, que deverá registrar a idade informada via INPUT, isto é, pelo teclado e

no momento em que é executada a instrução.

LINHA 60: são definidas e recebem atribuições as variáveis A$ e T$. É definida também a variável numérica C, que recebe o valor inicial 1 e que funcionará como contadora.

LINHA 70: é definida a variável numérica A, que recebe um valor obtido por "sorteio", já que é gerado aleatoriamente pelo computador através da função RND, complementada pela da variável do sistema TIME e as funções de multiplicação por 100 e soma de + 1.

LINHA 80: a variável C é incrementada em 1 a cada ciclo do estágio, enquanto a "idade" informada ao computador não coincidir com o valor de A gerado na linha 70. Na mesma linha, a instrução condicional IF... THEN desvia o fluxo de execução do programa para a sub-rotina da linha 140 quando a variável contadora C assume os valores de 50 e 100.

LINHA 90: é feita a verificação do valor de A em relação a I, sendo o fluxo de execução do programa orientado em função da "dedução" feita pelo computador, ou seja: se o valor de A for diferente do de I, a execução retorna à linha 70; se forem iguais, são executadas as linhas 100 a 130, finalizando o programa.

5.4 PROGRAMA "TABUADA NO MSX"

```
10 CLS: PRINT TAB(12)"TABUADA NO MSX"
20 PRINT,,,,"DIGITE UM NÚMERO E TECLE
RETURN";
30 INPUT N
40 PRINT,,,,"TABUADA DE";N;")"
50 M=N*10
60 C=0
70 PRINT
80 FOR L=N TO M STEP N
90 C=C+1
100 PRINT TAB(11)N;"*";C;"=";L
110 NEXT L
120 PRINT,,,,
130 INPUT "MAIS UM NÚMERO (S/N)";R$
140 IF R$="S" OR R$="s" THEN 10
150 PRINT,,,,"AINDA É TEMPO... TECLE F5
"
```

COMENTÁRIOS

LINHA 30: é definida a variável N cujo valor deverá ser atribuído através do teclado quando for executada a linha de instrução em que se encontra.

LINHA 50: é definida a variável M, que deverá ter um valor igual ao de N multiplicado por 10, para funcionar como parâmetro de valor final da instrução da linha 80.

LINHAS 60 a 90: a variável C é definida e recebe o valor 0. Na linha 90 passa a funcionar como variável contadora e é incrementada em 1 a cada ciclo do "laço" formado por FOR e NEXT nas linhas 80 e 110, a fim de produzir os "multiplicandos" da tabuada.

LINHAS 80 a 110: O "laço" constituído pelas instruções FOR e NEXT é incumbido de controlar os 10 ciclos da tabuada, de forma que produzam sempre múltiplos do número assumido pela variável N. A variável I definida no início do "laço" tem a incumbência de incrementar cada ciclo do "laço" em passos (STEP) do valor de N, até atingir o valor de M, "produzindo" os produtos da tabuada sem efetuar qualquer multiplicação, o que, evidentemente, reflete um artifício ou truque, utilizado a título de demonstrar mais um recurso de programação, já que o pequeno programa não tem a finalidade de explorar a capacidade do computador para efetuar operações aritméticas.

5.5 PROGRAMA "MEDINDO E SUBLINHANDO PALAVRAS"

```
10 CLS: REM MEDINDO E SUBLINHANDO PALAV
RAS
20 PRINT,,"DIGITE UMA PALAVRA OU UMA FR
ASE:"
30 PRINT
40 INPUT A$
50 FOR C=1 TO LEN(A$)
60 B$=B$+CHR$(195)
70 NEXT C
80 PRINT
90 PRINT A$: PRINT B$
100 PRINT
```

```
110 PRINT,,"LETRAS E ESPAÇOS =";LEN(A$)
120 A$="": B$=""
130 PRINT,,"Para outra rodada, pression
e 'RETURN'"
140 PRINT,,"Para encerrar, aperte qualq
uer tecla."
150 R$=INPUT$(1): IF ASC(R$)=13 THEN 10
 ELSE END
```

COMENTÁRIOS

LINHA 40: é definida e introduzida a variável A$ que deve receber dados através do teclado por força da instrução INPUT.

LINHA 50: é criado um "laço" repetitivo com o número de ciclos equivalente à quantidade de caracteres que forem atribuídos à variável A$. A quantidade de caracteres de A$ é verificada pela função LEN.

LINHA 60: a variável alfanumérica B$ é definida e sua atribuição é feita de forma que, a cada ciclo do "laço" das linhas 50 e 70, elā receba um caractere "‾", correspondente a CHR$(195). Como o "laço" é dimensionado em função do comprimento de A$, a variável B$ assume a mesma dimensão de A$, incluindo eventuais espaços em branco.

LINHA 90: aqui, a instrução PRINT faz com que A$ e B$ sejam impressas ou escritas no vídeo em linhas diferentes e sucessivas. E como B$ tornou-se uma linha tracejada, por força da instrução da linha 60, funciona como "sublinha" da palavra ou frase digitada. Experimente mudar os dois-pontos (:) da linha 90 por ponto-e-vírgula (;), depois de rodar o programa, e rode-o novamente para perceber a diferença.

LINHA 110: informa a quantidade de caracteres de A$ e, conseqüentemente, a de B$, novamente por meio da função LEN.

LINHA 150: a variável R$ é definida e sua atribuição é feita através da instrução INPUT$, de forma que receba um único caractere, determinado por (1) após ela, através do teclado, mesmo que se tente imputar-lhe uma palavra ou mais, caso em que será assumido apenas o primeiro caractere digitado.

FUNÇÃO ASC

Fornece o código ASC II do primeiro caractere de uma "string". Na linha 150 do programa 5.5 verifica o código de R$ e o compara com o número 13, que é o código correspondente à tecla RETURN. Se for igual, o fluxo de execução do programa é dirigido para a linha 10, iniciando nova rodada. Se for diferente de 13, o programa é encerrado. Essa "decisão" é tomada pelo computador por força das instruções IF/THEN/ELSE, como será visto nos Capítulos 7 e 11.

INSTRUÇÃO INPUT

Permite atribuir valores ou dados a variáveis durante a execução do programa, como visto nos programas 5.3, 5.4 e 5.5.

É importante notar que a instrução INPUT não permite que se use vírgulas e aspas nas atribuições feitas às variáveis. Assim, se uma vírgula for usada numa atribuição, o computador considera como "extra" o que estiver após a vírgula e não registra na variável, escrevendo a seguinte notação no vídeo:

EXTRA ANULADO (HOTBIT) / EXTRA IGNORED (EXPERT).

Experimente no programa 5.5, ao ser solicitada a entrada de uma palavra ou frase, digitar:

"SÃO PAULO, CIDADE LIMPA."

e note que a variável A$ da linha 40 ficará registrada apenas como "SÃO PAULO".

Tal situação poderá ser contornada com o uso da instrução LINE-INPUT.

INSTRUÇÃO LINEINPUT

Permite atribuir a uma variável alfanumérica uma "string" ou conjunto de caracteres de até 254 espaços, incluídos os "em branco". Ao

contrário da instrução INPUT, LINEINPUT permite o uso de vírgulas e de aspas, bem como de quaisquer outros sinais e símbolos.

Substitua a linha 40 do programa 5.5 pela que segue:

```
40 LINEINPUT A$
```

e rode o programa, observando que, ao executar a instrução, o computador exibe apenas o cursor como indicação de que aguarda entrada de dados, em vez do sinal de interrogação apresentado pela instrução INPUT.

Introduza frases variadas com vírgulas e palavras entre aspas e note a diferença.

Importante: se for digitar frases com mais de 200 caracteres, incluídos os intervalos entre palavras, não esqueça de reservar o espaço suficiente, através da instrução CLEAR, comentada a seguir.

Se não for feita a reserva, ao ser excedido o espaço referido pela utilização mencionada, o computador sustará a execução do programa, apresentando no vídeo a seguinte notação:

```
FALTA AREA 'STRING' (HOTBIT)/OUT OF STRING SPACE (EXPERT).
```

INSTRUÇÃO CLEAR

Define ou reserva espaço na memória do computador para armazenamento de variáveis "string" ou alfanuméricas.

Ao ser ligado, o MSX reserva automaticamente apenas 200 espaços para essa finalidade. Quantidades maiores devem ser reservadas através da instrução em foco.

Ao ser acionada, CLEAR zera todas as variáveis anteriormente definidas, fecha todos os arquivos eventualmente abertos e anula todas funções definidas por DEFFN, DEFSNG, DEFDBL, DEFSTR, DEFUSR, DEFINT, razão por que deve ser sempre colocada no início do programa em que tenha de ser usada.

Na função descrita, sua sintaxe é:

```
CLEAR |q|
```

sendo q a quantidade de "bytes" a reservar.

CLEAR pode ser usada também para limitar a área da RAM para utilização de programas em BASIC, de forma que reserve espaço, além dela, para rotinas ou programas em linguagem de máquina. Tal limite deve estar situado entre os seguintes endereços da RAM:

33567 e 62336.

Sua sintaxe é, em tal caso:

CLEAR |q|,|endereço|

podendo ser usada no modo direto ou indireto.

A área disponível na RAM e, indiretamente, o endereço do limite da mesma podem ser verificados através da função FRE.

FUNÇÃO FRE

Fornece a quantidade de "bytes" disponíveis para programas em BASIC ou para variáveis alfanuméricas.

Sua sintaxe é, respectivamente:

PRINT FRE (0)

PRINT FRE ("")

INSTRUÇÃO SWAP

Sua função é trocar o conteúdo de variáveis de mesma categoria. Exemplos num programa:

5.6 PROGRAMA "SWAP"

```
10 CLS: A=10: B=20
20 A$="DEZ": B$="VINTE"
30 PRINT,,"A = ";A;A$
```

```
40 PRINT,,"B = ";B;B$
50 SWAP A,B
60 SWAP A$,B$
70 PRINT,,"A = ";A;A$
80 PRINT,,"B = ";B;B$
```

A instrução SWAP pode ser usada com grande eficiência em programas de ordenação de variáveis numéricas e alfanuméricas.

INSTRUÇÃO DEFDBL

Define ou declara variáveis numéricas como de dupla precisão e as processa com até 14 dígitos. Exemplo:

```
10 DEFDBL A
20 A = 2/6
30 PRINT A
```

INSTRUÇÃO DEFSNG

Define ou declara variáveis numéricas como de simples precisão e as processa com até 6 dígitos. Exemplo:

```
10 DEFSNG A
20 A = 7/3
30 PRINT A
```

INSTRUÇÃO DEFSTR

Define ou declara variáveis numéricas como "strings" e as processa como tais. Exemplo:

```
10 DEFSTR A, B
20 A = "19": B = "86"
30 PRINT A + B
```

FUNÇÃO VARPTR

Fornece o endereço da RAM em que se encontra o conteúdo de qualquer variável ou do primeiro "byte" do bloco de controle de um arquivo.

Exemplos para obter endereços:

- De uma variável definida, A, por exemplo, e com atribuição, A = 20, por exemplo (fora dessas condições o computador retornará à notação de FUNÇÃO ILEGAL):

```
PRINT VARPTR (A)
```

- De uma variável dimensionada, DIM B(15), por exemplo:

```
PRINT VARPTR (B(0))
```

sendo fornecido o endereço da variável dimensionada B de menor índice. Deve ser lembrado que o endereço de variáveis dimensionadas muda sempre que uma variável independente for definida.

- Do bloco de controle de um arquivo:

```
PRINT VARPTR (# n)
```

sendo n o número de arquivo aberto pela instrução OPEN.

VARPTR pode ser usada no modo direto ou indireto. Sua maior utilização é em programas em linguagem de máquina.

Se o endereço indicado por VARPTR for um número negativo, deverá ser somado a 65536 para se obter o endereço real.

5.7 VARIÁVEIS ESPECIAIS

São variáveis próprias do sistema operacional do computador, com funções específicas.

CSRLIN

Fornece a coordenada vertical do cursor, representada por números de linhas da tela de vídeo nos modos TEXTO 1 e 2, variando entre 0 e 23. Como sabemos, nesses modos o vídeo é operado em 24 linhas.

POS

Fornece a coordenada horizontal do cursor, representada por números de colunas da tela nos modos TEXTO 1 e 2, variando de 0 a 39.

LPOS

Fornece a posição do cabeçote da impressora dentro do "buffer" da mesma.

TIME

Fornece o valor do temporizador interno do computador

Durante o processamento de programas, o temporizador é incrementado a cada 1/60 de segundo em uma unidade, variando de 0 a 65535, e pára quando o computador efetua operações de entrada/saída, como gravação ou recuperação de dados em fita cassete.

TIME pode assumir qualquer valor entre 0 e 65535 atribuído diretamente ou através da instrução LET e ser incrementada a partir de tal valor. Exemplo:

```
10 CLS: PRINT "TIME = ";TIME
20 TIME=0: PRINT "TIME = 0"
30 PRINT "30 SEGUNDOS DECORRERÃO..."
40 IF TIME<1800 THEN 40
50 PRINT "TIME = ";TIME
60 PRINT "30 SEGUNDOS JÁ DECORRERAM..."
```

Rode o programa com um cronômetro em mãos para constatar a precisão com que funciona o temporizador do MSX. Faça-o pressionando a tecla F5 quando o ponteiro de segundos do cronômetro estiver bem em cima

do ponto 0 ou de uma outra marca que facilite o controle.

OBSERVAÇÕES FINAIS

- As variáveis podem ser definidas por qualquer letra seguida de outra letra ou de um número.
- Na denominação de variáveis, apenas dois caracteres são significativos para o computador, embora possam ser usados mais de dois.
- Não podem ser usados como nomes de variáveis os que são próprios de comandos, instruções, funções e operadores do BASIC MSX, sendo os mesmos reservados.
- Variáveis numéricas podem ser caracterizadas também pelos seguintes sinais:

  % para declarar uma variável inteira.
  ! para declarar uma variável de simples precisão.
  # para declarar uma variável de dupla precisão.

- Na falta de declaração ou definição, variáveis numéricas são processadas automaticamente com dupla precisão.
- Os valores de variáveis de simples precisão são armazenados e expressos com até seis dígitos.
- Os valores de variáveis de dupla precisão são armazenados e expressos com até quatorze dígitos.
- Variáveis inteiras podem representar números entre -32767 e +32767. Fora dessa faixa, ocasionam um "overflow" no computador, que emite a notação correspondente.

Sub-rotinas 6

As vezes temos de elaborar programas que, para cumprir determinadas tarefas, devem repetir com considerável freqüência uma ou várias instruções.

Nesses casos, recorremos a "laços" ou a sub-rotinas, que, além de evitar linhas repetidas de instruções, agilizam os programas, tornando-os mais práticos e concisos, como também economizam "bytes" na memória do computador.

INSTRUÇÃO GOSUB

No programa apresentado a seguir vamos pôr em prática a instrução GOSUB, que desvia o fluxo de execução do programa para determinada SUB-ROTINA.

Para que funcione com GOSUB, uma sub-rotina deve conter o comando RETURN como última linha de instrução.

6.1 PROGRAMA "ORDENAÇÃO ALFABETICA DE NOMES"

```
10 SCREEN0: WIDTH 40: KEYOFF
20 CLS: PRINT SPC(5): INPUT "Quantos no
mes quer ordenar";Q
30 CLEAR 50*Q: Q=FRE("")/50
40 DIM P$(Q): C=0
50 PRINT
```

```
60 C=C+1: INPUT "Qual é o nome";P$(C)
70 IF C=)2 THEN GOSUB 160
80 PRINT: PRINT TAB(9);
90 INPUT "Outro nome (s/n)";R$
100 IF R$="S" OR R$="s" THEN 50 ELSE 11
0
110 CLS: PRINT,," LISTA DOS NOMES EM O
RDEM ALFABÉTICA:"
120 PRINT: FOR K=1 TO C
130 PRINT TAB(10) P$(K): NEXT K
140 PRINT,,"Para rever lista: 'GOTO 110
' e 'RETURN'."
150 END
160 'SUB-ROTINA DE ORDENAÇÃO
165 FOR I=1 TO C-1
170 FOR J=I+1 TO C
180 IF P$(I))P$(J) THEN SWAP P$(I),P$(J
)
190 NEXT J: NEXT I
200 IF C=Q THEN 110
210 RETURN
```

COMENTÁRIOS

LINHA 10: estabelece o modo TEXTO 1 para o vídeo, com 40 colunas de largura e elimina da última linha as palavras referentes às teclas de função programável.

LINHA 20: é solicitada a quantidade de nomes a ordenar, para atribuição de valor à variável Q, definida na mesma linha, para servir de parâmetro às instruções das linhas 30 e 40.

LINHA 30: a primeira instrução faz a necessária reserva de espaços na área de variáveis, em função da quantidade de nomes determinada através da linha 20, sendo estimados 50 espaços para cada nome. Como a instrução CLEAR procede a uma "limpeza" automática na área antes de fazer a reserva - portanto anulando a variável Q, antes definida -, a segunda instrução da linha redefine Q e lhe devolve o valor originalmente estabelecido.

LINHA 40: a variável P$ é definida e dimensionada para armazenar os no-

mes. A variável C também é definida e lhe é atribuído o valor inicial 0. Funcionará como contadora dos nomes entrados.

LINHA 60: começa a "contar" a entrada de nomes, que é iniciada através de atribuições feitas à variável P$, que passa a ser indexada.

LINHA 70: a partir da entrada do segundo nome, desvia o fluxo de execução do programa para a SUB-ROTINA DE ORDENAÇÃO.

LINHA 90: após o retorno da sub-rotina, o computador consulta através do vídeo se vai haver ou não continuidade de introdução de nomes, o que pode ser feito até o limite estabelecido na linha 20, se a tecla S for pressionada. Em caso contrário, é apresentada no vídeo a lista de nomes, já em ordem alfabética.

LINHA 100: através da instrução IF... THEN, controla a seqüência de execução do programa

LINHAS 110 a 140: são responsáveis pela impressão da lista no vídeo.

LINHA 150: impede que o programa em execução, ao atingi-la, passe a executar novamente a SUB-ROTINA que, em função da linha 200, entraria num "laço" infinito.

LINHAS 160 a 210: constituem a SUB-ROTINA DE ORDENAÇÃO, utilizando a instrução SWAP, notável recurso do BASIC MSX.

Observação: Em computadores que não contam com instrução equivalente a SWAP, a mesma ordenação comandada pela linha 180 teria de ser feita da seguinte forma:

```
180 IF P$(I)>P$(J) THEN 190
190 N$ = P$(I)
200 P$(I) = P$(J)
210 P$(J) = N$
```

Ciclos repetitivos ou laços ("loops") 7

Como falamos no capítulo anterior, a fim de atender à necessidade de elaborar programas que devem repetir instruções determinado número de vezes, ou que demandam dados cuja obtenção depende de repetição freqüente de instruções, recorremos a sub-rotinas ou a "laços", que constituem ciclos repetitivos em programas.

Já tratamos das sub-rotinas no capítulo anterior. Trataremos agora dos "laços", de grande aplicação em programas de arquivo ou fichário, nos quais se enquadram os cadastros, bancos de dados, controles de estoque etc.

Vamos examinar a seguir os dois recursos mais práticos para efetuar ciclos repetitivos: com auxílio de VARIÁVEL CONTADORA e a instrução de desvio GOTO e através das instruções complementares FOR e NEXT.

7.1 CICLO REPETITIVO COM VARIÁVEL CONTADORA

É organizado definindo-se uma variável numérica, C, por exemplo, que a cada ciclo do programa ou do estágio é incrementada em uma unidade. A repetição dos ciclos é comandada pela instrução GOTO e sua finalização é feita pela instrução IF... THEN.

INSTRUÇÃO IF... THEN

Desvia ou altera a seqüência de execução de um programa em conformidade com a dedução lógica feita pelo computador sobre uma proposi

ção condicional.

Se a condição proposta por IF for verdadeira, será executada a instrução seguinte a THEN. Caso contrário, será executada a instrução de linha numerada seguinte ou a que suceder a instrução ELSE, se esta constar após THEN.

IF... THEN é classificada como instrução de desvio condicional.

Sua sintaxe é:

IF |condição| THEN |instrução| ELSE |instrução|

O programa apresentado a seguir ilustra a explanação:

7.2 PROGRAMA "PALPITES PARA A LOTO"

```
10 CLS: PRINT TAB(6)"QUANTOS PALPITES D
ESEJA";: INPUT Q
20 C=0: PRINT
30 P=INT(RND(+TIME)*100)
40 PRINT SPC(16);P
50 C=C+1: IF C=Q THEN 70
60 GOTO 30
70 PRINT,,TAB(10)"OUTRA RODADA? (S/N)":
 R$=INPUT$(1)
80 IF R$="S" OR R$="s" THEN 10 ELSE PRI
NT,,TAB(14)"BOA SORTE!"
90 END
```

Observação: No caso de desejar mais do que 20 palpites, a linha 40 deve ser alterada para:

```
40 PRINT P;"-";
```

COMENTÁRIOS

LINHA 10: determina a quantidade de palpites a serem gerados pelo computador, através da definição da variável Q, que deverá receber um valor por INPUT.

LINHA 20: a variável C é definida e controlará a quantidade de palpites gerados.

LINHA 30: é definida como P uma variável numérica que recebe, a cada ciclo do estágio, um valor gerado aleatoriamente pelo computador através da instrução RND complementada por TIME.

LINHA 50: C é incrementada em 1 a cada ciclo do estágio e conta quantos "palpites" são gerados. Através da instrução IF... THEN nessa mesma linha, o computador verifica se o valor de C se iguala ao de Q e, em caso positivo, desvia a execução do programa para a linha 70, finalizando o ciclo repetitivo. A instrução GOTO é subentendida pelo computador após THEN.

LINHA 60: a instrução GOTO comanda a repetição do ciclo, fazendo a execução do programa retornar para a linha 30, enquanto C for menor do que Q. Se, em vez de desviar para a linha 30, o desvio for para a linha 20, o programa entra num "laço infinito", que pode ser interrompido apenas por um "break" - teclas CTRL e STOP pressionadas simultaneamente - ou desligando o computador.

LINHAS 70 a 90: finalizam ou dão continuidade ao programa. A instrução GOTO é subentendida pelo computador após THEN, como na linha 50.

Observação: Outros exemplos do recurso exposto podem ser consultados e vistos nos programas 5.3 - O MSX ADIVINHA SUA IDADE, e 5.4 - TABUADA NO MSX.

7.3 CICLO REPETITIVO COM FOR E NEXT

INSTRUÇÕES FOR e NEXT

FOR: define e introduz uma variável contadora ou, como também é denominada, de controle, criando ou originando um "laço".

NEXT: elemento final do "laço" criado por FOR e acionador da seqüência de contagem de uma variável contadora definida pela mesma.

FOR e NEXT permitem que uma série de instruções seja realizada dentro de seu elo.

Sua sintaxe é:

```
FOR V = n TO n'
.....
NEXT V
```

sendo V uma variável contadora e n e n' os parâmetros de controle para a mesma.

O ciclo repetitivo organizado com FOR e NEXT funciona nos moldes do anterior, isto é, ao atingir determinada linha de instrução do estágio, retorna a uma linha anterior e executa novamente as instruções contidas no "laço". Porém, depende de menor número de instruções, funcionando e interrompendo-se automaticamente em função dos parâmetros estabelecidos para a variável contadora que o integra.

O "laço" constituído por FOR e NEXT também pode incluir instruções de desvio condicional que interrompem os ciclos, se necessário, como será visto logo mais.

O mesmo ciclo repetitivo organizado no programa 7.2 - PALPITES PARA A LOTO, nas linhas 20 a 60 (= a 5 linhas com seis instruções), pode ser realizado na seguinte forma com FOR e NEXT:

```
20 FOR C = 1 TO Q
30 P = INT(RND(-TIME)*100)
40 PRINT SPC(15);P: NEXT C
```

O programa apresentado a seguir ilustra de maneira prática o uso de "laços" com FOR e NEXT:

7.4 PROGRAMA "FICHÁRIO DE ENDEREÇOS"

```
10 CLEAR 1000
20 FOR C=1 TO 10: PRINT
30 CLS: PRINT SPC(14)"FICHA Nº";C
40 PRINT,,"DIGITE O NOME:": PRINT: INPU
T N$(C)
50 PRINT,,"ENDERECO, BAIRRO, CIDADE:":
PRINT: LINEINPUT E$(C)
60 PRINT,,"CEP-TELEFONE:": PRINT: INPUT
 T$(C)
70 PRINT,,TAB(10)"OUTRA FICHA? (S/N)":
```

```
R$=INPUT$(1)
80 IF R$="S" OR R$="s" THEN 90 ELSE 100
90 NEXT C
100 CLS: PRINT TAB(9)"FICHÁRIO DE ENDER
EÇOS"
110 FOR I=1 TO 10
120 PRINT,,"NOME: ";N$(I)
130 PRINT"ENDEREÇO: ";E$(I)
140 PRINT"CEP-TEL.: ";T$(I)
150 NEXT I
160 PRINT,,"Para rever fichas: 'GOTO 10
0' / RETURN."
170 PRINT,,"Para parar/continuar a list
a, use STOP."
180 END
```

COMENTÁRIOS

LINHA 10: reserva espaço para as variáveis que serão utilizadas na execução do programa.

LINHA 20: instrução inicial de um "laço" ou ciclo repetitivo que, por força do valor final atribuído à variável C, deverá repetir dez vezes a execução das instruções das linhas nele contidas, permitindo, assim, que o fichário tenha dez fichas. Além de contadora, a variável C funciona como índice, quando utilizada com FOR e NEXT, indexando as variáveis alfanuméricas N$, E$ e T$ que compõem as fichas.

LINHAS 30 a 70: instruções para obtenção dos dados que compõem cada ficha, armazenando-os nas variáveis alfanuméricas N$, E$, T$, indexadas a cada ciclo do estágio pela variável C, que as interliga pelo mesmo índice.

LINHA 80: após a execução de cada ciclo do "laço", enseja ao usuário decidir sobre a repetição ou não do próximo, o que lhe permite usar ou não todas as fichas "disponíveis".

LINHA 90: instrução complementar do "laço", a qual comanda e coordena a execução e firalização do mesmo.

LINHAS 110 e 150: formam um segundo "laço", este para exibir no vídeo as fichas existentes no fichário.

LINHAS 120, 130, 140: comandam a exibição dos dados das fichas.

7.5 UM "LAÇO" DENTRO DO OUTRO

A utilização desse recurso é extremamente valiosa em programação. Vários "laços" podem ser "aninhados" ou encadeados de forma a se obter tantas subindexações quantas necessárias a determinada finalidade. Por exemplo: dados de vendas de uma loja de departamentos, com n departamentos, cada departamento com n vendedores.

O programa apresentado a seguir ilustra o exemplo:

7.6 PROGRAMA "VENDAS POR DEPTº COM MEDIA POR VENDEDOR"

```
10 SCREEN 0: WIDTH 40: COLOR 15,1
20 CLS: CLEAR 5000
30 INPUT "Quantos departamentos";Q:PRIN
T
40 INPUT "Qual o nº maior de func. num
deptº";QF
50 DIM D(Q): DIM F(QF): DIM M(Q): DIM T
(Q): DIM V(Q,QF)
60 FOR C=1 TO Q
70 PRINT,,"Quantos funcionários no dept
º";C;:INPUT F(C)
80 NEXT C
90 FOR I=1 TO Q
100 FOR J=1 TO F(I)
110 CLS: PRINT,,"Vendas do vendedor";J;
"do deptº:";I:PRINT: INPUT V(I,J)
120 T(I)=T(I)+V(I,J)
130 M(I)=T(I)/F(I)
140 NEXT J: NEXT I
150 CLS: PRINT"DEPTº/ QTDE FUNC"; TAB(1
8)"VENDAS/MES"; TAB(30)"MEDIA/FUNC": PR
INT
160 FOR C=1 TO Q
170 PRINT "Deptº";C;"/";F(C);"Func ";TA
```

```
B(18)USING"#######,.##";T(C);: PRINT TAB
(30)USING"#######,.##";M(C)
180 NEXT C
190 PRINT,,"P/rever dados digite 'GOTO
150'/ RETURN.      P/nova rodada tecle
F5."
```

COMENTÁRIOS

LINHAS 30 a 50: dimensionam as variáveis que serão utilizadas no programa, em função da quantidade de departamentos e do maior número de funcionários de um departamento.

LINHAS 60 a 80: "laço" simples através do qual é determinada a quantidade de funcionários de cada departamento, registrando os dados nas variáveis indexadas F1, F2 e F3, ou mais

LINHAS 90 e 100: início dos "laços aninhados". O "laço de fora", utilizando a variável I, indexa os departamentos da loja, enquanto o "laço de dentro", que utiliza a variável J, indexa os vendedores de cada departamento. A dimensão do laço de fora é determinada por Q (representando a quantidade de departamentos), enquanto o laço de dentro é dimensionado por F (representando a quantidade de vendedores).

LINHA 110: encarrega-se de obter os dados que devem ser registrados na variável indexada por I e J.

LINHA 120: soma a venda de cada departamento.

LINHA 130: calcula o valor médio de venda de cada deptº por vendedor.

LINHA 140: linha final dos "laços aninhados"; a primeira instrução (= NEXT J) controla a quantidade de vendedores de cada departamento, enquanto a segunda (= NEXT I) controla a quantidade destes.

LINHAS 150 a 180: encarregam-se de imprimir no vídeo os dados obtidos e processados.

Observação: Vimos que os programas apresentados utilizam VARIAVEIS INDEXADAS. Para melhor entendê-las, estude atentamente o Capítulo 8 - DIMENSIONANDO E INDEXANDO VARIAVEIS.

Dimensionando e indexando variáveis 8

No programa 7.4 - FICHÁRIO DE ENDEREÇOS, através da instrução da linha 20, determinamos que o fichário teria dez fichas. Isto porque estabelecemos para a variável que obrigatoriamente deve ser usada com a instrução FOR um valor inicial 1 e um valor final 10 (FOR C=1 TO 10).

Se tivéssemos estabelecido para C um valor final 15, por exemplo, o programa seria executado até o 10º ciclo do estágio e sofreria interrupção no 11º, surgindo no vídeo a notação:

INDICE FORA DO LIMITE (HOTBIT) /
SUBSCRIPT OUT OF RANGE (EXPERT)

significando que o valor 15 estabelecido no "laço" FOR e NEXT excedeu o limite.

Mas que limite foi excedido?

8.1 VARIÁVEL INDEXADA

Como já comentamos, quando uma variável numérica integra as instruções FOR e NEXT, ela é caracterizada como INDICE, sendo assim denominada. E qualquer variável subscrita por esse índice é denominada VARIÁVEL INDEXADA. Qualquer variável pode ser indexada por outros recursos que não o do laço com FOR e NEXT.

Quando não especificamos através de instruções adequadas, o computador permite, automaticamente, que qualquer variável seja utilizada com indexação, desde que o limite máximo do INDICE seja 10.

Se o limite do índice for superior e não houver DIMENSIONAMENTO adequado para utilização de uma variável com indexação, ocorrerá a interrupção mencionada.

Vale a pena repetir: a utilização de variáveis indexadas com índice superior a 10 só é possível se forem previamente dimensionadas.

INSTRUÇÃO DIM

Define conjuntos de variáveis relacionadas por um mesmo nome, reservando espaços para os mesmos na memória do computador.

Um conjunto de variáveis nessa condição é denominado também matriz, vetor ou "array".

As variáveis de um mesmo conjunto são distinguidas por índices e denominadas variáveis indexadas ou subscritas.

Conjuntos com índices iguais ou inferiores a 10 não precisam ser dimensionados, sendo seu processamento assumido automaticamente, conforme explicado acima.

No MSX a indexação de uma matriz de variáveis começa com o índice 0. A indexação pode ser feita de forma direta, ou através de variável contadora, ou através do laço com FOR e NEXT.

Uma matriz pode ter de uma a 255 dimensões, dependendo apenas da memória disponível no computador.

Se forem usadas diversas variáveis dimensionadas dentro de um mesmo laço, o dimensionamento deverá ser igual para todas.

Vamos pôr em prática o que foi dito:

8.2 PROGRAMA "NOVA LISTA DE PREÇOS"

```
10 CLS: SCREEN 0: WIDTH 40
20 DEFSNG A-Z: CLEAR 2000
30 DIM D$(20): DIM C(20): DIM P(20)
40 FOR I=1 TO 20
50 CLS: PRINT"DENOMINAÇÃO DO ÍTEM:":PRI
NT: INPUT D$(I):REM ÍTEM ATÉ 22 ESPAÇOS
```

```
60 PRINT,,"CóDIGO:";: INPUT C(I)
70 PRINT,,"PREÇO:";: INPUT P(I)
80 PRINT,,"Outro ítem? (S/N)":R$=INPUT$
(1)
90 IF R$="S" OR R$="s" THEN NEXT I ELSE
 GOTO 100
100 CLS: PRINT SPC(9)"LISTA DE PREÇOS 0
2/86": PRINT SPC(9)"
  "
110 PRINT: PRINT
120 PRINT"PRODUTO";TAB(23)"CóDIGO";TAB(
35)"PREÇO"
130 PRINT
140 FOR I=1 TO 20
150 IF D$(I)="" THEN 170
160 PRINT D$(I);TAB(24)USING "######";C(
I);:PRINT TAB(31) USING "######,.##";P(I
)
170 NEXT I
180 PRINT,,"  PREÇOS CONGELADOS EM FEVE
REIRO/1986."
190 END
```

COMENTARIOS

LINHA 10: as instruções permitem usar a largura máxima da tela.

LINHA 20: a instrução DEFSNG define como de simples precisão as variáveis numéricas que serão utilizadas no programa, o que permite economizar "bytes" da memória do computador, já que, como vimos, uma variável de simples precisão usa quatro "bytes" apenas, enquanto uma de dupla precisão usa oito. E, não havendo definição, o computador utiliza as de dupla precisão, o que pode ser inconveniente em programas longos.

LINHA 30: as variáveis que armazenam os dados da lista de preços são dimensionadas.

LINHA 40: a instrução FOR define a variável I como índice, criando um laço de 20 ciclos, através dos quais se processará a indexação das variáveis e captação por INPUT dos dados que serão armazenados por elas.

LINHAS 50 a 70: as variáveis do estágio são interligadas e indexadas de acordo com o número de cada ciclo e registram os dados que lhe são imputados através do teclado e que comporão a lista de preços.

LINHAS 80 e 90: permitem continuar até a finalização ou interromper os ciclos do estágio, caso o número de itens para a lista seja inferior ao dimensionado. Em qualquer dos casos, a execução do programa segue na linha 100, a não ser que os ciclos não sejam interrompidos por força das instruções dessas linhas e prossigam normalmente até sua finalização em conformidade com a linha 40.

LINHAS 100 a 160: incumbem o computador de imprimir a lista, sendo que a linha 150 cuida de que não "constem" da mesma os itens inexistentes ou "em branco".

8.3 INTERLIGAÇÃO DE VARIÁVEIS INDEXADAS E CORRELAÇÃO DE DADOS E VALORES

Tomando como base o programa 8.2 - NOVA LISTA DE PREÇOS, verificamos que no primeiro ciclo do laço principal, sendo o valor de I = 1, as variáveis introduzidas através de INPUT são indexadas com o valor de I, por força da indicação feita para cada uma entre parênteses. Assim, D$(I) passa a ser D$(1), C(I) passa a ser C(1), P(I) passa a ser P(1). D$(1) registra um nome e C(1) e P(1) os valores que lhe são imputados no mesmo ciclo do laço, de sorte que esses dados ficarão interligados por variáveis de mesmo índice.

O mesmo ocorre nos demais ciclos e cada grupo de variáveis é numerado sucessivamente até o último ciclo realizado pelo estágio.

Assim, dado o fato de que, em cada ciclo, as variáveis D$, C e P recebem um mesmo número de índice, dizemos que variáveis de mesmo índice são INTERLIGADAS, contendo dados e valores correlacionados, o que facilita a elaboração de listas de preços, tabelas, fichários etc.

É apresentado a seguir outro programa que utiliza os recursos comentados.

8.4 PROGRAMA "BUSCA DE NOMES"

```
10 CLS: PRINT SPC(4): INPUT "Quantos no
mes quer digitar";Q
20 CLEAR 50*Q: Q=FRE("")/50: DIM N$(Q)
30 PRINT,,TAB(6)"Entendido. Pode começa
r!"
40 FOR C=1 TO Q
50 PRINT,,TAB(10)"Digite o";C;:P=POS(1)
: L=CSRLIN
60 LOCATE(P-1),(L): PRINT"O nome:"
70 PRINT: INPUT N$(C)
80 PRINT,,TAB(5)"Lembre-se de que são";
Q;"nomes!"
90 FOR T=1 TO 400: NEXT T: CLS: NEXT C
100 CLS: PRINT,,"OK. Digite o nome que
quer buscar:"
110 PRINT,,: INPUT B$
120 FORC=1 TO Q
130 IF B$=N$(C) THEN 180
140 NEXT C
150 PRINT,,TAB(10)"Nome não encontrado!
"
160 PRINT,,TAB(10)"Outra busca? (s/n)":
 R$=INPUT$(1)
170 IF R$="S" OR R$="s" THEN 100 ELSE 2
10
180 PRINT,,"O nome procurado, "
190 PRINT,,N$(C);","
200 PRINT,,"está registrado na variável
 N$ indexada com o número:";C: GOTO 160
210 PRINT,,"Programa encerrado.": END
```

COMENTÁRIOS

LINHA 10: permite que o usuário determine a quantidade de nomes que entrarão no programa.

LINHA 20: reserva espaço na área das variáveis "string", em função da quantidade determinada de nomes, sendo estimados 50 espaços para cada nome. Na mesma linha é restaurado o valor de Q, anulado ao ser executada a instrução CLEAR, para que DIM dimensione a variável N$.

LINHAS 40 e 90: início e fim do laço repetitivo que solicitará a quantidade Q de nomes que deverão ser digitados.

LINHAS 50 e 60: incumbem-se de solicitar a digitação dos nomes, indicando, ao mesmo tempo, à contagem deles. As variáveis POS e CSRLIN, próprias do sistema operacional do computador, são usadas para determinar a estética de impressão dos dizeres das mesmas linhas.

LINHA 70: introduz a variável N$, indexando-a de acordo com o número do ciclo, a fim de que lhe seja atribuído um nome.

LINHAS 100 a 140: constituem o estágio do programa elaborado para "buscar" um dos nomes colocados na memória do computador através da variável N$ indexada.

O nome que se deseja "buscar" - ou saber de existe registrado na memória do computador - é atribuído à variável definida como B$.

O laço das linhas 120 a 140 faz com que o computador compare o nome que consta em B$ com os de N$(de 1 a Q).

Se encontrar em N$(de 1 a Q) um nome igual, executará em seguida a linha 180 do programa e seqüentes. Caso contrário, será executada a linha 150 e seguintes até 170, a partir da qual seguirá para a de número 100 ou 210, conforme determinar a resposta dada pelo usuário, através do teclado, à pergunta colocada no vídeo pela linha 160.

INSTRUÇÃO ERASE

Elimina ou redimensiona um conjunto de variáveis indexadas tipo matriz ou "array".

A instrução ERASE tem acesso direto a um conjunto ou matriz existente na área de variáveis, sem interferir ou destruir os registros de outros conjuntos.

Se houver necessidade de eliminar ou redimensionar um conjunto de variáveis tipo matriz num programa em execução, por exemplo, sem que os registros e arranjos já feitos na área sejam destruídos, poderá ser usada a instrução ERASE em comando direto. Bastará interromper o programa com CTRL + STOP e fazer o comando. Por exemplo:

```
ERASE M$ / RETURN
```

e, em seguida:

```
DIM M$(15) / RETURN
```

caso M$ tenha sido dimensionada para 12 índices e o programa deva ter execução com M$ dimensionada para 15 índices.

Naturalmente, em tal caso, a continuação do programa deverá ser comandada através de CONT ou GOTO |nº de linha| para que os registros já feitos nas demais variáveis sejam preservados. Vale a pena lembrar que o comando RUN procede a um CLEAR na área de variáveis antes de dar início à execução de um programa.

A instrução ERASE poderá ser usada estrategicamente em linhas de instruções de programas, com finalidades diversas.

Tentar redimensionar um conjunto de variáveis num programa, sem o concurso de ERASE, causa interrupção na execução, fazendo o computador emitir a seguinte mensagem de erro:

```
DIM REDEFINIDO EM |nº de linha| (HOTBIT) /
                          REDIMENSIONED ARRAY (EXPERT)
```

O programa apresentado a seguir ilustra a explicação:

8.5 PROGRAMA "ERASE"

```
10 DIM M$(12)
20 FOR C=1 TO 12
30 READ M$(C)
40 PRINT M$(C)
50 NEXT C
60 DIM M$(4)
70 FOR C=1 TO 4
80 READ M$(C)
90 PRINT M$(C)
100 NEXT C
110 DATA JAN,FEV,MAR,ABR,MAI,JUN,JUL,AG
O,SET,OUT,NOV,DEZ,PRIMAVERA,VERÃO,OUTON
O,INVERNO
```

Rode o programa e note que o mesmo será interrompido com a mensagem de erro citada.

Altere então a linha 60 para:

```
60 ERASE M$: DIM M$(4)
```

e rode novamente o programa, que apresentará agora todos os dados da tabela da linha 110 DATA no vídeo.

Data, Read, Restore 9

9.1 MODO REMOTO OU POR TABELA PARA ATRIBUIÇÕES A VARIÁVEIS

No Capítulo 5 - LIDANDO COM VARIÁVEIS, referimo-nos a um terceiro modo de atribuir dados ou valores a variáveis, denominando-o modo REMOTO ou POR TABELA. Logo veremos a razão de tal denominação.

As instruções que constituem o título deste capítulo propiciam recursos de programação que devem ser utilizados sempre que o programa tenha de recorrer a dados constantes - numéricos ou alfanuméricos - em quantidade elevada e diversificada.

Os dados em tais condições são colocados numa tabela - INSTRUÇÃO DATA - e lidos pelo computador, um de cada vez, seqüencialmente, e somente uma vez, através da INSTRUÇÃO READ, sempre que esta constar, em sintaxe adequada, de uma linha de instrução no programa.

Quando os dados se esgotarem, a tabela poderá ser restabelecida pela INSTRUÇÃO RESTORE, para nova leitura.

Para que a leitura da tabela se processe, é necessário que a instrução READ defina uma variável, a qual, assimilando o dado lido, recebe assim sua atribuição.

Podem ser definidas diversas variáveis para a leitura de uma tabela: numéricas ou alfanuméricas, uma única variável ou variáveis indexadas.

Assim como as variáveis são utilizadas para "fazer funcionar" as instruções DATA, READ, RESTORE, estas mesmas instruções servem para definir e atribuir dados a variáveis, como veremos a seguir:

Digite: 10 DATA JAN
20 READ A$: PRINT A$

Tecle F5 e note que o computador escreverá no vídeo apenas JAN.

Digite: PRINT A$

Tecle RETURN e note que A$ = JAN.

Altere agora a linha 10 para:

10 DATA JAN, FEV

Tecle F5 e note que não houve mudança no resultado do programa.

Digite: PRINT A$

Tecle RETURN e verifique que A$ continua = JAN.

Acrescente agora a seguinte linha ao programa:

30 GOTO 20

Tecle F5 e note que, além de escrever no vídeo

JAN
FEV

o computador informará:

SEM 'DATA' EM 20 (HOTBIT) / OUT OF DATA IN 20 (EXPERT)

porque, dando continuidade à execução do programa e retornando à linha 20, onde não encontra mais dados para ler (lembre-se de que cada dado é lido apenas uma vez), o computador "pára" e informa por quê.

Verifique agora o que a variável A$ registra:

Digite: PRINT A$ / RETURN .

A variável A$ agora é = FEV, o que significa que uma mesma variável pode ler todos os dados de uma tabela, mas "guardará" apenas o último dado lido.

Continue a experiência, alterando a linha 10 para:

10 DATA JAN, FEV, MAR, ABR, MAI, JUN, JUL, AGO, SET,
OUT, NOV, DEZ

Mantenha as linhas 20 e 30 e tecle F5.

Constate que a variável A$ permitiu a leitura e impressão de todos os dados da tabela, sozinha, e que sua atribuição final é = DEZ .

A experiência demonstra que, ao longo de um programa, se for necessário usar dados de uma tabela (DATA), basta recorrer à instrução READ complementada por uma variável.

Na linha do programa em que for colocada a instrução READ, ela buscará o primeiro dado ainda não lido da tabela e o colocará à disposição da instrução que a segue, para execução. Devemos lembrar apenas que dados numéricos devem ser lidos por variáveis numéricas e dados alfanuméricos por variáveis alfanuméricas.

Quando todos os dados da tabela forem lidos e houver necessidade de relê-los, em outro estágio do programa, por exemplo, deverá ser usada a instrução RESTORE no início do novo estágio. Tal instrução "restaura" os dados da tabela.

Na experiência citada usamos uma única variável para que o computador processasse as instruções DATA e READ e, eventualmente, RESTORE.

Vamos agora usar as mesmas instruções para definir e atribuir dados a variáveis.

Digite:

```
10 DATA JAN, FEV, MAR, ABR, MAI, JUN
20 READ A$: PRINT A$
30 READ B$: PRINT B$
40 READ C$: PRINT C$
50 READ D$: PRINT D$
60 READ E$: PRINT E$
70 READ F$: PRINT F$
```

Tecle F5 e constate facilmente como as variáveis A$, B$, C$, D$, E$ e F$ foram definidas e receberam atribuições.

Vamos agora usar o mesmo processo para definir uma variável indexada, subscrevê-la certo número de vezes, seis, por exemplo, e atribuir a cada subscrita um dado de uma tabela.

Digite:

```
10 DATA LINDO, DIA, HOJE, MAS, MUITO, Q
UENTE
20 READ A$(1): PRINT A$(1)
30 READ A$(2): PRINT A$(2)
```

```
40 READ A$(3): PRINT A$(3)
50 READ A$(4): PRINT A$(4)
60 READ A$(5): PRINT A$(5)
70 READ A$(6): PRINT A$(6)
```

Tecle F5 e constate o que foi afirmado.

Acrescente agora ponto-e-vírgula (;) no final de cada uma das linhas numeradas de 20 a 60 e tecle F5.

Depois acrescente um espaço após cada palavra da linha 10, de maneira que fique assim:

```
10 DATA LINDO ,DIA ,HOJE ,MAS ,MUITO ,QUENTE
```

Tecle F5 e constate que também o espaço passa a fazer parte da atribuição feita às variáveis, agora indexadas.

NOTA: Não obstante ser o CRUZADO a nova moeda em vigor no Brasil na data de publicação deste livro, o programa apresentado a seguir mantém os dados monetários no padrão CRUZEIRO e as menções às atuais OTNs na forma anterior, ORTN, como referência dos valores e das desvalorizações da moeda substituída durante os últimos seis meses de sua vigência.

9.2 PROGRAMA "CONVERSÃO DE CR$s EM ORTNs"

```
10 DATA SET,53.437,OUT,58.3,NOV,63.547,
DEZ,70.613,JAN,80.047,FEV,93.039
20 CLS: SCREEN 0: WIDTH 40
30 CLEAR 2000: COLOR 15,1
40 FOR C=1 TO 6
50 PRINT,,TAB(12);"MES: ";:READ A$(C):
PRINT A$(C)
60 PRINT,,"Importância em Cr$:";:INPUT
V(C)
70 READ A(C):G(C)=V(C)/A(C)
80 T=T+G(C): TM=T/C
90 NEXT C
100 CLS: PRINT,,SPC(2);"TABELA DE CONVE
RSÃO DE CR$s EM ORTNs"
```

```
110 PRINT
120 PRINT,,"MES";TAB(10)"IMPORT";TAB(25
)"ORTN";TAB(34)"GORTN": REM (GORTN=GANH
O EM ORTN)
130 PRINT
140 FOR C=1 TO 6
150 PRINT A$(C);TAB(6) USING "######,.##
#";V(C);
160 PRINT TAB(19) USING "######,.###";A(
C);
170 PRINT TAB(33) USING "###.##";G(C)
180 NEXTC
190 PRINT,,TAB(7) USING "MÉDIA MENSAL E
M ORTN:###.##";TM
```

COMENTARIOS

LINHA 10: elaboração da tabela que contém nomes de seis meses e valores das OTNs correspondentes. Os valores citados são os reais referentes aos últimos seis meses de vigência da moeda CRUZEIRO, substituída pela atual moeda CRUZADO. Os dados são separados por vírgulas para que sejam lidos separadamente, já que os meses serão atribuídos a variáveis alfanuméricas e os valores a variáveis numéricas, para fins de cálculos que serão efetuados pelo computador durante a execução do programa.

LINHA 40: início do laço que permitirá indexar as variáveis para leitura e assimilação dos dados da tabela.

LINHA 50: define e indexa a variável A$, que procederá à leitura dos dados alfanuméricos da tabela, assimilando-os como atribuições.

LINHA 60: define e indexa a variável numérica V, que assimilará através de INPUT as importâncias em CR$ que deverão ser convertidas em OTNs.

LINHA 70: define e indexa a variável A, que efetuará a leitura dos valores da tabela, assimilando-os como atribuições. Define e indexa a variável G, que assimilará o valor do quociente da importância imputada a V, dividida pelo valor da OTN assimilado por A.

LINHA 80: define a variável T e lhe confere a função de "somadora" dos

valores de G. Define também a variável TM, que registra o quociente de T dividido por C, que reflete a média do total acumulado dividido pelo número de meses.

LINHA 90: instrução final do primeiro laço do programa.

LINHAS 100 a 190: Produzem e imprimem no vídeo a tabela final, em padrão monetário uniforme, com todos os dados processados. Note a singularidade de instrução permitida por PRINT USING na linha 190.

Manipulação de "Strings" 10

A manipulação de "strings" (conjunto de caracteres, letras ou símbolos) impõe-se como uma necessidade em programação, com finalidades diversas.

Por essa razão, a linguagem BASIC dispõe de funções e instruções tais como:

ASC, CHR$, INKEY$, INPUT$, INSTR, LEFT$, LEN, LINEINPUT, MID$, RIGHT$, SPACE$, STR$, STRING$, VAL,

usadas e exemplificadas nos seus mais variados usos neste livro.

Vamos agora agrupá-las e repassá-las ligeiramente, na primeira parte deste capítulo, a título de exercício e prática:

ASC - Obtenha os códigos dos caracteres e símbolos do MSX com o seguinte programa:

```
10 PRINT "Digite um caractere e tenha s
eu código:"
20 INPUT A$
30 PRINT,,"O código de ";A$;" é";ASC(A$
)
40 GOTO 20
```

CHR$ - Examine no vídeo todos os caracteres do MSX com o seguinte programa:

```
10 SCREEN 1: WIDTH 30
20 FOR C=1 TO 31
30 PRINT CHR$(1);CHR$(C+64);" ";
40 NEXT C
50 FOR C=32 TO 255
60 PRINT CHR$(C);" ";
70 NEXT C
```

Observação: Foi adotado o modo TEXTO 2 (SCREEN 1) para exibição dos caracteres gráficos como eles são efetivamente. No modo TEXTO 1 alguns caracteres são parcialmente cortados. Note também na linha 30 como devem ser as instruções para obter os primeiros 31 caracteres do MSX.

INKEY$ - Perceba como esta função verifica se alguma tecla é usada ou pressionada durante sua atuação. INKEY$ verifica incessantemente todos os circuitos do teclado e, se alguma tecla é pressionada, aponta imediatamente. Constate com o pequeno programa que segue:

```
10 PRINT INKEY$;
20 P=POS(0): IF P=10 THEN 40
30 GOTO 10
40 PRINT " - FIM DE PROGRAMA - "
```

Pressione a tecla F5 e, após surgir a palavra RUN no vídeo, pressione aleatoriamente algumas teclas.

INKEY$ não apresenta no vídeo nenhum sinal que indique estar o computador aguardando entrada de dados.

Para que o programa não se estenda indefinidamente ou tenha que ser interrompido por CTRL + STOP, a linha 20 faz com que seja sustada sua execução quando a impressão dos caracteres das teclas pressionadas atinge a décima coluna da tela.

INPUT$ - Limite o tamanho das palavras em "strings" ou variáveis alfanuméricas:

```
10 PRINT "DIGITE SEIS MESES DO ANO:"
20 FOR C=1 TO 6
30 A$(C)=INPUT$(3)
40 PRINT A$(C): NEXT C
```

Note que, ao digitar a terceira letra, a função INPUT$ já considera a "string" completa e a imprime no vídeo, porque seu parâmetro foi determinado como três, entre aspas na linha 30.

A função INPUT$ pode ser analisada através de programa idêntico ao exemplificado para INKEY$, apenas mudando a linha 10 como segue:

```
10 INPUT$(1);
```

Ao ser acionada, INPUT$ coloca o cursor no vídeo, indicando que aguarda entrada de dados via teclado.

As funções INKEY$ e INPUT$ são consideradas FUNÇÕES DE ENTRADA, como INPUT.

A diferença essencial entre INKEY$ e INPUT$ é que a primeira só aceita um caractere por vez, enquanto a segunda admite tantos quantos forem determinados por seu parâmetro, usado obrigatoriamente entre aspas após ela.

INSTR - Conheça o número da posição inicial de "sub-strings" dentro de uma "string":

```
10 A$="CLAMOROSAMENTE"
20 PRINT INSTR(A$,"AMOR")
30 PRINT INSTR(A$,"MORO")
40 PRINT INSTR(A$,"MENTE")
50 PRINT INSTR(A$,"CLAMO")
```

LEFT$ - Separe partes de uma "string", do lado esquerdo para o lado direito, e forme com elas novas "strings":

```
10 A$="ARDOROSAMENTE"
20 B$=LEFT$(A$,5):PRINT B$
30 C$=LEFT$(A$,8):PRINT C$
```

LEN - Conheça o comprimento (em "espaços") de nomes ou palavras. Note que a função LEN "mede" também os espaços "em branco":

```
10 PRINT "Digite um nome ou uma frase:"
20 INPUTA$
30 PRINT,,"Comprimento do escrito:";LEN
```

```
(A$);"espaços."
40 PRINT: PRINT
50 GOTO10
```

LINEINPUT - Transforme em atribuição de uma variável alfanumérica qualquer "string" com até 254 espaços:

```
10 CLEAR 1000
20 PRINT "Digite o que quiser até 254 e
spaços:"
30 LINEINPUT A$
40 PRINT A$
50 GOTO 20
```

MID$ - Proceda a um "fatiamento" de palavras ou "strings":

```
10 A$="UNÍSSONO"
20 B$=MID$(A$,5,2)
30 C$=MID$(A$,3,4)
40 D$=MID$(A$,4,1)+MID$(A$,6,1)+MID$(A$
,4,1)
50 PRINT B$
60 PRINT MID$(A$,5,4)+", "+B$+" "+C$
70 PRINT D$+", ESTOU COM "+MID$(A$,5,4)
```

A função MID$ permite também substituir parte de uma "string" por outra:

```
10 A$="ENFORCANDO"
20 B$="TORT"
30 MID$(A$,3,4)=B$
40 PRINT A$
```

RIGHT$ - Selecione partes de uma "string", do lado direito para o esquerdo, e forme com elas novas "strings":

```
10 A$="AMOROSAMENTE"
20 B$=RIGHT$(A$,5): PRINT B$
30 PRINT RIGHT$(A$,4)
40 C$=RIGHT$(A$,11)
50 PRINT C$
```

SPACE$ - Crie "strings" feitas apenas com espaços:

```
10 A$=SPACE$(10): B$="PRATICANDO"
20 PRINT A$; B$
30 PRINT B$; A$; "COM STRINGS"
40 B$=B$+A$+"VOU APRENDENDO.":PRINT B$
```

STR$ - Transforme valores ou números em "strings":

```
10 PRINT "DIGITE DOIS NÚMEROS:": PRINT
20 INPUT A,B
30 PRINT,,"SOMA DOS DOIS NÚMEROS:";A+B
40 A$=STR$(A): B$=STR$(B)
50 PRINT,,"SOMA DOS DOIS TRANSFORMADOS
EM STRINGS:":PRINT A$+B$
```

STRING$ - Produza "strings" de caracteres e do tamanho que desejar (até 255 espaços):

```
10 A$=STRING$(30,45):REM 30=QTDE., 45=C
óD. DO CARACTERE
20 B$=STRING$(20,35)
30 C$=STRING$(15,199)
40 PRINT A$: PRINT B$: PRINT C$
```

VAL - Obtenha valores de "strings", se eles de fato existirem para o MSX:

```
10 A$="47.55 CZ$ P/KG"
20 B$="P/KG CZ$ 47.55"
30 C$="15 1986SP18"
40 PRINT A$;"/ VAL =";VAL(A$)
50 PRINT B$;"/ VAL =";VAL(B$)
60 PRINT C$;"/ VAL =";VAL(C$)
```

10.1 CONCATENAÇÃO DE "STRINGS"

Chamamos de concatenação de "strings" a operação de juntá-las inteiras ou em partes, de forma a alterá-las ou elaborar outras, para as

mais diversas finalidades. Na "passagem" por MID$ e SPACE$, tivemos exemplos de concatenação. Vejamos outro:

```
10 CLS: A$ = " NÃO "
20 B$ = " LATE "
30 C$ = " MORDE "
50 E$ = D$+B$+A$+C$: PRINT E$
60 PRINT,,"Compare a frase de cima com
a seguinte:"
70 PRINT,,D$+C$+A$+B$
```

10.2 COMPARAÇÃO DE "STRINGS"

As "strings" podem ser comparadas quanto à ordem alfabética. Tal recurso permite fazer a ordenação de palavras, nomes e até frases, tornando possível a "procura" ou "busca" de dados em programas de arquivo ou outros:

10.3 PROGRAMA "COMPARANDO STRINGS"

```
10 CLS: PRINT,,"DIGITE UMA PALAVRA:": I
NPUT A$
20 PRINT,,"DIGITE MAIS UMA PALAVRA:": I
NPUT B$
30 PRINT,,"Alfabeticamente..."
40 IF A$=B$ THEN 90
50 IF A$<B$ THEN PRINT,,A$ ELSE PRINT,,
B$
60 PRINT,,SPACE$(15);"é a palavra prece
dente."
70 PRINT,,"Tecle F5 para novas comparaç
ões."
80 END
90 PRINT,,"As palavras são iguais.": GO
TO 70
```

10.4 PROGRAMAS COMENTADOS

Os programas apresentados a seguir utilizam os principais recursos examinados neste capítulo:

10.5 PROGRAMA "LIMITANDO O TAMANHO DE STRINGS"

```
10 CLS: CLEAR 3000: INPUT "Quantas pala
vras quer escrever";Q: PRINT
20 INPUT "Quantos caracteres por palavr
a";T
30 DIM A$(Q)
40 FOR C=1 TO Q
50 PRINT,,TAB(7)"Digite uma das palavra
s:": PRINT
60 INPUT A$(C)
70 IF LEN(A$(C))>T THEN GOSUB 140
80 PRINT,,"Com";T;"caracteres a palavra
 fica:"
90 PRINT,,TAB(7)A$(C)
100 PRINT TAB(7) STRING$(LEN(A$(C)),195
)
110 NEXT
120 PRINT,,TAB(7)"Para nova rodada tecl
e 'RETURN'": R$=INPUT$(1)
130 IF ASC(R$)=13 THEN RUN ELSE END
140 T$=LEFT$(A$(C),T)
150 A$(C)=T$: T$=""
160 RETURN
```

Observação: pode-se usar na linha 140, com a mesma finalidade, a instrução MID$, na seguinte configuração:

```
140 T$ = MID$(A$(C),1,T)
```

onde 1 determina o primeiro caractere da palavra, a partir do qual será feita a seleção de letras, na quantidade determinada por T.

COMENTÁRIO

LINHA 70: através de LEN é feita a verificação se o tamanho da palavra digitada é maior do que o determinado na linha 20. Em caso positivo, a execução do programa é desviada para a sub-rotina iniciada na linha 140, onde a palavra é "cortada", a fim de se enquadrar no tamanho especificado pela variável T. Como a instrução LEFT$ controla a "string" a partir do lado esquerdo, o "corte" é efetuado na parte final da palavra.

10.6 PROGRAMA "IMPRIMINDO STRINGS POR PARTES"

```
10 M$="JANFEVMARABRMAIJUNJULAGOSETOUTNO
VDEZ"
20 N$="/01/01/03/04/05/06/07/08/09/10/1
1/12"
30 D$=" 31 28 31 30 31 30 31 31 30 31 3
0 31"
40 CLS: FORC=1 TO 36 STEP 3
50 PRINT MID$(M$,C,3);" MES ";
60 PRINT MID$(N$,C,3);" COM ";
70 PRINT MID$(D$,C,3);" DIAS."
80 T=T+VAL(MID$(D$,C,3))
90 NEXT C
100 PRINT,,"O ANO TEM:";T;"DIAS."
110 PRINT,,"JAN começa no";INSTR(M$,"JA
N");"º lugar da 'string' M$."
120 PRINT"DEZ começa no";INSTR(M$,"DEZ"
);"º lugar da mesma."
```

COMENTÁRIOS

LINHA 40: por força da instrução STEP 3 no laço iniciado na mesma linha, a variável contadora C é incrementada em 3 a cada ciclo. Como integra a instrução MID$ das linhas 50, 60 e 70, faz com que o controle desta avance 3 passos a cada ciclo do laço, o que permite à instrução PRINT exibir no vídeo, a cada execução, 3 caracteres consecutivos de cada "string" por vez, em ordem sucessiva.

O valor assumido por C a cada ciclo determina a posição ou ponto de início de ação sobre a "string", enquanto o dígito 3, também integrante da instrução, determina a quantidade de caracteres a selecionar para exibição no vídeo ou impressão.

LINHA 80: obtém o "valor" (VAL) das "sub-strings" D$ e acumula na variável T, para execução da linha 100.

10.7 PROGRAMA "LETREIRO LUMINOSO CORRENTE"

```
10 SCREEN 1: WIDTH 32: KEYOFF
20 PRINT"DIGITE UMA FRASE (MÁX. DE 30 E
SPAÇOS). COMECE COM UM EM BRANCO."
30 PRINT
40 LINEINPUT A$
50 C=LEN(A$): CLS
60 LET B$=RIGHT$(A$,C-1)+LEFT$(A$,1)
70 A=A+1: IF A=14 THEN A=1
80 COLOR A,15,A-1
90 LOCATE 0,10: PRINT STRING$(32,198)
100 LOCATE 0,12: PRINT B$: A$=B$
110 LOCATE 0,14: PRINT STRING$(32,215)
120 FOR I=1 TO 50: NEXT I: GOTO 60
```

COMENTÁRIOS

LINHA 50: mede a frase para a função da linha 60.

LINHA 60: faz a troca entre a primeira e a última letra da frase ou da "string" atribuída à variável A$, passando a nova composição para a variável B$. A cada ciclo do laço (linhas 60 e 120) é tirada a letra inicial da frase e colocada no fim da mesma.

LINHA 70: define e controla os valores da variável A, mudando-os a cada ciclo do laço das linhas 60 e 120, a fim de que funcionem como guias de cores na linha 80.

LINHAS 90 e 110: são responsáveis pela moldura do "letreiro".

LINHA 100: imprime o "letreiro" e efetua a permuta das variáveis A$ e B$, tornando A$ igual à última forma assumida por B$, para que

se reinicie o ciclo de troca de letras da linha 60, de forma que o luminoso continue "corrente".

Finalmente, um programinha que relembra uma brincadeira bastante conhecida que se fazia com a caixa de fósforos de marca GRANADA:

```
10 A$="GRANADA"
20 PRINT,,"Uma vez fui a ";A$
30 PRINT,,"Lá, conheci ";MID$(A$,3,3)
40 PRINT,,"Com ela gastei toda ";LEFT$(
A$,5)
50 PRINT,,"Voltei sem ";RIGHT$(A$,4)
```

INSTRUÇÃO WIDTH

Sua função é determinar a quantidade de colunas para impressão nos modos TEXTO 1 e 2, podendo variar de 1 a 40 no modo 1 e de 1 a 32 no modo 2.

O programa apresentado a seguir dá uma idéia de sua ação:

10.8 PROGRAMA "ESPREMENDO STRINGS"

```
10 CLS: A$="BRASIL, PÁTRIA BONDOSA, GRA
NDIOSA, AMADA"
20 WIDTH 40: PRINT A$: GOSUB 100
30 WIDTH 30: PRINT A$: GOSUB 100
40 WIDTH 20: PRINT A$: GOSUB 100
50 WIDTH 10: PRINT A$: GOSUB 100
60 WIDTH 5:  PRINT A$: GOSUB 100
70 WIDTH 3:  PRINT A$: GOSUB 100
80 WIDTH 40: PRINT,,A$
90 END
100 FOR A=1 TO 1500: NEXT A
110 RETURN
```

A instrução WIDTH é válida também no modo direto. Exemplo:

```
WIDTH 40 / RETURN
```

Tomada de decisões pelo computador

11

Dizem que os computadores não "pensam", o que é verdade se o significado de pensar for considerado na sua acepção mais pura.

Mas, se analisarmos determinadas ações dos computadores e aceitarmos o sofisma de que "é preciso pensar para decidir", torna-se duvidosa aquela assertiva, principalmente quando observadas certas atitudes e ações de muitos seres humanos - alguns até notáveis, talvez em conseqüência das mesmas...

A verdade é que, sob determinadas condições, o computador pode tomar decisões absolutamente corretas, como veremos a seguir. E a explicação para esse fato é que a linguagem BASIC conta com determinadas instruções tão poderosas - exemplo das quais IF... THEN - que, usadas em expressões ou instruções condicionais, possibilitam ao computador desenvolver ações que equivalem a "tomar decisões", na mais ampla acepção da expressão, induzindo-nos a crer que "a máquina pensa mesmo".

Faça algumas experiências com os pequenos programas apresentados a seguir, analise os seus resultados e pense se "a máquina não pensa mesmo".

11.1 PROGRAMA "NÚMERO PAR OU ÍMPAR"

```
10 CLS:LOCATE 5,10
20 INPUT "DIGITE UM NÚMERO ATÉ 32767";N
30 LOCATE 12,14
40 IF N MOD 2=0 THEN PRINT"O NÚMERO É P
AR" ELSE PRINT"O NÚMERO É ÍMPAR"
50 PRINT
```

```
60 FOR C=1 TO 1000: NEXT C
70 GOTO 10
```

COMENTÁRIOS

LINHA 20: é definida a variável numérica N, que deverá receber um número através de INPUT.

LINHA 40: através da função MOD o computador "analisa" se o número imputado a N é par ou ímpar e informa de acordo com sua "dedução".

11.2 PROGRAMA "ADIVINHAÇÃO"

```
10 FOR C=1 TO 300: NEXT C
20 CLS: A$="ADIVINHOU": B$="ERROU": C$=
"DIGITE DE NOVO. FORA DE FAIXA."
30 LOCATE 7,10
40 PRINT"DIGITE UM NÚMERO DE 1 A 5": IN
PUT A: IF A)5 THEN GOTO 90
50 B=INT(RND(-TIME)*5)+1
60 LOCATE 14,14
70 IF A=B THEN PRINT A$ ELSE PRINT B$
80 GOTO 10
90 PRINT,,C$: GOTO 10
```

COMENTÁRIOS

LINHA 20: as variáveis A$, B$ e C$ são definidas e recebem atribuições que "ajudarão" o computador a informar sobre sua "decisão" ou "conclusão".

LINHA 40: a variável A é definida para receber um valor de 1 a 5 através de INPUT. O número imputado a A é "analisado" pelo computador através da instrução IF. Se for maior que 5, o computador "decide" que o programa deve ser desviado para a linha 90, informa a ocorrência - imprimindo no vídeo o conteúdo da variável C$ - e reinicia a execução do programa.

LINHA 50: é gerado pelo computador um número aleatório (até 5) para B.

LINHA 70: o computador "analisa" os valores de A e B e "decide" se são iguais ou diferentes e informa sua conclusão valendo-se de uma das variáveis A$ ou B$.

11.3 PROGRAMA "COMPARAÇÃO"

```
10 CLS: LOCATE 0,10: PRINT "DIGITE 3 NU
MEROS SEPARADOS POR VÍRGULAS"
20 PRINT
30 INPUT A,B,C: PRINT
40 IF A=B AND A=C THEN PRINT "A = B / A
 = C"
50 IF A)B AND A)C THEN PRINT "A ) B / A
 ) C"
60 IF A(B AND A(C THEN PRINT "A ( B / A
 ( C"
70 IF A)B AND A(C THEN PRINT "A ) B / A
 ( C"
80 IF A(B AND A)C THEN PRINT "A ( B / A
 ) C"
90 IF A=B AND A)C THEN PRINT "A = B / A
 ) C"
100 IF A=B AND A(C THEN PRINT "A = B /
A ( C"
110 IF A)B AND A=C THEN PRINT "A ) B /
A = C"
120 IF A(B AND A=C THEN PRINT "A ( B /
A = C"
130 PRINT,,"TECLE F5 PARA NOVA RODADA"
```

COMENTARIOS

LINHA 30: define as variáveis A, B e C para receberem valores através de INPUT.

LINHAS 40 a 120: o computador compara o valor de A com os valores de B e C, sob todas as possibilidades existentes, e com auxflio dos operadores aritméticos, relacionais e lógicos, devendo "decidir" e informar qual a condição de A em relação a B e C. Note a rapidez com que

é feita a "análise" pelo computador.

11.4 PROGRAMA "O MSX TOMA VÁRIAS DECISÕES"

```
10 CLS: KEYOFF
20 A$(1)="CONSTITUCIONAL": A$(2)="PROMO
CIONAL": A$(3)="VERDADEIRAMENTE": A$(4)
="APAIXONADAMENTE": A$(5)="CAVALHEIRESC
O"
30 PRINT"POSSO FAZER-LHE UMA PERGUNTA?
(S/N)": INPUT R$
40 IF R$<>"S" THEN 130
50 CLS: PRINT,,"OK. RESPONDA-ME EM CINC
O SEGUNDOS:": C=INT(RND(-TIME)*5)+1
60 PRINT,,"QUANTAS LETRAS TEM A PALAVRA
:": PRINT: PRINT TAB(10) A$(C)
70 B=TIME: INPUT A
80 IF TIME>B+300 THEN 160
90 IF NOT A=LEN(A$(C)) THEN PRINT,,"ERR
OU!": GOTO 110
100 PRINT,"ACERTOU!"
110 PRINT
120 INPUT "NOVA TENTATIVA (S/N)"; R$
130 IF NOT R$="S" THEN 140 ELSE 50
140 PRINT,,"OK. PROGRAMA ENCERRADO."
150 END
160 PRINT,,"TEMPO ESGOTADO!": GOTO 110
170 RUN
```

COMENTÁRIOS

LINHA 20: é definida e indexada a variável A$, sem DIMensionamento, já que sua subscrição é inferior a 10. Cada subscrita recebe uma palavra diferente como atribuição.

LINHAS 30 e 40: fazem o computador dirigir uma consulta via vídeo ao usuário e, em função da resposta obtida via teclado, "decidir" se prossegue ou não com a execução do programa.

LINHA 50: no caso de continuar a execução do programa, define a va-

riável C e atribui a ela um valor aleatório entre 1 e 5.

LINHA 60: apresenta no vídeo a variável A$ indexada com o valor assumido pela variável C, a fim de que o usuário responda, via teclado, e dentro de cinco segundos, quantas letras tem a mesma.

LINHA 70: através do valor atribuído à variável B pela variável TIME do sistema operacional do MSX, começa a contagem de tempo. É definida a variável A, que deverá registrar através de INPUT a quantidade de letras da palavra apresentada.

LINHA 80: através dela o computador verifica se a resposta foi digitada dentro do prazo estipulado, caso em que passa para a linha seguinte. Em caso contrário, executa a linha 160.

LINHA 90: o computador analisa se a resposta dada pelo usuário está ou não certa, informando sua "conclusão" e continuando a execução do programa em conformidade com a mesma.

LINHA 120: nova consulta é feita ao usuário, cuja resposta via teclado será "analisada" pelo computador, que "decidirá" qual o rumo a tomar em função de sua "conclusão".

11.5 CONDIÇÕES MÚLTIPLAS

Nos quatro programas apresentados podemos notar que são criadas "condições múltiplas" para que o computador as analise, o que é feito através da instrução condicionante IF (= se). Quando uma condição proposta é "verdadeira", o computador executa a parte complementar da instrução, THEN (= então), que poderá ser um comando ou uma instrução.

Se a linha de instrução incluir em seqüência ELSE |e instrução|, e a condição da proposição for "falsa", será executado o que determinar a instrução ELSE. ELSE significa 'senão', 'de outra forma' ou 'de outro modo'. Se não houver ELSE, será executada a instrução seguinte.

11.6 OPERADORES DE RELAÇÃO

Pudemos notar também que as instruções que implicam "em tomada de decisão" pelo computador são elaboradas com auxílio dos "operadores de relação", que são:

```
=   igual
<>  diferente
<   menor
>   maior
=<  igual ou menor
=>  igual ou maior
```

11.7 OPERADORES LOGICOS

Outros operadores utilizados nas instruções que implicam "em tomada de decisão" pelo computador são os "operadores lógicos":

```
NOT = não
AND = e
OR  = ou (x e/ou y)
XOR = ou (x ou y)
IMP = implicação
EQV = equivalencia
```

Calculando e manipulando números 12

Usar um computador de padrão MSX para certos cálculos - como alguns que faremos aqui apenas a título de exemplo e prática - equivale a usar um iate de luxo numa piscina.

Todavia, para explorar ainda que apenas parcialmente a potencialidade de um MSX em cálculos, um livro todo não seria suficiente.

Assim, neste capítulo apenas repassaremos as instruções e/ou comandos, funções e regras essenciais que permitem "praticar" cálculos básicos. Aprendendo a efetuá-los e sabendo processá-los juntamente com os recursos de programação do BASIC MSX, conjugados com os operadores lógicos e de relação, será difícil encontrar limites para sua utilização em matéria de cálculos.

12.1 CÁLCULOS ARITMÉTICOS NO MODO DIRETO

Expressões simples:

São efetuados usando-se a instrução PRINT seguida dos operandos separados pelos sinais operadores e pressionando RETURN. Exemplos:

- ADIÇÃO: PRINT 100+33 / RETURN
- SUBTRAÇÃO: PRINT 100-33 / RETURN
- MULTIPLICAÇÃO: PRINT 12*12 / RETURN
- DIVISÃO: a) com quociente completo na forma decimal:
  PRINT 33/6 / RETURN (resultado = 5.5)
  b) com apenas a parte inteira do quociente:
  PRINT 33\6 / RETURN (resultado = 5)

Observação: se o dividendo contiver fração decimal, esta será eliminada antes da divisão, o mesmo acontecendo com o divisor. Exemplos:

```
PRINT 33.4\6 / RETURN  (resultado = 5)
PRINT 33\6.5 / RETURN  (resultado = 5)
```

- POTENCIAÇÃO:

```
PRINT 12^2 / RETURN
```

Particularidade: operador MOD: permite obter apenas o resto da divisão:

```
PRINT 33 MOD 6 / RETURN  (resultado = 3)
```

Observação: se qualquer dos operandos (dividendo ou divisor) contiver fração decimal, esta será desprezada antes de ser efetuada a operação:

```
PRINT 33.4 MOD 6 / RETURN  (resultado = 3)
PRINT 33 MOD 6.6 / RETURN  (resultado = 3)
```

Expressões complexas:

Os cálculos são efetuados de acordo com a seguinte ordem de prioridade:

1ª - EXPONENCIAÇÃO
2ª - RADICIAÇÃO
3ª - MULTIPLICAÇÃO
4ª - DIVISÃO
5ª - ADIÇÃO
6ª - SUBTRAÇÃO

A ordem de prioridade é alterada com o uso de parênteses. Compare os exemplos de a) com os de b):

a)
```
PRINT 12+6*7-5 / RETURN  (resultado = 49)
PRINT 15-5+10/2*6 / RETURN  (resultado = 40)
PRINT 7+3*10/2*5-4 / RETURN  (resultado = 78)
PRINT SQR(225)/2*5+5^2-15 / RETURN  (resultado = 47.5)
PRINT 18-2*5+25/5-2+3^3/3*2 / RETURN  (resultado = 29)
```

b)
```
PRINT 12+6*(7-5) / RETURN  (resultado = 24)
PRINT 15-(5+10)/2*6 / RETURN  (resultado = -30)
PRINT (7+3)*10/(2*5)-4 / RETURN  (resultado = 6)
PRINT SQR(225)/(2*5)+(5^2)-15 / RETURN  (resultado = 11.5)
PRINT 18-((2*(5+25/5)-2))+(3^3)/3*2 / RETURN
                                    (resultado = 18)
```

12.2 CÁLCULOS DE MATEMÁTICA COM FUNÇÕES DEFINIDAS NO MSX

São efetuados usando-se a instrução PRINT seguida da FUNÇÃO específica e do operando. Exemplos:

PRINT ABS(-66) / RETURN (resultado = 66, que é o valor absoluto do número relativo -66)

PRINT SQR(144) / RETURN (resultado = 12, que é a raiz quadrada ou de índice 2 do número 144)

Observação: para se extrair raiz de outro índice, o número deve ser elevado à potência do valor inverso desse índice. Raiz cúbica, ou de índice 3, do número 3375, por exemplo, é igual a 3375 elevado à potência do número 1/3 (inverso de 3), ou 3375^(1/3) no computador, resultando 14.9999...)

PRINT ATN(0.5) / RETURN (resultado = .463647..., que é o valor do arco-tangente da tangente de um ângulo, sendo os valores expressos em radianos.)

PRINT COS(0.5) / RETURN (resultado = .877582..., que é o valor do co-seno do ângulo expresso em radianos.)

PRINT EXP(3) / RETURN (resultado = 20.0855..., que é o número "e" (2.71828...) elevado à terceira potência.)

PRINT LOG(20.08553) / RETURN (resultado = 3, que é o logaritmo natural ou neperiano do número processado.)

Observação: para se obter o logaritmo decimal de um número dado, 1.000, por exemplo, deve-se dividir o logaritmo natural fornecido pelo logaritmo natural de 10. Exemplo:

PRINT LOG(1000)/LOG(10) / RETURN (resultado = 3, que é o logaritmo decimal (ou de Briggs) do número 1000.)

PRINT SGN(x) / RETURN (Fornece o valor unitário +1 ou -1 (sinal) gerador do número x relativo, ou o valor 0 de um número x neutro.)

PRINT SIN(0.70) / RETURN (resultado = .644217, que é o valor do seno do ângulo de 0.70 radianos.)

PRINT TAN(0.75) / RETURN (resultado = .9315964, que é o valor da tangente do ângulo de 0.75 radianos.)

Observação: a conversão de graus para radianos é feita pela fórmula: G/180*3,141592, em que G é o valor em GRAUS. A conversão de radianos para graus é feita pela fórmula: R/3,141592*180, em que R é o valor em RADIANOS. 3,141592653... é o valor de PI, cuja função não é definida no MSX.

12.3 FUNÇÕES DE CONVERSÕES DE NÚMEROS

CDBL - Converte números ou expressões em números equivalentes de dupla precisão. Exemplos:

```
10 A# = CDBL(25/12): B# = CDBL(12+8*6/(5/8))
20 PRINT A#: PRINT B#
```

CINT - Transforma uma expressão em número inteiro. Exemplo:

```
10 A% = CINT((12/15)/3+5^2)
20 PRINT A%
```

CSNG - Converte números ou expressões em números de simples precisão. Exemplo:

```
10 A! = CSNG(18+6/2*(12/8))
20 PRINT A!
```

FIX - Fornece apenas a parte inteira de um número fracionário, seja o número positivo ou negativo. Exemplos:

```
10 PRINT FIX(15.6)
20 PRINT FIX(-15.6)
```

INT - Fornece a parte inteira de um número fracionado, com arre-

dondamento para cima se o número for negativo. Exemplos:

```
10 PRINT INT(15.6)
20 PRINT INT(-15.6)
```

12.4 RND - UMA FUNÇÃO CURIOSA DE NÚMERO

RND - Fornece um número aleatório entre 0 e 1. Exemplos:

```
10 PRINT RND(1)
20 PRINT RND(-1)
```

Se o programa for rodado diversas vezes, os números gerados na primeira rodada serão repetidos. Para evitar essa situação, é usada a variável TIME do sistema operacional do computador. Exemplos:

```
10 PRINT RND(-TIME)
20 PRINT RND(+TIME)
```

Para obter números aleatórios maiores do que 1 e inteiros:

```
10 PRINT INT(RND(-TIME)*n)
```

sendo n o número de limite máximo.

12.5 AS FUNÇÕES BIN$, HEX$, OCT$

BIN$ - Transforma um número decimal em binário, sendo este fornecido na forma de "string". Exemplos:

```
10 PRINT BIN$(13)
20 PRINT BIN$(1986)
```

HEX$ - Transforma um número decimal em hexadecimal, sendo este fornecido na forma de "string". Exemplos:

```
10 PRINT HEX$(10)
20 PRINT HEX$(65535)
```

OCT$ - Transforma um número decimal em octal, sendo este fornecido na forma de "string". Exemplos:

```
10 PRINT OCT$(100)
20 PRINT OCT$(15000)
```

12.6 CONVERSÕES DE NÚMEROS BINÁRIOS, HEXADECIMAIS E OCTAIS

Podem ser efetuadas rapidamente pelo computador, através da instrução PRINT, nos modos direto ou indireto:

PRINT &B |número binário| fornece o número decimal equivalente ao binário fornecido no argumento. Exemplos:

```
PRINT &B11111 / RETURN
10 PRINT &B010011101
```

PRINT &H |número hexadecimal| fornece o número decimal equivalente ao hexadecimal fornecido no argumento. Exemplos:

```
PRINT &H2F2D
10 PRINT &H37A5
```

PRINT &O |número octal| fornece o número decimal equivalente ao octal fornecido no argumento. Exemplos:

```
PRINT &O1100
10 PRINT &O11234
```

Observação: se o número decimal convertido for indicado pelo computador com sinal negativo, deve ser somado a 65536 para ser obtido o número real.

12.7 OUTRAS FUNÇÕES MATEMÁTICAS

As funções matemáticas já citadas até aqui estão definidas em rotinas residentes na memória ROM do MSX, de forma que basta acioná-las para utilizá-las.

Todavia, há diversas outras funções matemáticas derivadas daquelas e que não foram incluídas como rotinas residentes. Estas podem ser definidas em BASIC MSX, conforme as expressões ou fórmulas que seguem:

```
SECANTE = 1/COS(X)
COSECANTE = 1/SIN(X)
COTANGENTE = 1/TAN(X)
ARCO SENO = ATN(X/SQR(-X*X+1))
ARCO CO-SENO = -ATN(X/SQR(-X*X+1))+1.5708
ARCO SECANTE = ATN(X/SQR(X*X-1))+SGN(SGN(X)-1)*1.5708
ARCO CO-SECANTE = ATN(X/SQR(X*X-1))+(SGN(X)-1)*1.5708
ARCO CO-TANGENTE = ATN(X)+1.5708
SENO HIPERBÓLICO = (EXP(X)-EXP(-X))/2
CO-SENO HIPERBÓLICO = (EXP(X)+EXP(-X))/2
TANGENTE HIPERBÓLICA = EXP(-X)/EXP(X)+EXP(-X)*2+1
SECANTE HIPERBÓLICA = 2/(EXP(X)+EXP(-X))
CO-SECANTE HIPERBÓLICA = 2/(EXP(X)-EXP(-X))
CO-TANGENTE HIPERBÓLICA = EXP(-X)/(EXP(X)-EXP(-X))*2+1
ARCO SENO HIPERBÓLICO = LOG(X+SQR(X*X+1))
ARCO CO-SENO HIPERBÓLICO = LOG(X+SQR(X*X-1))
ARCO TANGENTE HIPERBÓLICA = LOG((1+X)/(1-X))/2
ARCO SECANTE HIPERBÓLICA = LOG((SQR(-X*X+1)+1)/X)
ARCO CO-SECANTE HIPERBÓLICA = LOG((SGN(X)*SQR(X*X+1)/X)
ARCO CO-TANGENTE HIPERBÓLICA = LOG((X+1)/(X-1))/2
```

Observação: As fórmulas acima foram extraídas integralmente do MANUAL DO USUÁRIO do HOTBIT HB-8000.

INSTRUÇÃO DEF FN

Quando uma função sem rotina residente na memória do computador deve ser usada com freqüência num programa, ela poderá ser previamente definida pelo usuário através da instrução DEF FN e inserida no programa tantas vezes quantas necessárias.

Supondo-se, a título de exemplo, que tenhamos de lidar freqüentemente com logaritmos decimais (a função residente no MSX é de logaritmos naturais), definimos a função como segue:

```
10 DEF FN LD = LOG(A)/LOG(10)
```

em que LD representa o nome da função e A a chamada da função e o argumento para sua execução, como exemplificado a seguir:

```
20 INPUT A
30 PRINT FN LD
```

DEF FN LD e FN LD (nas linhas 10 e 30) podem ser digitadas sem os espaços de separação: DEFFNLD e FNLD.

A instrução DEF FN só é válida no modo indireto, isto é, quando utilizada numa linha de instrução de programa.

12.8 PROGRAMAS COMENTADOS

Os programas apresentados a seguir utilizam, a título de exemplo, alguns dos principais recursos examinados neste capítulo.

12.9 PROGRAMA "CALCULANDO NOTA MEDIA DE PROVAS"

```
10 CLS: INPUT "QUANTOS CANDIDATOS PREST
ARAM EXAMES";Q: PRINT
20 CLEAR 100*Q: Q=FRE("")/100
30 DIM N$(Q): DIM D(Q): DIM R(Q): DIM C
G(Q): DIM NM(Q)
40 FOR I=1 TO Q
50 CLS: LOCATE 0,5: PRINT "NOME DO CAND
IDATO: (MÁX. 20 ESPAÇOS)"
60 PRINT: INPUT N$(I)
70 PRINT,,"NOTA DE DATILOGRAFIA:": INPU
T D(I)
80 PRINT,,"NOTA DE REDAÇÃO:": INPUT R(I
)
90 PRINT,,"NOTA DE CONHECIMENTOS GERAIS
:": INPUT CG(I)
100 GOSUB 200
110 NEXT I
120 CLS: PRINT SPC(6)"NOTAS MÉDIAS DOS
CANDIDATOS"
130 PRINT SPC(6) STRING$(27,195)
```

```
140 PRINT,,"NOME";TAB(24) "NOTAS";TAB(3
3) "MÉDIA"
150 FOR I=1 TO 6
160 PRINT N$(I);TAB(22) D(I);R(I);CG(I)
;:PRINT TAB(33) USING "##.#";NM(I)
170 NEXT I:PRINT,,TAB(7)"Notas e média
ponderadas."
180 PRINT,,"Para rever lista: 'GOTO 120
' / RETURN"
190 END
200 NM(I)=((D(I)*3)+(R(I)*3)+(CG(I)*4))
/10
210 RETURN
```

COMENTÁRIOS

De forma geral, as linhas de instruções do programa apresentado, bem como dos programas seguintes, enquadram-se em comentários já feitos sobre instruções equivalentes de programas apresentados nos capítulos precedentes.

Assim, restringiremos os comentários apenas às partes do programa não comentadas anteriormente.

LINHAS 40 a 110: formam o laço constituído para captar as notas que serão enviadas à sub-rotina da linha 200, que calcula a nota média de cada examinando com ponderação. Através da indexação das variáveis dimensionadas para a finalidade, as notas ficam vinculadas aos nomes.

LINHAS 120 a 160: incumbem-se de apresentar no vídeo a lista de examinandos e respectivas notas.

LINHAS 200 e 210: sub-rotina para cálculo das notas médias das provas.

12.10 PROGRAMA "VALOR DE PRESTAÇÃO MENSAL"

```
10 CLS: COLOR 6,10: KEYOFF
20 PRINT TAB(6) "VALOR DE PRESTAÇÃO MEN
SAL"
30 PRINT TAB(6) STRING$(25,195)
```

```
40 PRINT,,TAB(10) "VALOR A FINANCIAR"
50 PRINT,,TAB(12) "CZ$";: INPUT V
60 PRINT,,TAB(10) "TAXA DE JUROS (%):"
70 PRINT,,TAB(15): INPUT I
80 PRINT,,TAB(10) "QUANTAS PARCELAS?"
90 PRINT,,TAB(16): INPUT N
100 P=V*((I/100)/(1-(1+I/100)^(-N)))
110 PRINT,,TAB(9) "VALOR DA PRESTAÇÃO:"
120 PRINT
130 PRINT SPC(12);"CZ$";:PRINTUSING"###
##,.##";P
140 PRINT,,TAB(8) STRING$(21,192)
150 PRINT TAB(8) STRING$(21,195)
160 PRINT,,TAB(9) "Novo cálculo? (s/n)"
: R$=INPUT$(1)
170 IF R$="S" OR R$="s" THEN 10 ELSE EN
D
```

COMENTÁRIOS

LINHAS 40 a 90: encarregam-se de obter, via teclado, e registrar em variáveis numéricas os dados necessários para os cálculos.

LINHA 100: define em BASIC MSX a fórmula de matemática financeira através da qual serão efetuados os cálculos para obtenção do valor a ser considerado como o da prestação mensal de um financiamento.

LINHAS 110 a 130: são responsáveis pela apresentação do resultado dos cálculos, na formatação especificada pela instrução PRINTUSING.

12.11 PROGRAMA "CALCULANDO TAXA DE FINANCIAMENTO"

```
10 CLS: SCREEN 0: COLOR 6,10: KEYOFF
20 S$=STRING$(21,192): T$=STRING$(21,19
5)
30 PRINT TAB(9)"TAXA DE FINANCIAMENTO"
40 PRINT,,TAB(9) S$
50 PRINT,,TAB(11) "VALOR FINANCIADO"
60 PRINT,,TAB(9) T$
```

```
70 PRINT,,TAB(12) "CZ$";: INPUT V
80 PRINT,,TAB(9) S$
90 PRINT,,TAB(10)"VALOR DA PRESTAÇÃO:"
100 PRINT,,TAB(9) T$
110 PRINT,,TAB(12) "CZ$";: INPUT P
120 PRINT,,TAB(9) S$
130 PRINT,,TAB(10) "QUANTAS PRESTAÇÕES?
"
140 PRINT,,TAB(9) T$
150 PRINT,,TAB(16): INPUT N
160 I=((P*N/V)^(1/(N+1)))-1
170 J=(V*I)/(1-(1+I)^(-N))
180 IF ABS(P-J)>1 THEN 250
190 PRINT,,TAB(9) S$
200 PRINT,,TAB(11) "TAXA DE JUROS (%):"
210 PRINT,,TAB(9) T$
220 PRINT
230 PRINT SPC(15): PRINTUSING "##.#####"
;I*100
240 GOTO 270
250 I=I+((P-J)/P)*I
260 GOTO 170
270 PRINT,,TAB(9) S$: PRINT TAB(9) T$
280 PRINT,,TAB(10) "Novo cálculo? (s/n)
": R$=INPUT$(1)
290 IF R$="S" OR R$="s" THEN 10 ELSE EN
D
```

COMENTÁRIOS

O programa acima segue a mesma linha do programa anterior, "VALOR DE PRESTAÇÃO MENSAL", apresentando diferença apenas quanto à fórmula para cálculo da taxa de juros compostos, que, por tratar-se de taxa incidente sobre financiamento com amortizações mensais, torna-se bastante complexa, sendo resolvida através do algoritmo das linhas 160, 170, 180, 250 e 260.

Por essa mesma razão, os resultados apresentados pelo programa não podem ser considerados absolutamente precisos sob o ponto de vista matemático-financeiro, não obstante oferecer uma aproximação bastante

ideal da realidade, atendendo perfeitamente os propósitos de sua apresentação.

As linhas 20, 40, 60, 80, 100, 120, 140, 190, 210 e 270 funcionam apenas como sublinhas ou molduras dos dados exibidos no vídeo.

12.12 PROGRAMA "OPERANDO NÚMEROS RELATIVOS"

```
10 CLS:KEYOFF: DEFINT A,B,C,D
20 A=RND(-TIME)*100
30 IF A=)50 THEN A=A*-1
40 B=-RND(-TIME)*100
50 PRINT,,TAB(10)"NÚMEROS RELATIVOS:":
PRINT
60 PRINT "A =";A
70 PRINT "B =";B
80 PRINT,,TAB(5)"Multiplicação de A por
 B:": PRINT TAB(5) STRING$(24,195): PRI
NT
90 C=A*B
100 PRINT "A * B = C; C = ";C
110 PRINT"Sinal de C = ";SGN(C)
120 PRINT "Valor absoluto de C =";ABS(C
)
130 PRINT,,TAB(5)"Adição de A com B:":
 PRINT TAB(5) STRING$(17,195): PRINT
140 D=A+B
150 PRINT"A + B = D; D = ";D
160 PRINT"Sinal de D =";SGN(D)
170 PRINT"Valor absoluto de D = ";ABS(D
)
180 PRINT,,TAB(5)"Tecle F5 para outros
exemplos."
```

COMENTÁRIOS

LINHAS 20 a 40: geram números aleatórios entre 0 e 100. Quando o número gerado pela linha 20 for igual ou maior de 50, a linha 30 se incumbe de transformá-lo em número negativo, multiplicando-o por -1. A

linha 40 gera apenas números negativos.

LINHAS 80 a 100 e 130 a 150: operam multiplicação e adição dos números relativos gerados.

LINHAS 110, 120, 160 e 170: fornecem o SGN e ABS dos resultados das operações realizadas com os números relativos gerados.

12.13 PROGRAMA "CÁLCULOS TRIGONOMETRICOS"

```
10 DEFSNG A-Z: CLS
20 DEFFNF=ATN(C/SQR(-C*C+1))
30 DEFFNG=-ATN(D/SQR(-D*D+1))+1.5708
40 PRINT SPC(10): INPUT "Ângulo em grau
s:";A
50 B=A*.017453293#
60 PRINT,,SPC(5);A;"graus =";B;"radiano
s."
70 C=SIN(B)
80 D=COS(B)
90 E=TAN(B)
100 PRINT,,"Seno =";C
110 PRINT,,"Co-seno =";D
120 PRINT,,"Tangente =";E
130 PRINT,,"Arco-seno =";FNF
140 PRINT,,"Arco-co-seno =";FNG
150 PRINT,,"Arco-tangente = ";ATN(E)
160 PRINT,,SPC(5)"Tecle F5 para novos c
álculos."
```

COMENTÁRIOS

LINHA 10: são definidas como de simples precisão as variáveis numéricas compreendidas entre A e Z, que serão usadas no programa, a fim de que sejam apresentadas com apenas seis dígitos. Se não houver tal definição, os resultados serão apresentados com 14 dígitos, em vista da natureza dos cálculos, o que é desnecessário, considerando a finalidade do presente programa.

LINHAS 20 e 30: através da instrução DEF FN são definidas funções para

cálculos do arco-seno e arco-co-seno, não incluídas nas rotinas residentes no MSX.

LINHA 50: muda para RADIANOS o ângulo entrado em GRAUS. Os cálculos trigonométricos são efetuados pelo MSX em RADIANOS.

LINHAS 70, 80 e 90: através das funções SIN, COS e TAN, os valores de SENO, CO-SENO e TANGENTE do ângulo considerado são atribuídos às variáveis C, D e E, respectivamente, e fornecidos através das linhas 100 a 120.

LINHAS 130 a 150: através das funções definidas nas linhas 20 e 30 e da função ATN residente, são calculados e fornecidos os valores do arco-seno, arco-co-seno e arco-tangente do ângulo considerado. O valor de arco-tangente é expresso com mais de seis dígitos por ser definido através da função própria e não através de uma variável.

Instruções especiais de desvio

13

* ON ERROR GOTO / RESUME / ERROR / ERL / ERR
ON |x| GOTO
ON |x| GOSUB
* ON INTERVAL GOSUB
* ON KEY GOSUB
* ON STOP GOSUB
* ON SPRITE GOSUB
* ON STRIG GOSUB

Tais instruções indicam ao computador para onde |número de linha| deve ser desviado ou dirigido o fluxo de execução do programa nas situações previstas ou criadas pelas mesmas. As assinaladas com asterisco (*) são consideradas instruções de tratamento de interrupções.

Os programas apresentados neste capítulo ilustram a ação das primeiras seis. As duas últimas serão vistas no Capítulo 18 - "SPRITES".

INSTRUÇÃO ON ERROR GOTO

Para o computador, esta instrução significa que, ocorrendo algum erro durante a realização de um programa, a sua execução deve ser desviada para o ponto especificado na mesma, geralmente uma rotina de tratamento de erro.

Os erros que podem ocorrer são os do modo direto (erros de sintaxe, por exemplo), ou os definidos pelo programador.

INSTRUÇÃO RESUME

O retorno (saída da rotina) à execução normal do programa principal é feito através da instrução RESUME, que poderá indicar um número de linha do programa, por exemplo, RESUME 20.

Se a instrução RESUME especificar 0, a execução retornará para a linha onde é iniciada a rotina de tratamento do erro.

Se for complementada por NEXT, retornará para a linha seguinte à da ocorrência do erro.

INSTRUÇÃO ERROR

Permite simular a ocorrência de um erro ou definir códigos de erros pelo usuário, além dos já definidos no sistema operacional do computador e constantes da tabela específica. Exemplo:

```
20 INPUT "DIGITE UM NÚMERO ENTRE 1 e 5";A
30 IF A<1 OR A>5 THEN ERROR 70
```

Uma instrução desse tipo só poderá constar num programa do qual faz parte uma instrução precedente do tipo:

```
10 ON ERROR GOTO 200
```

de forma que, se o valor atribuído a A for menor do que 1 ou maior do que 5, o programa será desviado para a rotina de tratamento de erro, a qual poderá ser:

```
200 IF ERR = 70 THEN PRINT "O NÚMERO DEVERÁ SER ENTRE 1 E 5!": RESUME NEXT
```

Os erros definidos no sistema operacional do **MSX** utilizam os códigos de números 1 a 59. O usuário poderá usar quaisquer números entre 60 e 255 para definir códigos de seu interesse. Poderá, todavia, usar a instrução ERROR sem número de código definido. Nesse caso, as instruções citadas teriam a configuração seguinte (linhas 10, 20, 30 e 200):

```
10 ON ERROR GOTO 200
20 INPUT "DIGITE UM NUMERO ENTRE 1 E 5"
;A
30 IF A<1 OR A >5 THEN ERROR
40 PRINT A
50 GOTO 20
200  PRINT,, "O NÚMERO DEVERÁ SER ENTRE
 1 E 5!":RESUME NEXT
```

As linhas 40 e 50 completam o programa para uma experiência definitiva, permitindo que diversos números sejam experimentados.

VARIÁVEIS ESPECIAIS ERL E ERR

Essas variáveis do sistema operacional do computador são utilizadas para indicar a linha do programa em que ocorre um erro e o número do código de erro, este segundo a tabela publicada no manual de BASIC ou do usuário.

Os dois programas seguintes exemplificam o uso das instruções comentadas:

13.1 PROGRAMA "ON ERROR GOTO"

```
10 ON ERROR GOTO 100
20 CLS: LOCATE 0,7
30 READ A$: PRINT A$
50 GOTO 30
60 REM TABELA DATA
70 DATA JAN ,FEV ,MAR ,ABR ,MAI ,JUN ,
80 END
100 PRINT: PRINT,,"A tabela DATA não te
m mais dados p/ler."
110 PRINT,,TAB(6)"Linha do erro:";ERL
120 PRINT TAB(6)"Código do erro:";ERR
130 PRINT,,TAB(13)"Favor corrigir."
140 RESTORE
150 FOR I=1 TO 1500: NEXT I
160 RESUME 20
```

Observações: depois de rodar o programa uma ou duas vezes, altere a linha 160 para:

```
160 RESUME NEXT
```

e rode novamente o programa observando a diferença de mudança no fluxo de execução do mesmo.

Para corrigir o programa, evitando a ocorrência de erro apontada pelo computador, acrescente ao mesmo a seguinte linha:

```
40 IF A$="" THEN 80
```

13.2 PROGRAMA "RESUME"

```
10 ON ERROR GOTO 100
20 CLS: LOCATE 5,10
30 PRINT "Digite um número entre 1 e 5:
"
40 PRINT: R$=INPUT$(1): N=VAL(R$)
50 IF N<1 OR N>5 THEN ERROR
60 PRINT TAB(18) N
70 GOSUB 150
80 GOTO 20
100 PRINT,,TAB(7)"Número fora de faixa.
";
110 GOSUB 150
120 RESUME NEXT
150 FOR I=1 TO 500: NEXT I: RETURN
```

Observações: com o programa em execução, experimente digitar números maiores do que 5.

Experimente depois mudar a linha 120 para:

```
120 RESUME 20
```

e note que um número fora de faixa não será mais impresso no vídeo, em razão de o retorno ocasionado por NEXT não ser mais para a linha seguinte à da detecção do erro.

INSTRUÇÕES ON |VARIAVEL| GOTO / ON |VARIAVEL| GOSUB

Desviam a execução do programa para uma linha de instrução ou de uma sub-rotina. Se a instrução controlar desvio opcional para três linhas de instruções ou de sub-rotinas, a variável que complementa ON deverá assumir valor entre 1 e 3 e GOTO (ou GOSUB) DEVE SER COMPLEMENTADA com os três números das linhas de intruções ou das sub-rotinas. Exemplificando:

```
10 INPUT A
20 ON A GOTO 70, 120, 90
```

Conforme o valor de A - 1, 2 ou 3 -, a execução do programa será desviada para a linha de programa indicada em primeiro lugar (70), ou em segundo (120), ou em terceiro (90).

Os dois programas apresentados a seguir ilustram a explicação:

13.3 PROGRAMA "ON X GOTO"

```
10 CLS: LOCATE 0,5
20 PRINT"ESCOLHA O SISTEMA PARA CONVERT
ER UM Nº:"
30 PRINT,,TAB(5)"1 - DECIMAL PARA HEXAD
ECIMAL"
40 PRINT,,TAB(5)"2 - DECIMAL PARA BINÁR
IO"
50 PRINT,,TAB(5)"3 - DECIMAL PARA OCTAL
"
60 A$=INPUT$(1): X=VAL(A$)
70 LOCATE 2,15: INPUT "DIGITE O Nº A SE
R CONVERTIDO:";B
80 PRINT: PRINT; SPC(3)
90 ON X GOTO 100,120,140
100 PRINT B "EM HEXADECIMAL = ";HEX$(B)
110 GOTO 150
120 PRINT B "EM BINÁRIO = ";BIN$(B)
130 GOTO 150
140 PRINT B "EM OCTAL = ";OCT$(B)
150 PRINT,,TAB(5)"P/NOVA CONVERSÃO: F5"
```

13.4 PROGRAMA "ON VAL (A$) GOSUB"

```
10 CLS: LOCATE 5,7: PRINT "PRESSIONE UM
A DAS TECLAS 1,2,3"
20 A$=INPUT$(1)
30 LOCATE 8,11
40 ON VAL(A$) GOSUB 70,90,110
50 IF VAL(A$)<1 OR VAL(A$)>3 THEN GOSUB
 140
60 GOTO 10
70 PRINT "TECLA 1 = UM É POUCO"
80 GOSUB 130: RETURN
90 PRINT "TECLA 2 = DOIS É BOM"
100 GOSUB 130: RETURN
110 PRINT "TECLA 3 = TRÊS É DEMAIS"
120 GOSUB 130: RETURN
130 FOR I=1 TO 600: NEXT I: RETURN
140 PRINT "PRESSIONOU TECLA ERRADA!"
150 GOSUB 130: RETURN
```

Observação: com o programa em execução, experimente digitar um número maior do que 3 e note que a execução do programa passa para as linhas 50 e 140, pois o computador "compara" e constata que só estão indicados três desvios, não podendo em conseqüência executar a instrução se o valor de A$, sendo superior a 3, indicar mais rotinas do que as existentes ou especificadas na instrução.

INSTRUÇÃO ON INTERVAL

Para o computador, essa instrução significa: em intervalo de tempo |x| execute a sub-rotina |número de linha|.

O tempo |x| é medido pelo "clock" interno do MSX, que dá 60 pulsos a cada segundo.

Assim, uma instrução ON INTERVAL = 300 GOSUB |número de linha|, geralmente colocada no fim do programa, faz com que o computador execute a sub-rotina da linha de número especificado a cada cinco segundos. A instrução deve ser ativada ou habilitada por uma instrução INTERVAL ON. Vamos experimentá-la no programa que segue:

13.5 PROGRAMA "ON INTERVAL = X"

```
10 CLS: LOCATE 7,10: PRINT "O TELEFONE
ESTÁ CHAMANDO"
20 ON INTERVAL=300 GOSUB 100
30 INTERVAL ON
40 BEEP: BEEP
50 GOTO 40
100 FOR I=1 TO 1500: NEXT I
110 RETURN
```

Quando o programa estiver em execução, para "calar" a campainha do MSX, pressione simultaneamente as teclas CTRL e STOP. Não é preciso atender o telefone...

INSTRUÇÃO ON KEY GOSUB

Para o computador, essa instrução significa: se uma tecla de função programável (habilitada ou especificada na instrução seguinte) for pressionada, execute a sub-rotina |número de linha de seu início|.

A instrução ON KEY GOSUB deve ser habilitada ou ativada por uma instrução KEY (número da tecla de função desejada) ON, como veremos no programa apresentado a seguir:

13.6 PROGRAMA "ON KEY GOSUB"

```
10 CLS: ON KEY GOSUB 100,150,200
20 LOCATE 0,7: PRINT "PARA OUVIR MÚSICA
 PRESSIONE A TECLA F1"
30 PRINT,,"PARA OUVIR A CAMPAINHA TECLE
 F2"
40 PRINT,,"PARA SUSTAR O PROGRAMA TECLE
 F3"
50 KEY(1)ON:KEY(2)ON: KEY(3)ON
60 GOTO 60
70 PLAY "O4L4GFA-O5C8F8D"
100 PLAY "O4L4GFA-O5C8F8D"
110 PLAY "C8F8D04B-O5C04GF"
```

```
120 RETURN
150 FOR A=1 TO 25
160 BEEP: BEEP
170 NEXT A: RETURN
200 PRINT,,"FIM DE PROGRAMA":END
```

INSTRUÇÃO ON STOP GOSUB

Para o computador, essa instrução significa: se durante a execução do programa forem pressionadas as teclas CTRL e STOP simultaneamente, execute a sub-rotina |número de linha de seu início|.

Ao entrar na sub-rotina indicada pela instrução, a execução só poderá ser sustada por esse meio, isto é, através de CTRL + STOP, se fizer parte da sub-rotina do desvio a instrução STOP OFF, que desativa a instrução ON STOP GOSUB. Caso contrário, a sub-rotina só poderá ser usada para outra finalidade, inclusive a de indicar a maneira de "parar" o programa sem ser por força de erro. A instrução RETURN deverá finalizar a sub-rotina.

A ativação ou habilitação da instrução ON STOP GOSUB é necessariamente feita por STOP ON.

13.7 PROGRAMA "ON STOP GOSUB"

```
10 ON STOP GOSUB 100
20 STOP ON
30 CLS:LOCATE 0,10
40 PRINT "TENTE PARAR ESTE PROGRAMA PRE
SSIONANDO    SIMULTANEAMENTE 'CTRL' E '
STOP'."
50 PRINT: INPUT A$
60 IF A$="FIM" THEN 70 ELSE 30
70 PRINT,,"Digitou FIM porque se apavor
ou, hein?": END
100 PRINT,,"Pressionou CTRL+STOP, mas n
ão adiantou."
110 PRINT,,"Se quiser parar o programa
tem que di-"
```

```
120 PRINT"gitar FIM e pressionar a tecl
a RETURN."
130 FOR A=1 TO 2000: NEXT A
140 RETURN
```

13.8 DESATIVAÇÃO DAS INSTRUÇÕES ESPECIAIS DE DESVIO

As instruções:

```
ON INTERVAL GOSUB
ON KEY GOSUB
ON STOP GOSUB
```

podem ser desativadas e novamente ativadas em estágios diversos de um programa. Já vimos que as mesmas são ativadas pelas instruções:

```
INTERVAL ON
KEY ON
STOP ON
```

Sua desativação é feita pelas instruções:

```
INTERVAL OFF  e  INTERVAL STOP
KEY OFF       e  KEY STOP
STOP OFF      e  STOP STOP
```

Quando seguidas por OFF, a desativação é absoluta.

Quando seguidas por STOP, a desativação é relativa. Uma instrução principal existente no programa é memorizada e executada quando aparecer no programa uma instrução correspondente de ativação.

Gravação e carregamento de programas em fita cassete 14

CSAVE, CLOAD, CLOAD? / SAVE, LOAD, MERGE / BSAVE, BLOAD /
MOTOR ON, MOTOR OFF

A gravação em fita cassete de um programa existente na memória do computador e a operação inversa - carregar para a memória do computador um programa previamente gravado em fita cassete - constituem tarefa fácil quando se relacionam com um computador do padrão MSX, principalmente se o gravador utilizado possuir;

- plugue de entrada para controle remoto do motor;
- conta-giros.

O gravador deve ser monaural (se for estéreo, deverá poder funcionar como monaural) e é conectado ao microcomputador através de um cabo de três linhas.

A conexão do cabo ao microcomputador é feita através de conector único, enquanto a extremidade correspondente ao gravador é provida de três plugues que devem ser encaixados nas tomadas EAR, MIC e REMOTE conforme instruções do manual do usuário.

O acionamento e a parada do motor do gravador são automaticamente controlados pelo microcomputador.

14.1 GRAVAÇÃO E CARREGAMENTO DE PROGRAMAS EM BASIC NO FORMATO BINÁRIO

COMANDO CSAVE - Grava ou "salva" em fita cassete, em formato binário, um programa em BASIC existente na memória do MSX.

Procedimento:

- Colocar a fita no gravador e posicioná-la no ponto em que deve ser iniciada a gravação.
- Travar simultaneamente as teclas REC e PLAY (ou SAVE e LOAD se for um "datacorder") do gravador.
  Não há necessidade de ajustar os controles de volume e de tom do gravador. Os gravadores modernos dispõem de controle automático de nível de gravação.
- Digitar o comando CSAVE seguido do nome, colocado entre aspas, do programa a ser gravado. Exemplo: CSAVE "FIPES".
- Pressionar uma vez a tecla RETURN.

Ao terminar a gravação surgirá a notação OK no vídeo e o motor do gravador será desligado automaticamente pelo microcomputador.

COMANDO CLOAD - Carrega de uma fita cassete para a memória do computador um programa em BASIC previamente gravado em formato binário.

Procedimento:

- Colocar a fita no gravador e posicioná-la no ponto de início da gravação. Caso não seja no começo da fita, este ponto deverá ter sido anotado com referência ao conta-giros do gravador ou à escala existente na janela de visualização do ponto de bobinamento da fita cassete.
- Travar ou pressionar a tecla PLAY (ou LOAD) do gravador.
- Ajustar o controle de volume pouco acima do ponto médio, devendo o controle de tom ser ajustado no ponto máximo de reprodução de agudos.
- Digitar o comando CLOAD se o programa a ser carregado for o primeiro na posição em que se achar a fita. Se houver outros programas gravados e o que deve ser carregado tiver de ser "procurado" ou selecionado pelo computador desde o começo da fita, o comando CLOAD deverá ser digitado sempre seguido do nome do programa entre aspas: CLOAD "FIPES".
- Pressionar uma vez a tecla RETURN.

Se o comando CLOAD foi feito sem um nome de programa especificado e o processo de carregamento estiver desenvolvendo-se normalmente, surgirá no vídeo a notação ACHEI seguida do nome do programa.

Se o comando CLOAD foi feito com o nome do programa a ser carregado, outros que forem reproduzidos antes pelo gravador farão surgir no vídeo a notação PULEI seguida do nome do programa.

Ao completar-se o carregamento, surgirá a notação OK no vídeo e o motor do gravador será automaticamente desligado pelo computador.

Caso contrário, se o carregamento não se completar depois de algum tempo e o gravador continuar funcionando sem que haja mais programas na fita, o processo deve ser interrompido através das teclas CTRL e STOP pressionadas simultaneamente. Surgirá no vídeo a notação:

ERRO PERIFÉRICO (HOTBIT) / DEVICE I/O ERROR (EXPERT)

indicando a ocorrência de irregularidade.

Nesse caso, a fita deverá ser rebobinada ao ponto de referência e o processo reiniciado. Antes, porém, deve ser verificado se o controle de volume está no ponto certo. Em caso positivo, nova posição deve ser experimentada. Geralmente, falhas no carregamento, quando ocorrem, são devidas a desacerto no volume de reprodução da fita. Eventualmente, a causa poderá ser diferença no alinhamento do cabeçote do gravador, se a gravação do programa foi feita em outro aparelho.

14.2 CONFERÊNCIA DE GRAVAÇÃO DE PROGRAMA EM BASIC NO FORMATO BINÁRIO

COMANDO CLOAD? - Compara a gravação de um programa em BASIC, no formato binário, com o programa existente na memória do computador, e do qual foi feita a gravação.

Isto só deve ser feito se estiver na memória do microcomputador o programa do qual foi feita a gravação.

O comando CLOAD? pode ou não ser seguido do nome do programa entre aspas. O procedimento é o mesmo de CLOAD.

Se o programa foi gravado corretamente, ao término da verificação surgirá a notação OK no vídeo.

Se houver diferença entre a gravação feita e o programa existente na memória do computador, surgirá a notação:

ERRO / VERIF. (HOTBIT) / VERIFY ERROR (EXPERT)

devendo, então, ser feita nova gravação.

14.3 GRAVAÇÃO E CARREGAMENTO DE PROGRAMAS EM BASIC NO FORMATO ASC II

COMANDO SAVE - Grava ou "salva" em fita cassete, em formato ASC II, um programa em BASIC existente na memória do computador.

O procedimento com gravador e fita é o mesmo de CSAVE, diferindo apenas no comando, que deve ter formato como exemplificado a seguir:

SAVE "CAS:NOME"

NOME devendo ser o nome do programa.

COMANDO LOAD - Carrega de fita cassete para a memória do computador um programa em BASIC previamente gravado em formato ASC II.

O procedimento com gravador e fita é o mesmo de CLOAD, diferindo apenas no comando, que deve ter a formatação indicada a seguir:

LOAD "CAS:", R ou LOAD "CAS:NOME", R

sendo R opcional, para execução automática após o carregamento.

COMANDO MERGE - Carrega de fita cassete para a memória do computador um programa em BASIC previamente gravado no formato ASC II, juntando-o a outro existente na memória do computador.

Os programas a serem unidos por esse processo devem ter numeração diferente de linhas, pois, se tiverem números idênticos às linhas já existentes na memória do computador, estas serão substituídas pelas linhas que entrarem através de MERGE.

O procedimento com gravador e fita é o mesmo de CLOAD, diferindo apenas no comando, que deve ter a seguinte formatação:

MERGE "CAS:" ou MERGE "CAS:NOME"

NOME devendo ser o nome do programa conforme foi gravado.

14.4 GRAVAÇÃO E CARREGAMENTO DE PROGRAMAS EM LINGUAGEM DE MÁQUINA

COMANDO BSAVE - Grava ou "salva" em fita cassete um programa em linguagem de máquina, localizado em área determinada na memória do computador.

O procedimento com gravador e fita é o mesmo de CSAVE, diferindo apenas no comando, devendo constar dele, além do nome do programa, os endereços inicial e final da área a ser gravada. Se o endereço para execução do programa for diferente do de seu início, deverá constar do comando de gravação. Sua formatação será:

BSAVE "CAS:NOME",60000,61250,60050

sendo os números, respectivamente, endereço inicial, endereço final e endereço para execução do programa. Naturalmente, os números variarão de acordo com cada programa.

COMANDO BLOAD - Carrega de fita cassete para a memória do computador um programa previamente gravado em linguagem de máquina.

O procedimento com gravador e fita é o mesmo de CLOAD, diferindo apenas no comando, que deverá ter a seguinte formatação:

BLOAD "CAS:", R ou BLOAD "CAS:NOME", 60050 ou R

sendo opcional a colocação do número indicativo do endereço para execução do programa, podendo ser colocada a letra R em seu lugar.

Quando for indicado no comando BLOAD o endereço para início de execução do programa - que deverá coincidir com o especificado no comando BSAVE correspondente - ou for colocada a letra R após o comando, conforme exemplo acima, o programa entra automaticamente em execução após o carregamento da fita.

O comando BLOAD poderá conter também indicação de deslocamento de área, na forma de uma vírgula após o endereço de execução e um número inteiro negativo ou positivo, caso em que todos os endereços de gravação serão deslocados pelo valor indicado. Se não for indicado um deslocamento, depois de carregado o programa será armazenado entre os endereços inicial e final indicados em BSAVE.

COMANDO MOTOR ON - Desativa o controle exercido remotamente pe-

lo computador sobre o motor do gravador.

Normalmente, se for acionado o comando MOTOR ON e a tecla PLAY do gravador estiver pressionada, este funcionará, ao passo que, com o controle remoto do motor ativado, o gravador só funcionará em PLAY ou REC/PLAY se for feito no computador um dos comandos que implicam movimentação da fita.

COMANDO MOTOR OFF - Ativa ou habilita o controle remoto exercido pelo computador sobre o motor do gravador.

Observação: se o controle remoto estiver em ON e for feito o comando direto:

MOTOR / RETURN

o controle remoto passará para OFF.

Se estiver em OFF e for comandado novamente:

MOTOR / RETURN

o controle remoto passará para ON.

Os controles ou comandos MOTOR ON e MOTOR OFF são utilizados em programas de arquivos seqüenciais, como será visto no Capítulo 15 - PROCESSAMENTO DE ARQUIVOS.

Processamento de arquivos 15

Um dos relevantes serviços prestados por microcomputadores é o processamento de arquivos, sem dúvida alguma.

Enquadram-se nessa modalidade os "bancos de dados", que compreendem "fichários", "agendas", "cadastros" etc., e "controles de estoque", "contas a pagar ou a receber", "inventários do ativo fixo" etc.

De forma geral, no âmbito empresarial os programas de arquivos processados por microcomputadores o são através de discos ou disquetes e, necessariamente nestes casos, por programas ou sistemas operacionais de considerável porte.

Quando são usados disquetes, geralmente duas unidades de "disk-drivers" ou acionadores de discos são usadas.

A implantação de um sistema de processamento de dados a partir de tal dimensão, todavia, deve ser justificada por considerável volume de trabalho, devendo ser precedida de análise e avaliação criteriosas de suas finalidades e necessidades, bem como do "software" disponível como suporte, manutenção etc.

Bom número de pequenas empresas e profissionais liberais adotaram, talvez um pouco precipitadamente, sistemas desse porte e concluíram, pouco tempo depois, que a elevada ociosidade de seu equipamento comprometia seriamente o investimento feito.

Em razão desse fato, e de vários outros problemas a considerar, aconselha-se que, antes da adoção de tal solução, seja feito um aprofundamento na matéria através dos dispositivos periféricos comuns e que operam perfeitamente com um computador do padrão MSX: um aparelho de TV como monitor e um gravador cassete, de preferência um que tenha si-

do desenvolvido especificamente para essa finalidade.

Um impressora poderá eventualmente ser indispensável, se houver necessidade de emissão de considerável volume de listas ou relatórios.

Não será difícil constatar que, apenas com esses recursos, excelentes resultados poderão ser obtidos no processamento de diversas modalidades de arquivos, tornando perfeitamente dispensável, em muitos casos, a adoção de outro tipo de equipamento.

Se é que podem ser consideradas assim, as principais desvantagens do processamento de arquivos por gravador cassete são a menor velocidade, comparada com o processamento por acionador de disco, e a impossibilidade de leitura de acesso direto ou aleatório, sendo praticável apenas o processamento chamado "seqüencial".

Tais desvantagens, todavia, poderão ser minimizadas, como veremos a seguir, através de programas gerenciadores eficientes e de boa qualidade.

15.1 COMANDOS E/OU INSTRUÇÕES DO MSX PARA PROCESSAMENTO DE ARQUIVOS

Os comandos e/ou instruções usados no processamento de arquivos no MSX são os seguintes:

OPEN	Abre um arquivo.
PRINT #	Insere dados no arquivo.
PRINT # USING	Insere dados formatados no arquivo.
INPUT #	Obtém ou recupera dados do arquivo.
LINEINPUT #	Obtém ou recupera dados do arquivo, sem as restrições de INPUT #.
MAXFILES	Determina o número e quantidade de arquivos.
EOF	Testa fim de arquivo durante a leitura da fita.
CLOSE #	Fecha um arquivo.

Deve ser lembrado que "vídeo no modo texto" e "vídeo no modo gráfico" são também enquadrados como dispositivos de "arquivo" no sistema MSX, embora neles os dados possam apenas ser inseridos e não recuperados ou lidos pelo computador. Enquadramento idêntico é dado a uma impressora.

Os únicos dispositivos que podem armazenar dados fornecidos pe-

lo computador e fornecê-los de volta são o gravador - através de fitas cassete - e o acionador de disco ou "disk-driver" - através de discos ou disquetes -, ambos meios magnéticos. Neste livro abordaremos apenas processamento de arquivos através de gravador cassete.

Ao serem abertos arquivos, devem ser discriminados os dispositivos usados, de acordo com as seguintes siglas, sempre seguidas de dois pontos (:):

A: ou B: - para acionador de disco 1 ou 2.
CAS: - para gravador cassete.
CRT: - para vídeo no modo texto.
GRP: - para vídeo no modo gráfico.
LPT: - para impressora.

Dos dispositivos acima, como já dito, os únicos que recebem e fornecem dados de arquivo são o acionador de disco e o gravador, podendo, portanto, um arquivo ser aberto para esses dispositivos tanto para entrada como para saída de dados, de forma que a instrução OPEN deve incluir sempre o nome do arquivo. Por exemplo, um arquivo de dados pessoais pode receber o nome de DATAPES, devendo as instruções para sua abertura e processamento ter a seguinte configuração:

```
|nº de linha| OPEN "CAS:DATAPES" FOR OUTPUT AS #1
```

para gravar dados do computador numa fita, e

```
|nº de linha| OPEN "CAS:DATAPES" FOR INPUT AS #1
```

para recuperar dados da fita para o computador.

O número 1 após AS # indica o número de arquivo. Se um único arquivo estiver sendo usado, não há necessidade de usar a instrução MAXFILES antecedendo a OPEN. Caso contrário, sim.

Para os demais dispositivos, dado o fato já mencionado de que funcionam apenas como repositórios de dados de um arquivo, não fornecendo os mesmos de volta ao computador - não havendo, portanto, leitura dos mesmos - as instruções para abertura de arquivo são as seguintes:

```
|nº de linha| OPEN "CRT:" FOR OUTPUT AS #1
```

```
nº de linha  OPEN "GRP:" FOR OUTPUT AS #1
nº de linha  OPEN "LPT:" FOR OUTPUT AS #1
```

15.2 COMEÇANDO A ARQUIVAR

Já vimos como "abrir" um arquivo e quais os comandos usados no seu processamento. Vamos agora começar a pôr em prática o que aprendemos.

Para serem "arquivados", os dados devem ser armazenados em variáveis ou atribuídos às mesmas. As variáveis funcionam como pastas de cartolina num arquivo convencional.

Digamos que desejamos arquivar os seguintes dados, separadamente: 10, 33, 17.34, JESUS, MARIA, JOSE:

Procedimentos:

- A primeira providência é avaliar aproximadamente o espaço que será necessário na área de variáveis alfanuméricas e fazer a reserva correspondente. Estimamos, com folga, 500 "bytes" para o pequeno arquivo que vamos organizar. Fazemos então a primeira linha do programa gerenciador:

```
50 CLEAR 500
```

- Colocamos os dados a arquivar em variáveis, de acordo com a sua natureza, e as dispomos numa ou mais linhas de programas:

```
100 A = 10: B = 33: C = 17.34
110 A$ = "JESUS": B$ = "MARIA": C$ = "JOSE"
```

- Abrimos o arquivo, na modalidade de saída de dados do computador para o gravador:

```
120 OPEN "CAS:DADOS" FOR OUTPUT AS #1
```

- Inserimos as instruções que permitem gravar os dados na fita, devendo ser seguidas dos dados na ordem em que devem ser gravados e, conseqüentemente, lidos ou recuperados depois:

```
130 PRINT #1, A, B, C
140 PRINT #1, A$;","; B$;","; C$
```

(Note que as variáveis numéricas podem ser colocadas seguidas apenas de vírgulas separando-as, enquanto as variáveis alfanuméricas devem ser separadas por ponto-e-vírgula (;) e vírgula entre aspas (",") e novamente ponto-e-vírgula (;) se, em cada caso, uma série ou diversas variáveis forem usadas após uma instrução PRINT.)

- Fechamos o arquivo:

```
150 CLOSE #1
160 END
```

- Colocamos uma fita no gravador.
- Travamos as teclas REC e PLAY do gravador.
- Pressionamos uma vez a tecla RUN do MSX, tecla F5.

Em instantes, os dados estarão "arquivados" na fita, isto é, estarão gravados na mesma. Aparecerá o OK costumeiro no vídeo e, se o controle remoto do motor do gravador estiver conectado ao microcomputador, o gravador parará automaticamente. Caso contrário, após aparecer o OK no vídeo, deverá ser acionada a tecla STOP do gravador

15.3 LENDO OU RECUPERANDO DADOS DO ARQUIVO

Procedimentos:

- Rebobinamos a fita cassete até o ponto pouco anterior ao de início de gravação de dados.
- Digitamos o seguinte programa para leitura ou recuperação dos dados da fita para o computador:

A primeira linha para reserva de espaço na área própria:

```
200 CLEAR 500
```

Abertura do arquivo para entrada de dados da fita para o computador:

```
210 OPEN "CAS:DADOS" FOR INPUT AS #1
```

Linhas de instruções que permitem que os dados lidos na fita sejam armazenados em variáveis:

```
220 INPUT #1, A,B,C
230 INPUT #1, A$,B$,C$
```

Linha de instrução que verifica se não há mais dados para ler no arquivo, fechando-o se não houver:

```
240 IF EOF(1)= -1 THEN CLOSE #1
```

- Digitamos também as seguintes instruções para que os dados sejam colocados no vídeo logo após a leitura da fita:

```
250 PRINT A;B;C
260 PRINT A$;B$,C$
270 END
```

- Travamos a tecla PLAY (ou LOAD) do gravador, se o controle remoto de acionamento de seu motor estiver conectado ao microcomputador.
- Digitamos GOTO 200 e pressionamos a tecla RETURN do MSX.

Após alguns instantes, estarão os dados do arquivo exibidos no vídeo, se tudo tiver sido feito corretamente.

Observação: caso o gravador não tenha controle remoto de acionamento controlado pelo microcomputador, os dois últimos itens dos procedimentos devem ser invertidos na ordem de execução.

15.4 FORMATAÇÃO DAS INSTRUÇÕES DE ARQUIVAMENTO

Como observamos em relação às linhas 130 e 140 do programa "arquivamento de dados", as variáveis numéricas (que funcionam como pastas onde são guardados os dados numéricos) podem ser colocadas no arquivo sendo descritas após a instrução PRINT #1, na ordem em que devem ser arquivadas e apenas separadas por vírgulas.

Notamos também que o mesmo não acontece com as variáveis alfanuméricas (que funcionam como pastas onde são arquivados dados em geral, sem valor quantitativo), devendo ser as mesmas arquivadas através da instrução PRINT #1, com separação de ponto-e-vírgula, vírgula entre aspas e novamente ponto-e-vírgula.

Se for desobedecido este critério, isto é, se, por exemplo, diversas variáveis alfanuméricas forem separadas apenas por vírgulas - como no caso das variáveis numéricas - a primeira variável alfanumérica colocada após a instrução PRINT #1 assumirá os dados de todas as outras variáveis que estiverem depois dela para serem arquivadas, ficando estas "vazias" e criando a maior confusão no arquivo.

Todavia, existe outra maneira de arquivar dados atribuídos a variáveis alfanuméricas. A referida linha 140 do programa de arquivamento teria então a seguinte formatação:

```
140 PRINT #1,A$: PRINT #1,B$: PRINT #1,C$
```

Constate a validade, repetindo a experiência com essa alteração feita. O exercício servirá para fixar melhor na memória esse recurso.

15.5 FORMATAÇÃO ESPECIAL DE DADOS NUMERICOS PARA ARQUIVO

Embora conste na lista de instruções para processamento de arquivos, a instrução PRINT USING não oferece conveniência de uso na gravação de dados em fita cassete. Ao contrário, chega a causar pequenos problemas nesse uso particular, principalmente quando entra uma vírgula no formato instruído.

A prática tem demonstrado que dados numéricos devem ser gravados e lidos na sua forma original e formatados de acordo com o padrão desejado apenas no estágio de impressão no vídeo ou pela impressora. Apenas a título de experiência, faremos uso da referida instrução na modalidade em que não oferece problemas sérios.

Sejam os seguintes valores que devem ser arquivados em formato monetário, com duas casas decimais para centavos e sem divisão de três em três casas na parte inteira: 123.453, 3721.5, 52.434. (Lembremo-nos de que o ponto substitui a vírgula em números, no computador.)

Usando ainda o mesmo programa de arquivamento, digitamos as li-

nhas 100 e 130 do mesmo como seguem:

```
100 A = 123.453: B = 3721.5: C = 52.434
130 PRINT #1, USING "#####.##";A,B,C
```

Rode o programa e note que a formatação não será perfeita, cortando apenas a terceira casa decimal dos números processados.

15.6 FORMATAÇÃO ESPECIAL DE DADOS ALFANUMERICOS PARA ARQUIVO

Sabemos que variáveis alfanuméricas não aceitam atribuições com aspas e, quando definidas pela instrução INPUT, o que estiver depois de uma vírgula, na atribuição, é considerado EXTRA, não sendo registrado na variável.

Supondo que devamos arquivar dados em frases como:

SANGUE, SUOR, LÁGRIMAS. e
A EXPRESSÃO "TOP-DOWN", DE ORIGEM INGLESA, SIGNIFICA "DE CIMA PARA BAIXO".

e ainda usando o programa mencionado, fazemos as linhas 110 e 140 como seguem:

```
110 A$ = "SANGUE, SUOR, LÁGRIMAS.": B$ = "A EXPRES-
    SÃO 'TOP-DOWN', DE ORIGEM INGLESA, SIGNIFICA
    'DE CIMA PARA BAIXO'."
140 PRINT #1,A$: PRINT #1,B$
```

A atribuição da variável A$ foi direta (sem INPUT); portanto, comporta vírgulas.

A atribuição da variável B$, também direta, aceita vírgulas, porém não aceita aspas, e foi feita substituindo-se as aspas por apóstrofos.

Quando não for conveniente adotar este recurso - uso de apóstrofos em lugar de aspas - a atribuição deverá ser feita através da instrução LINEINPUT. Nesse caso, a linha 110 deverá ser elaborada na seguinte forma:

```
110 A$ = "SANGUE, SUOR, LÁGRIMAS.": LINEINPUT B$
```

devendo o programa ser completado normalmente. Quando o mesmo receber o comando RUN, parará na segunda instrução da linha 110, colocando no vídeo o cursor para indicar que está aguardando entrada de dados para a variável B$. Deverá então ser digitada a segunda frase, na sua forma original, com aspas, vírgulas e o que for necessário. Após ser pressionada a tecla RETURN, o programa prosseguirá normalmente em execução.

Vamos juntar todas as linhas do programa assim formado, inicialmente com a primeira versão da linha 110, para melhor compreensão. Com relação às variáveis numéricas, vamos arquivá-las na sua forma original e formatá-las num mesmo padrão quando forem escritas no vídeo:

15.7 PROGRAMA "ARQUIVANDO DADOS EM ARQUIVO CASSETE"

```
50 CLEAR 500
100 A=5123.457#: B=3721.5: C=111.098
110 A$="SANGUE, SUOR, LÁGRIMAS.": B$="A
 EXPRESSÃO 'TOP DOWN', DE ORIGEM INGLESA
, SIGNIFICA 'DE CIMA PARA BAIXO'."
120 OPEN "CAS:DATA" FOR OUTPUT AS #1
130 PRINT #1,A,B,C
140 PRINT #1,A$: PRINT #1,B$
150 CLOSE #1
160 END
```

O programa de leitura e impressão dos dados deverá ser ajustado na linha 230, em razão das atribuições feitas às variáveis A$ e B$ na linha 110. A linha 250 deverá ser alterada também, a fim de que os valores das variáveis A, B e C sejam escritos em padrão monetário. O programa de leitura deverá ficar como segue:

15.8 PROGRAMA "LEITURA DE DADOS DE ARQUIVO CASSETE"

```
200 CLEAR 500
210 OPEN "CAS:DATA" FOR INPUT AS #1
220 INPUT #1,A,B,C
230 LINEINPUT #1,A$: LINEINPUT #1,B$
```

```
240 IF EOF(1)=-1 THEN CLOSE #1
250 PRINTUSING "$$######,.##";A;B;C
260 PRINT A$: PRINT B$
270 END
```

Os dois programas deverão ser digitados e introduzidos no computador ao mesmo tempo.

Depois de conferidos e estando o gravador pronto para gravar, comande RUN.

Quando aparecer no vídeo o OK, rebobine a fita cassete ao ponto inicial da gravação, coloque o gravador no modo PLAY e comande em seguida GOTO 200 / RETURN, para recuperar ou ler os dados "arquivados" na fita.

Faça em seguida a mesma experiência com a segunda versão da linha 110, conforme procedimento indicado no item 15.6.

15.9 ORGANIZANDO ARQUIVO SEQÜENCIAL

Vimos como gravar e ler dados em arquivo CAS, isto é, arquivo em fita cassete.

Por razões didáticas, lidamos com número irrisório de dados, os quais foram gravados e lidos, ou recuperados da fita, de uma só vez, não propiciando uma idéia perfeita de um arquivo "seqüencial".

Vamos agora organizar arquivos que comportam considerável volume de dados ou registros e que nos darão uma idéia mais abrangente de suas características e possibilidades.

Teoricamente, um arquivo não tem limites quanto à quantidade de registros. Pode ocupar 1/10 de uma fita cassete ou uma quantidade infinita delas para arquivar tantos dados ou registros quantos forem necessários.

Todavia, em razão da praticabilidade que devem oferecer e dependendo da finalidade a que se destinam, alguns tipos de arquivo, quando muito volumosos, devem ser divididos em blocos ou grupos dimensionados em função da capacidade da memória RAM instalada no microcomputador, como será visto adiante.

15.10 MODALIDADES PARA GRAVAÇÃO E LEITURA DE ARQUIVOS CAS

PRIMEIRA MODALIDADE

A gravação dos dados na fita é efetuada automaticamente pelo computador, através de um programa gerenciador, à medida que os dados são digitados. Neste caso, o arquivo não tem limites.

Esta modalidade se presta especialmente para arquivos de dados permanentes, mala-direta e outros em que os registros são permanentes.

Leitura dos dados: nesta modalidade de arquivo os dados ou registros não são acessados diretamente. Para ler um dado que se acha no 219º registro, por exemplo, os 218 registros anteriores devem ser percorridos na fita pelo cabeçote de leitura do gravador para que o mesmo seja obtido ou carregado na memória do computador.

SEGUNDA MODALIDADE

A gravação dos dados em fita é efetuada também com auxílio de um programa gerenciador, após todos os dados terem sido digitados ou inseridos no microcomputador.

Esta modalidade de arquivo se aplica quando os dados devem ser manipulados com certa freqüência, para os mais diversos fins: ordenação, atualização, alteração, cancelamento, nova inserção etc. Por essa razão, são acumulados na memória do computador, antes de serem gravados em fita, fato que, em princípio, limita o arquivo à capacidade de memória RAM do microcomputador, mas apenas em princípio, pois o arquivo pode ser dividido em grupos, classificados por letras ou códigos, como se, num arquivo convencional de escritório, fossem usadas diversas gavetas, cada uma com certo número de pastas em determinada ordem. Por outro lado, pode a memória RAM do microcomputador ser expandida. O MSX comporta mais do que 500 "kbytes" de RAM.

Leitura da fita: deve ser feita pela totalidade, em função da capacidade da memória RAM do microcomputador.

Vantajosamente, todavia, uma vez carregados todos os registros da fita no computador, o acesso a qualquer registro é direto e aleatório, o que permite que a busca de um dado, por qualquer dos campos do

registro, seja extremamente rápida, ou imediata.

Como a diferença entre ler a fita de um arquivo para obter determinado registro e carregar todos os registros da fita no microcomputador é bastante relativa, pois depende da posição em que se encontra o registro na fita, a escolha da modalidade a ser adotada dependerá de diversos fatores a serem considerados.

Exporemos a seguir as duas modalidades, com programas aplicativos, a fim de que suas características possam ser bem percebidas.

15.11 ARQUIVO SEQÜENCIAL DE DADOS PERMANENTES

Como sabemos, um arquivo pode conter dados permanentes ou temporários.

Um arquivo de dados biográficos de personagens da História, por exemplo, enquadra-se na modalidade de arquivo de dados permanentes. Nossa experiência inicial será feita com um arquivo dessa modalidade.

A primeira providência necessária será elaborar o programa gerenciador:

15.12 PROGRAMA GERENCIADOR DE ARQUIVOS DE DADOS PERMANENTES

```
10 SCREEN0:WIDTH40:COLOR1,3
20 MAXFILES=1:KEYOFF
30 CLEAR 5000:C=1
40 CLS:LOCATE 10,7:PRINT"PROGRAMA BIOGR
AFIAS"
50 PRINT,,TAB(8)"G - COLETA E GRAVA DAD
OS"
60 PRINT,,TAB(8)"L - LÊ/RECUPERA OS DAD
OS"
70 INPUTR$:IF R$="G"ORR$="g"THEN80 ELSE
 300
80 CLS:LOCATE9,10:PRINT"AGUARDE UNS INS
TANTES...":OPEN"CAS:BIOGR"FOR OUTPUT AS
 #1
```

```
90 MOTOROFF: CLS
100 IFC=1 THEN PRINT,,SPC(7)"Comece a d
igitar os dados:":FORA=1TO400:NEXTA:GOT
O 130
110 IFC>1THEN PRINT,,SPC(10)"Outra entr
ada? (s/n)":R$=INPUT$(1)
120 IFR$="S"ORR$="s"THEN 130 ELSE 200
130 CLS:PRINT,,TAB(12)"BIOGRAFIA Nº";C
140 PRINT,,TAB(35)"NOME:":LINEINPUTA$
150 PRINT,,TAB(25)"DADOS PESSOAIS:":LIN
EINPUTB$
160 PRINT,,TAB(27)"OUTROS DADOS:":LINEI
NPUTC$
170 MOTORON
180 PRINT#1,A$:PRINT#1,B$:PRINT#1,C$
190 C=C+1:GOTO90
200 CLOSE#1:GOTO40
300 CLS:LOCATE0,10:PRINT"REBOBINE A FIT
A AO PONTO INICIAL E COLO-QUE O GRAVADO
R NA MODALIDADE 'PLAY'.  EM SEGUIDA, DI
GITE O NOME A SER PROCURADO."
310 PRINT:C=0:INPUT P$
320 PRINT,,SPC(8)"AGUARDE UNS INSTANTES
..."
330 OPEN"CAS:BIOGR"FOR INPUT AS #1
340 IFEOF(1)=-1THEN400
350 LINEINPUT#1,A$
360 LINEINPUT#1,B$
370 LINEINPUT#1,C$
380 C=C+1
390 IF A$=P$ THEN 450 ELSE 340
400 PRINT,,TAB(10)"NOME NÃO ENCONTRADO.
"
410 CLOSE #1
420 PRINT:PRINT STRING$(40,192):PRINT,,
SPC(5)"TECLE 'M' PARA VOLTAR AO MENU.":
R$=INPUT$(1)
430 IF R$="M" OR R$="m" THEN 40 ELSE 43
0
450 CLOSE # 1
```

```
460 CLS:PRINT TAB(12)"BIOGRAFIA Nº";C
470 PRINT TAB(35)"NOME:":PRINTA$
480 PRINT,,TAB(25)"DADOS PESSOAIS:":PRI
NTB$
490 PRINT,,TAB(27)"OUTROS DADOS:":PRINT
C$
500 GOTO410
```

Digitado e conferido o programa, convirá gravá-lo no começo da fita que será usada para arquivar os dados, já que o programa gerenciador deverá ser carregado no microcomputador todas as vezes que tiver de ser feita uma consulta ou leitura de um dado arquivado.

15.13 ARQUIVANDO DADOS PERMANENTES

Estando o programa na memória do computador, aperte a tecla F5. Aparecerá no vídeo o pequeno "menu" do programa.

Em seguida, tecle G e RETURN. Através do vídeo, o computador começará a solicitar os dados para arquivar: NOME, em seguida DADOS PESSOAIS, por fim DADOS GERAIS. Como é usada a instrução LINEINPUT para entrada de dados, aparecerá apenas o cursor (em vez do sinal de interrogação) após cada item.

Digitados os dados de cada item (até 254 espaços cada), deverá ser pressionada a tecla RETURN. Os dados serão então armazenados em um "buffer" do computador, correspondente ao arquivo aberto pela instrução OPEN, e, em seguida, gravados automaticamente na fita, desde que o gravador esteja conectado adequadamente ao computador.

A entrada de dados foi organizada de maneira que facilite o controle dessas quantidades e a separação silábica: seis linhas completas de 40 espaços e uma última com 14 espaços poderão ser utilizadas para cada item. Nessa disposição será fácil controlar os espaços.

Se 508 espaços não forem suficientes para os itens DADOS PESSOAIS e OUTROS DADOS, parte destes poderá ser arquivada no item NOME.

Se durante a digitação de dados for pressionada a tecla RETURN antes do tempo, ou ocorrer algum outro lapso, pressione CTRL e STOP simultaneamente e, em seguida, faça o comando direto GOTO 130 / RETURN e

digite novamente os dados da biografia interrompida.

Se não desejar continuar a entrada de dados, pressione a tecla N quando surgir no vídeo a pergunta OUTRA ENTRADA? Pressione a tecla S para continuar a entrada de dados.

Ao ser encerrada a entraca de dados, o "menu" será exibido automaticamente no vídeo e os dados já estarão arquivados na fita.

15.14 LENDO OU RECUPERANDO DADOS PERMANENTES

Estando o "menu" do programa gerenciador no vídeo, pressione a tecla L do microcomputador. Ato contínuo, o usuário será lembrado através do vídeo que a fita cassete que contém os dados deverá ser rebobinada até o ponto inicial da gravação (de dados, não do programa gerenciador), devendo o gravador ser colocado na modalidade PLAY ou LOAD e em seguida ser digitado o nome a ser procurado. Feito isso e pressionada a tecla RETURN, o computador acionará o gravador e começará a "busca" no arquivo (fita) e logo exibirá os dados arquivados sob o nome indicado ou informará não ter encontrado tal nome caso o mesmo não conste do arquivo.

É importante frisar que o nome a ser procurado no arquivo deve ser digitado exatamente como foi arquivado. Se for digitado com S um nome arquivado com Z , ou sem acento quando arquivado com acento ou, ainda, digitado todo em letras maiúsculas quando arquivado apenas com as iniciais maiúsculas, não será encontrado pelo computador no arquivo.

15.15 SITUAÇÕES DE USO DO ARQUIVO DE DADOS PERMANENTES

Algumas situações em que um arquivo de dados permanentes é usado para consultas ou obtenção de dados:

a) As consultas são esporádicas ou pouco freqüentes e em ordem salteada, isto é, não obedecem a ordem alfabética.
b) As consultas são bastante ou muito freqüentes e em ordem também salteada.
c) As consultas são constantes e em ordem consecutiva ou alfabética.

No caso a, o arquivamento poderá ser feito num só grupo e, de preferência, fora de ordem alfabética, para evitar que tenha de ser lida a fita quase até o fim para buscar dados arquivados sob o nome de Wagner ou Ziraldo, por exemplo.

No caso b, será conveniente, para fins de consulta, que o arquivamento seja feito em grupos de registros em diversas fitas ou em segmentos de uma mesma fita (dividindo-a em três partes de cada lado, por exemplo, com marcas bem nítidas sobre uma pequena tira de papel adesivo de cor clara colada junto à janela para controle de bobinamento da fita) e, nesse caso, em ORDEM ALFABETICA APENAS PELA INICIAL DO NOME, tomando como base para arquivamento a maior incidência de determinadas letras.

Supondo uma fita dividida em seis segmentos (três em cada lado), a divisão poderia ser aproximadamente como exemplificada a seguir:

1º grupo: letras A a D
2º grupo: letras E a H
3º grupo: letras I a L
4º grupo: letras M a O
5º grupo: letras P a S
6º grupo: letras T a Z

Nesse caso, para uma consulta de dados arquivados sob o nome de Marco, por exemplo, seria usado o quarto segmento da fita (correspondendo ao primeiro do segundo lado), o que abrevia bastante o processo de busca.

No caso c, o arquivamento de dados poderá ser feito conforme a ou b, dependendo da quantidade de dados e da conveniência do usuário, e a parte da leitura do programa de gerenciamento, a partir da linha 300, deverá ser substiuída pela que segue:

```
300 C=1:CLS:LOCATE0,10:PRINT"REBOBINE A
 FITA AO PONTO INICIAL E COLO-QUE O GRA
VADOR NA MODALIDADE 'PLAY'. EM SEGUID
A, PRESSIONE A TECLA 'RETURN'.":R$=INPU
T$(1)
310 IFASC(R$)=13THEN 320 ELSE 310
320 PRINT,,SPC(8)"AGUARDE UNS INSTANTES
..."
```

```
330 OPEN"CAS:BIOGR"FOR INPUT AS #1
340 IFEOF(1)=-1THEN460
350 LINEINPUT#1,A$
360 LINEINPUT#1,B$
370 LINEINPUT#1,C$
380 MOTOROFF
390 CLS:PRINTTAB(12)"BIOGRAFIA Nº";C;
400 PRINTTAB(35)"NOME:":PRINTA$
410 PRINT,,TAB(25)"DADOS PESSOAIS:":PRI
NTB$
420 PRINT,,TAB(27)"OUTROS DADOS:":PRINT
C$:C=C+1
430 PRINTSTRING$(40,192):PRINT,,SPC(3)"
PRESSIONE 'RETURN' PARA PROSSEGUIR":R$=
INPUT$(1)
440 IFASC(R$)=13THEN450 ELSE 440
450 MOTORON:GOTO340
460 MOTOROFF:CLOSE#1:PRINT,,"FIM DO ARQ
UIVO."
```

Nessa forma, todos os registros do arquivo serão mostrados no vídeo a partir do ponto inicial da fita ou do ponto inicial do segmento da fita em que se encontram, seqüencialmente, um de cada vez, a cada toque da tecla RETURN.

Se todos os registros do arquivo deverem ser impressos em papel - através de impressora conectada ao microcomputador - bastará fazer as seguintes alterações no programa de leitura, nesta última versão:

LINHAS 380, 430 e 440: Eliminar completamente.
LINHA 390: Eliminar a instrução CLS.
LINHAS 390 a 420: Alterar a instrução PRINT para LPRINT e eliminar as vírgulas após PRINT (linhas 410 e 420).
LINHA 450: Alterar para LPRINT: LPRINT: GOTO 340
LINHA 460: Eliminar a instrução MOTOR OFF.
AJUSTAR a impressora para imprimir em 40 colunas.

15.16 ARQUIVO SEQÜENCIAL DE DADOS TEMPORÁRIOS

(Com acesso direto quando carregado no microcomputador)

Como exemplo de arquivo de dados temporários, vamos adotar um fichário de empregados de uma empresa. Nesse tipo de arquivo, diversos dados podem sofrer alterações: endereço, função, salário etc. Cancelamentos e novas inserções também ocorrem (estamos falando no fichário de "pessoal em atividade", não no fichário fiscal). O fichário poderá ser ordenado pelo número de registro do empregado, pelo nome, pela função ou outro campo, de forma que seja facilitada a busca de uma ficha ou de um grupo delas.

Geralmente, uma ficha ou um registro de arquivo dessa modalidade engloba número considerável de campos. No exemplo dado a seguir adotaremos apenas quatro campos: (1) número de registro, (2) nome, (3) função e (4) dados gerais. A ordenação e o agrupamento poderão ser feitos por número de registro, por nome ou por função.

15.17 PROGRAMA GERENCIADOR DE ARQUIVO DE DADOS TEMPORÁRIOS

```
10 SCREEN0:WIDTH40:COLOR 15,1
20 MAXFILES=1:KEYOFF
30 LOCATE12,12:INPUT"QUANTAS FICHAS";Q
40 CLEAR 200*Q:DEFSNGA-Z:Q=FRE("")/200
50 DIMR(Q+1):DIMN$(Q+1):DIMF$(Q+1):DIMD
$(Q+1)
60 A$(1)="NÚMERO DE REGISTRO":A$(2)="NO
ME":A$(3)="FUNÇÃO":A$(4)="DADOS GERAIS"
100 REM MENU
110 CLS:LOCATE 12,3:PRINT"FICHÁRIO PESS
OAL"
120 PRINT,,
130 PRINT,,TAB(7)"I - INICIA REGISTROS
EM FICHAS"
140 PRINT,,TAB(7)"C - CONTINUA REGISTRO
S"
150 PRINT,,TAB(7)"E - EXIBE FICHAS NO V
IDEO"
160 PRINT,,TAB(7)"O - ORDENA FICHAS"
170 PRINT,,TAB(7)"P - PROCURA FICHA PAR
A                    CONSULTA OU A
LTERAÇÃO"
```

```
180 PRINT,,TAB(7)"G - GRAVA FICHAS EM F
ITA"
190 PRINT,,TAB(7)"R - RECUPERA FICHAS D
A FITA"
200 PRINT,,TAB(7)"L - LISTA FICHAS EM P
APEL"
210 R$=INPUT$(1)
220 IFR$="I"ORR$="i" THEN400
230 IFR$="C"ORR$="c" THEN550
240 IFR$="E"ORR$="e" THEN600
250 IFR$="O"ORR$="o" THEN1000
260 IFR$="P"ORR$="p" THEN1200
270 IFR$="G"ORR$="g" THEN1400
280 IFR$="R"ORR$="r" THEN1500
290 IFR$="L"ORR$="l" THEN1700
300 GOTO210
400 REM entrada de dados
410 IFC=0THENCLS:PRINT,,TAB(12)"FICHAS
ESGOTADAS":GOTO530
420 IFC<0THENC=C+1:N=C
430 CLS:PRINT,,TAB(14)"FICHA Nº";C:PRIN
T
440 PRINT,,A$(1);":";:INPUTR(C)
450 PRINT,,A$(2);":";:INPUTN$(C)
460 PRINT,,A$(3);": ";:LINEINPUTF$(C):I
FLEN(F$(C))>250 THENGOSUB510:GOTO 450
470 PRINT,,A$(4);": ";:LINEINPUTD$(C):I
FLEN(D$(C))>250 THENGOSUB510:GOTO460
480 PRINT,,TAB(9)"Nova ficha? (s/n)";:R
$=INPUT$(1)
490 IFR$="S"ORR$="s"THEN410ELSE100
500 REM subrotinas
510 PRINT,," DIGITE NOVAMENTE. EXCEDEU
250 ESPAÇOS.":RETURN
520 FORK=1TO1000:NEXTK:RETURN
530 FORK=1TO1000:NEXTK:GOTO100
540 CLS:LOCATE2,12:PRINT"AS FICHAS NÃO
CONTÊM NENHUM REGISTRO":GOTO530
550 C=N:IFN<>0THEN410ELSE540
560 PRINT,,"NAO CONSTA NAS FICHAS ESSE
";A$(E):E=0:GOTO530
```

```
570 CLS:LOCATE0,10:PRINTSPC(5)"APRONTE
O GRAVADOR E TECLE 'RETURN' PARA GRAVAR
. OU QUALQUER OUTRA TECLA SE MUDAR DE I
DÉIA E QUISER VOLTAR AO MENU.":RETURN
580 CLS:LOCATE0,10:PRINT,,"O FICHÁRIO J
A CONTÉM REGISTROS. PARA RE-CUPERAR DAD
OS DA FITA, O FICHÁRIO DEVERÁESTAR VAGO
. NOVOS REGISTROS DEVEM ENTRARDEPOIS DE
 RECUPERADAS AS FICHAS DA FITA."
590 FORK=1TO3600:NEXTK:GOTO100
600 REM exibe fichas no video
610 IF N=0THEN540ELSEC=1
620 CLS:PRINT,,TAB(14)"FICHA Nº";C:PRIN
T
630 PRINT,,A$(1);R(C)
640 PRINT,,A$(2);": ";N$(C)
650 PRINT,,A$(3);": ";F$(C)
660 PRINT,,A$(4);": ";D$(C)
670 PRINT:PRINTSTRING$(40,192)
680 PRINT,,"1= Alterar, 2= Anular, 3= V
er ficha";C-1;",     4= Ver ficha";C+1;
", 5= Voltar ao MENU.":R$=INPUT$(1):IFV
AL(R$)<1ORVAL(R$)>5THEN620
690 ONVAL(R$)GOTO700,800,900,950,100
700 REM alteração
710 PRINT,,SPC(3)"Alterar: 1 = Registro
, 2 = Nome,          3 = Função, 4 = Da
dos gerais.":R$=INPUT$(1):IFVAL(R$)<1OR
VAL(R$)>4THEN620
720 ONVAL(R$)GOTO730,740,750,760
730 PRINT,,"Digite o novo número de reg
istro:":PRINT:INPUTNR:R(C)=NR:NR=0:GOTO
620
740 PRINT,,"Digite o novo nome:":PRINT:
INPUTNN$:N$(C)=NN$:NN$="":GOTO620
750 PRINT,,"Digite nova função:":PRINT:
LINEINPUTNF$:F$(C)=NF$:NF$="":GOTO620
760 PRINT,,"Digite novos dados:":PRINT:
LINEINPUTND$:D$(C)=ND$:ND$="":GOTO620
800 REM anulaçao
```

```
810 PRINT,,"Confirma anulação? (s/n)":R
$=INPUT$(1)
820 IFR$="S"ORR$="s"THEN830ELSE620
830 FORA=C+1TON
840 R(A-1)=R(A):N$(A-1)=N$(A)
850 F$(A-1)=F$(A):D$(A-1)=D$(A)
860 NEXTA:R(A-1)=0:N$(A-1)="":F$(A-1)="
":D$(A-1)=""
870 N=N-1
880 PRINT,,TAB(7)"ANULAÇÃO FEITA":GOSUB
520:GOTO620
900 REM retrocede ficha
910 IFC=1THEN620ELSEC=C-1:GOTO620
950 REM avança ficha
960 IFC=)NTHEN620ELSE C=C+1:GOTO620
1000 REM ordenação
1010 IFN=0THEN540:CLS:COLOR1,3
1020 CLS:LOCATE7,9:PRINT"ORDENAÇAO A SE
R FEITA POR:"
1030 PRINT,,TAB(7)"1 - ";A$(1);"?"
1040 PRINT,,TAB(15)"2 - ";A$(2);"?"
1050 PRINT,,TAB(22)"3 - ";A$(3);"?":R$=
INPUT$(1):IFVAL(R$)(1ORVAL(R$))3THEN102
0
1060 E=VAL(R$)
1070 PRINT,,TAB(10)"Ordenação por ";A$(
E)
1080 FORO=1TON-1
1090 FORP=O+1TON
1100 GOSUB1130
1110 NEXTP:NEXTO
1120 PRINT,,TAB(12)"ORDENAÇÃO PRONTA":G
OTO530
1130 IFE=1ANDR(O))R(P)THENGOSUB1170
1140 IFE=2ANDN$(O))N$(P)THENGOSUB1170
1150 IFE=3ANDF$(O))F$(P)THENGOSUB1170
1160 RETURN
1170 SWAPR(O),R(P):SWAPN$(O),N$(P)
1180 SWAPF$(O),F$(P):SWAPD$(O),D$(P)
1190 RETURN
1200 REM procura
```

```
1210 IFN=0THEN540:CLS:COLOR15,2
1220 CLS:LOCATE8,9:PRINT"PROCURA A SER
FEITA POR:"
1230 PRINT,,TAB(8)"1 - ";A$(1);"?"
1240 PRINT,,TAB(15)"2 - ";A$(2);"?"
1250 PRINT,,TAB(22)"3 - ";A$(3);"?":R$=
INPUT$(1):IFVAL(R$)<1ORVAL(R$)>3THEN121
0
1260 E=VAL(R$)
1270 PRINT,,"DIGITE 0 ";A$(E);":";:INPU
TR$
1280 ONEGOTO1290,1300,1310
1290 FORC=1TON:IFR(C)=VAL(R$)THEN620ELS
ENEXTC:GOTO560
1300 FORC=1TON:IFN$(C)=R$THEN620ELSENEX
TC:GOTO560
1310 FORC=1TON:IFF$(C)=R$THEN620ELSENEX
TC:GOTO560
1400 REM grava fichas em fita
1410 IFN=0THEN540:COLOR1,15
1420 GOSUB570:R$=INPUT$(1)
1430 IFASC(R$)=13THEN1440ELSE100
1440 CLS:LOCATE12,10:PRINT"GRAVANDO FIC
HAS"
1450 OPEN"CAS:FIPES"FOROUTPUTAS#1
1460 FORC=1TON
1470 PRINT#1,R(C):PRINT#1,N$(C):PRINT#1
,F$(C):PRINT#1,D$(C)
1480 NEXTC:CLOSE#1:GOTO100
1490 PRINT,,TAB(12)"GRAVAÇÃO FEITA!":GO
TO530
1500 REM recupera registros da fita
1510 IFN<>0THEN580
1520 GOSUB570:R$=INPUT$(1)
1530 IFASC(R$)=13THEN1540ELSE100
1540 CLS:LOCATE8,10:PRINT"RECUPERANDO F
ICHAS DA FITA"
1550 OPEN"CAS:FIPES"FORINPUTAS#1
1560 FORC=1TOQ
1570 IFEOF(1)=-1THEN1590
```

```
1580 INPUT#1,R(C):INPUT#1,N$(C):LINEINP
UT#1,F$(C):LINEINPUT#1,D$(C):NEXTC
1590 CLOSE#1:N=(C-1)
1600 PRINT,,TAB(10)"RECUPERAÇÃO FEITA!"
:GOTO530
1700 REM lista fichas em papel
1710 IFN=0THEN540:CLS:LOCATE12,9:PRINT"
LISTANDO FICHAS
 PELA IMPRESSORA"
1720 FORC=1TON
1730 LPRINT:LPRINT:LPRINT
1740 LPRINTA$(1);":";R(C):LPRINT
1750 LPRINTA$(2);": ";N$(C):LPRINT
1760 LPRINTA$(3);": ";F$(C):LPRINT
1770 LPRINTA$(4);": ";D$(C)
1780 NEXTC
1790 PRINT,,TAB(12)"IMPRESSÃO PRONTA":G
OTO530
```

Depois de digitado e conferido, convirá fazer logo em seguida duas ou três gravações do programa, prevenindo qualquer eventualidade. Um despercebido toque na tampa de "slot" de cartucho, ou no botão "reset", ou uma interrupção repentina no fornecimento de energia elétrica podem destruir o programa ou "apagá-lo" na memória do computador.

15.18 OPERANDO ARQUIVO DE DADOS TEMPORÁRIOS

O arquivo de dados temporários é operado através do programa gerenciador.

Estando o programa gerenciador na memória do computador, aperte a tecla F5. Surgirá no vídeo a pergunta QUANTAS FICHAS? Uma vez digitada a quantidade desejada e pressionada a tecla RETURN, será apresentado no vídeo o "menu" do programa que, praticamente, orientará todas as ações do usuário para formar e operar o arquivo.

Alguns itens do "menu" são complementados por "submenus", que viabilizam com rapidez operações derivadas necessárias, e notações informativas são colocadas no vídeo em face das diversas situações eventuais, de forma que o programa se caracteriza por ser auto-explicativo

e de fácil condução.

Como dissemos antes, os dados a serem gravados em fita cássete, nesta modalidade de arquivo, são antes inseridos e acumulados na memória do computador, na sua totalidade, se ocuparem uma quantidade admissível de "bytes", isto é, uma quantidade ligeiramente inferior à da capacidade da memória RAM da máquina. No caso de os dados excederem tal limite, devem ser divididos em blocos, como já mencionado. É esta particularidade - a de acumular os dados na memória do computador - que confere ao arquivo os recursos expostos no "menu" e "submenus", permitindo que os registros sejam acessados direta e aleatoriamente, para consultas, alterações ou cancelamentos, além de permitir que novas inserções sejam "encaixadas" ou arquivadas na ordem de campo que se desejar, através do recurso de ordenação disponível.

Nosso programa gerenciador prevê uma quantidade de 200 "bytes" para cada registro ou ficha (linha 40), estimados para a seguinte distribuição:

```
  4 para números de registros com até 6 dígitos
 32 para nome
 24 para função
140 para dados gerais
200 "bytes" para cada ficha ou registro.
```

Ao ser informada ao computador a quantidade de fichas desejadas (linha 30), a instrução CLEAR 200*Q (linha 40) faz a reserva correspondente na área de variáveis alfanuméricas, onde o arquivo será provisoriamente armazenado.

Assim, se a quantidade de fichas for igual a 100, por exemplo, a reserva para os dados do arquivo será de 20.000 "bytes".

O programa gerenciador ocupa aproximadamente 4200 "bytes".

A disponibilidade mínima normal do MSX é de 28.815 "bytes".

Conclui-se que um arquivo de 100 fichas com 200 "bytes" para cada uma, em média, pode ser processado na sua totalidade, com uma sobra aproximada de 5.000 "bytes" para o microcomputador, que os utiliza parcialmente no processamento do programa e do arquivo.

Quando arquivos maiores devem ser processados, duas alternativas podem ser consideradas, como já mencionamos: expandir a memória do

microcomputador ou dividir os arquivos em blocos.

15.19 FORNECENDO E MANIPULANDO DADOS PARA O ARQUIVO

De acordo com os itens do "menu", pressionando-se a tecla I, começa o computador a solicitar os dados para os diversos campos de cada ficha ou registro. Não havendo dados para um campo, basta pressionar a tecla RETURN para prosseguimento.

A entrada de dados poderá ser sustada após a elaboração de cada ficha através da opção oferecida ao usuário pelo vídeo.

Após a interrupção de entrada de dados ou execução de qualquer estágio do programa, o "menu" é reapresentado no vídeo.

Qualquer das opções do "menu" pode ser usada a qualquer momento: exibição das fichas, ordenação, procura etc.

O item CONTINUA REGISTROS permite reiniciar a entrada de dados a partir da próxima ficha disponível, após qualquer interrupção.

O item EXIBE FICHAS NO VIDEO coloca a ficha número 1 no vídeo e permite que as seguintes sejam exibidas com o simples toque da tecla 4 do microcomputador. As fichas podem avançar ou retroceder. Pressionando-se a tecla 3, as fichas são exibidas em retrocesso. O "submenu" que acompanha cada ficha indica esses recursos e outros.

A ficha número 1 poderá ser a primeira em ordem de NÚMERO DE REGISTRO ou em ordem alfabética por um dos campos NOME ou FUNÇÃO, dependendo da ordenação precedente que tenha sido feita.

O item ORDENA FICHAS coloca em ordem por número de registro ou em ordem alfabética por um dos campos NOME ou FUNÇÃO todas as fichas existentes na memória do computador. A opção é adotada através do "submenu" do próprio item.

O item PROCURA FICHA PARA CONSULTA OU ALTERAÇÃO permite que uma ficha seja buscada por qualquer dos campos NÚMERO DE REGISTRO, NOME ou FUNÇÃO. A opção é adotada através do "submenu" do próprio item.

O item GRAVA FICHAS EM FITA permite que os registros existentes na memória do computador sejam gravados em fita cassete e supervisiona a operação.

O item RECUPERA FICHAS DA FITA permite que os registros gravados em fita passem para a memória do computador, supervisionando a operação.

O item LISTA FICHAS EM PAPEL emite listagens das fichas através de impressora, supervisionando a operação.

15.20 GRAVANDO OU ARQUIVANDO EM FITA OS REGISTROS

O arquivamento dos registros em fita é feito após todos eles terem sido digitados ou inseridos na memória do computador, através do programa gerenciador, bastando pressionar a tecla G do computador, de acordo com o "menu" do programa. Ao terminar a gravação o usuário será informado através do vídeo e o programa voltará ao "menu".

Naturalmente, o gravador deverá estar conectado ao microcomputador e pronto para gravar.

15.21 RECUPERANDO OU LENDO OS REGISTROS DA FITA

A recuperação ou leitura dos registros da fita é feita através do programa gerenciador, sendo todos eles carregados em conjunto na memória do microcomputador.

O programa gerenciador deverá ser carregado previamente no computador. Depois de acionado o programa (tecla F5) e ser informada, via teclado, a quantidade desejada de fichas, que deverá no mínimo corresponder à existente na fita, bastará pressionar a tecla R do microcomputador, em conformidade com o "menu" do programa.

O gravador deverá estar adequadamente conectado ao microcomputador, posto na modalidade PLAY ou LOAD, e a fita com os dados do arquivo posicionada pouco antes do início da gravação feita.

Ao terminar o carregamento, o usuário será informado através do vídeo e o programa voltará ao "menu" para processamento normal.

15.22 REGRAVANDO O ARQUIVO

Quando forem feitas alterações nas fichas, o arquivo deverá ser regravado, a fim de reter as informações atualizadas.

Por essa razão, será conveniente ter o arquivo gravado em fita separada da que contém o programa gerenciador, de forma que a regravação possa ser feita no lado oposto do da gravação anterior. Se novas alterações ocorrerem, a nova regravação poderá ser feita no lugar da primeira gravação, de tal sorte que sejam conservadas sempre as últimas duas gravações do arquivo, para maior segurança e eliminação de dúvidas eventuais. Em casos especiais, poderá sempre ser mantida a gravação original do arquivo em fita à parte.

15.23 CONSIDERAÇÕES FINAIS

Deve ser lembrada aqui a observação já feita de que a diferença de tempo entre a leitura de uma fita de arquivo de dados permanentes e uma fita de arquivo de dados temporários com número equivalente de registros é muito relativa, e pode mesmo inexistir em função da posição em que se encontra um registro a ser lido na primeira modalidade.

A despeito de qualquer diferença de tempo existente, todavia, não há dúvida de que a modalidade de arquivo de dados temporários oferece recursos muito mais amplos e maior flexibilidade, sendo quase certo que sua adoção é preferível na maioria dos casos de arquivos em fita cassete.

Contudo, apenas a experiência e o conhecimento específico de cada caso é que podem orientar quanto à escolha da modalidade mais adequada a cada finalidade.

Linguagem de máquina 16

Se compararmos a velocidade de processamento de programas em BASIC com a de programas em linguagem de máquina, ou "Assembly", concluiremos que a segunda é, na maioria dos casos, dezenas de vezes mais rápida do que a primeira.

Isto não significa, porém, que o processamento de programas em BASIC seja lento, especialmente com referência ao MSX que, por ser dotado de três microprocessadores - central, de vídeo e de áudio -, é um dos microcomputadores de 8 "bits" mais rápidos que conhecemos na atualidade.

Para ser "entendida" pelo computador, a linguagem BASIC é convertida passo a passo, durarte a execução de um programa, em linguagem de máquina. O computador não "entende" outra linguagem.

É nessa conversão, feita internamente no computador por um programa interpretador existente na ROM, que reside a diferença de velocidade de processamento entre ambas as linguagens.

Mas, enquanto programar em BASIC é tarefa relativamente fácil, pois a linguagem foi criada com o propósito de facilitar e generalizar o uso de microcomputadores, programar em linguagem de máquina demanda estudo paciente e, muitas vezes, consideravelmente demorado.

Até que ponto vale a pena ser efetuado é questão que depende de fatores peculiares a cada situação e do grau de conveniência existente. Não há dúvida, entretanto, de que deve ser realizado através de estudo especializado, baseado em literatura específica de boa qualidade e, enfatizamos, somente depois de ter sido explorada toda potencialidade do computador através do BASIC.

No MSX, dada sua velocidade normal de processamento, é bem pouco provável que o usuário tenha de recorrer ao uso de linguagem de máquina. Informações internacionais reportam que não chega a 5% o número de usuários do MSX que adotam esse recurso.

No âmbito limitado deste capítulo, estudaremos as noções básicas e as instruções disponíveis no MSX para tratar e operar programas em linguagem de máquina.

Referir-nos-emos à linguagem de máquina como LM.

16.1 MODALIDADES DE PROGRAMAS EM LM

1ª) O programa é inteiramente em códigos LM. Denominaremos esse tipo de programa como LMT (= programa em LM total).

2ª) O programa é constituído por sub-rotinas em LM de um programa em BASIC, considerado gerenciador, ainda que este tenha apenas as linhas de instruções que definem o(s) endereço(s) de início de execução do programa e/ou de suas rotinas em LM, através da instrução DEFUSR, e as de chamada para sua execução, por meio da função USR. Denominaremos este tipo de programa como LMP (= programa em linguagem de máquina parcial).

16.2 COMO GRAVAR EM FITA CASSETE UM PROGRAMA EM LM

Programa do tipo LMT deve ser gravado através do comando BSAVE. Veja a esse respeito o Capítulo 14.

Programa do tipo LMP deve ser gravado por meio de um dos comandos CSAVE ou SAVE. O assunto é tratado no Capítulo 14.

16.3 COMO CARREGAR DE FITA CASSETE PARA A MEMÓRIA DO MSX UM PROGRAMA EM LM

Programa do tipo LMT deve ser carregado por intermédio do comando BLOAD.

Programa do tipo LMP deve ser carregado através de um dos comandos CLOAD ou LOAD, conforme tenha sido gravado.

16.4 COMO INSERIR NA MEMÓRIA DO MSX UM PROGRAMA EM LM AINDA NÃO GRAVADO

Depois de elaborado no papel, um programa em LM é inserido na memória do computador através da instrução POKE.

INSTRUÇÃO POKE

Permite introduzir códigos diretamente nas posições ou endereços de memória RAM do computador. Sua sintaxe é:

POKE |endereço de memória|,|código|

FUNÇÃO PEEK

Permite verificar os códigos constantes nas posições de memória do computador. Sua sintaxe é:

PRINT PEEK (endereço de memória)

Um programa em LM pode ser inserido em linha(s) REM(s) ou, preferivelmente, em área reservada no topo da memória RAM.

16.5 INSERINDO EM LINHA REM UM PROGRAMA EM LM

Procedimentos:

- Digitar uma linha de instrução REM com caracteres quaisquer (.x,1 etc.) em quantidade rigorosamente igual à de códigos do programa em LM. Por exemplo, se o programa for constituído por 20 códigos:

```
1 REM....................
```

- Digitar um dos seguintes programas montadores:

Se os códigos estiverem em sistema decimal:

```
10 FOR E = 32774! TO 32793!
20 INPUT C
30 POKE E, C
40 NEXT E
```

Se os códigos estiverem em sistema hexadecimal:

```
10 FOR E = 32774! TO 32793!
20 INPUT C$
30 C = VAL("&H"+C$)
40 POKE E, C
50 NEXT E
```

Observações: o endereço inicial para inserção dos códigos deverá ser rigorosamente o indicado (32774). Se for adotada outra linha REM, o endereço inicial desta para inserção de códigos deverá ser o 7º após o do último caractere da primeira linha.

A definição da variável E da linha 10 deve ser feita em relação à quantidade de códigos a serem introduzidos. Nos dois programas apresentados consideramos 20 códigos, como exemplo.

- Rodar o programa adequado e digitar os códigos um a um, à medida que a instrução INPUT da linha 20 colocar o ponto de interrogação no vídeo, indicando aguardar entrada de dados. Não esquecer de teclar RETURN após digitar cada código.
- Terminada a digitação dos códigos, deverá ser feito o comando direto LIST / RETURN para que os códigos digitados passem para a linha REM.
- Deletar as linnas 10 a 40 (ou 50) do programa montador.

Se o programa for do tipo LMT:

- Digitar as seguintes linhas de instruções:

```
10 DEFUSR = 32774
20 Z = USR(0)
```

Nota: se houver mais do que uma linha REM ou mais do que

um endereço para início de execução, devem os mesmos ser definidos através de DEFUSR, sendo as chamadas ou comandos para execução feitos através da função USR, a exemplo das linhas 10 e 20. Estas instruções adicionais deverão ser feitas com especificação da rotina ou sub-rotina considerada. Exemplo:

```
30 DEFUSR (1) = 32801
40 Z = USR 1, (0)
```

A rotina primeira é de número 0 e não precisa definição.

Se o programa for do tipo LMP:

Digitar em seguida à linha REM a listagem em BASIC, devendo constar da mesma as instruções para execução da(s) rotina(s) em LM. Tais instruções, como vimos, são constituídas por DEFUSR e USR.

INSTRUÇÃO DEFUSR

Especifica o endereço inicial de um programa ou de uma rotina em linguagem de máquina. Sua sintaxe é:

DEFUSR(n) = |endereço de início de programa ou rotina|

sendo n o número de programa ou de rotina.

Se n for omitido, o computador assumirá o valor 0 para o mesmo. n pode variar entre 0 e 9, o que indica que é possível armazenar até 10 programas ou rotinas em LM na memória do MSX.

FUNÇÃO USR

Inicia a execução de um programa ou rotina em LM. Sua sintaxe:

V = USR n(n')

sendo V qualquer variável de A a Z, de uso obrigatório, mas fictício

no caso, n o número de programa ou de rotina em LM e n' um valor também fictício, mas de uso obrigatório, podendo variar entre 0 e 9.

16.6 GRAVAÇÃO E CARREGAMENTO DE UM PROGRAMA EM LINHA REM

Um programa em LM em linha REM deve ser gravado através de CSAVE ou SAVE e carregado em conformidade, isto é, por CLOAD ou LOAD.

Observa-se, todavia, que nem todos os programas LM em linha REM voltam a funcionar depois de gravados e recuperados de fita, razão por que se recomenda inserir programas em LM no topo da RAM, conforme item 16.8, valendo o exposto apenas para estudo e aplicações especiais.

16.7 EXECUÇÃO DE UM PROGRAMA EM LM EM LINHA REM

Um programa em LM colocado em linha(s) REM(s), seja LMP ou LMT, é executado a partir das linhas de instruções em BASIC além da linha REM, através da instrução GOTO em comando direto. No exemplo citado de programa em LMT, bastará acionar o comando direto:

GOTO 10 / RETURN

16.8 INSERINDO NO TOPO DA RAM UM PROGRAMA EM LM

A primeira providência a tomar é reservar a área onde será localizado o programa.

O último endereço normalmente disponível para o usuário na RAM é 62335. Esse endereço deverá ser rebaixado a fim de "abrir" espaço para a reserva de área a ser feita.

Se o programa em LM tiver 300 códigos, por exemplo, poderá ser reservada uma área igual a 300 "bytes" ou maior. Vamos admitir uma reserva de 335 "bytes", o que acarretará o rebaixamento da área disponível para BASIC. ou para o usuário, ao endereço 62000.

A instrução utilizada para efetuar esse rebaixamento é a instrução CLEAR. Poderá ser usada no modo direto ou indireto, com a seguinte

formatação:

```
CLEAR 200, 62000
```

sendo que 200 é a reserva normal feita para variáveis alfanuméricas, a qual deve ser mantida se não houver necessidade maior.

Como devemos usar um programa montador para inserir os códigos em LM, vamos colocar a instrução CLEAR nesse programa:

```
10 CLEAR 200, 62000!
20 FOR E=62001! TO 62300!
30 READ C
40 POKE E, C
50 NEXT E
60 DATA
```

Atenção: na linha 60 devem ser colocados os códigos decimais do programa em LM, separados por vírgulas.

Se a quantidade de códigos for diferente da prevista na linha 20, esta deverá ser acertada em conformidade.

Se os códigos estiverem em sistema hexadecimal, as linhas 30 e 40 do programa montador devem ser alteradas para:

```
30 READ C$
40 POKE E, VAL("&H"+C$)
```

Terminada a digitação, deve ser feita uma conferência cuidadosa dos códigos digitados e, em seguida, ser feito o comando RUN para que os códigos introduzidos na linha DATA do programa montador passem para os endereços previstos na área reservada.

O programa montador poderá ou não ser deletado.

16.9 GRAVAÇÃO E CARREGAMENTO

Um programa na modalidade apresentada deve ser gravado em fita através do comando BSAVE, sendo o carregamento da fita para o computador feito por meio do comando BLOAD (reveja o Capítulo 14).

16.10 EXECUÇÃO DO PROGRAMA

Se foi indicado em BSAVE o endereço para início de execução do programa, o mesmo entrará automaticamente em execução após se ter completado o carregamento da fita.

Caso contrário, deverá ser feito um comando direto como segue:

```
DEFUSR = 62001: Z = USR(0)
```

16.11 COMO OPERAR PROGRAMAS EM LM

PROGRAMAS EM LMT

Geralmente entram automaticamente em execução após serem carregados de uma fita cassete. O usuário limita-se a operá-los através de mensagens emitidas pelo próprio programa através do vídeo. Sua eliminação da memória do computador geralmente só é possível fazendo-se um RESET ou desligando o computador.

Se o programa LMT tiver uma ou mais linhas de instruções em BASIC, ainda que específicas para colocá-lo em execução, enquadra-se na modalidade seguinte:

PROGRAMAS EM LMP

Geralmente são operados como programas em BASIC. As sub-rotinas em LM que o compõem são acionadas através de comandos constantes em suas linhas de instruções.

Um programa nessa modalidade pode ter até 10 sub-rotinas em LM. O início, isto é, o endereço de início de execução de cada uma, deve ser definido no programa em BASIC através da instrução DEFUSR (número da sub-rotina) = |endereço da sub-rotina|, como vimos antes.

O acionamento ou a chamada para execução de cada sub-rotina é feito através da função USR |número de rotina|,(argumento fictício). O argumento fictício, como vimos antes, deve ser qualquer número entre 0 e 9, não podendo ser dispensado.

Modo gráfico no MSX 17

O modo gráfico é utilizado para representar no vídeo pontos, linhas, desenhos, figuras e gráficos de forma geral.

O modo gráfico subdivide-se em:

- MODO GRÁFICO DE ALTA RESOLUÇÃO, correspondente a SCREEN 2.
- MODO GRÁFICO DE BAIXA RESOLUÇÃO, correspondente a SCREEN 3.

No modo SCREEN 2, o vídeo dispõe de 49152 pontos gráficos ou de imagem, distribuídos em 192 linhas horizontais de 256 pontos cada uma ou 256 colunas de 192 pontos cada uma.

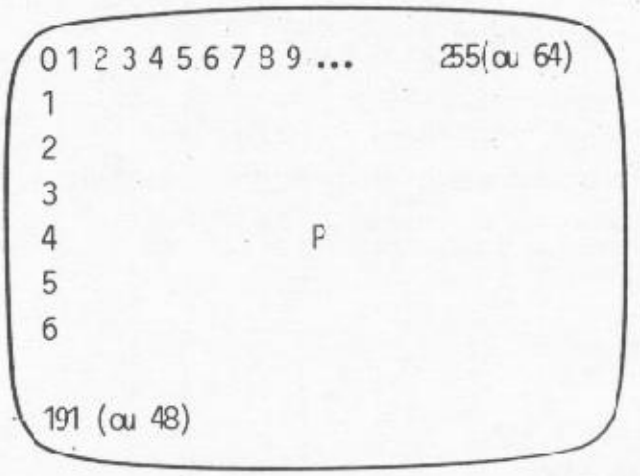
No modo SCREEN 3, o vídeo dispõe de 3072 pontos, distribuídos em 48 linhas horizontais de 64 pontos cada uma ou 64 colunas de 48 pontos cada uma.

Os pontos de imagem são determinados pelo encontro das coordenadas X e Y, sendo que X é a coordenada horizontal (linha) e Y a coordenada vertical (coluna).

A ordem de indicação para comandos ou instruções é X,Y.

O PONTO INICIAL da TELA é 0 e situa-se no canto superior esquerdo do vídeo.

A ilustração apresentada a seguir permite uma visualização, para melhor compreensão e lembrança, da disposição de linhas e colunas na na tela de vídeo, de seus números e limites:



P = ponto central
P em screen 2: X = 128, Y = 96
P em screen 3: X = 32, Y = 24

Para serem utilizados, os modos SCREEN 2 e SCREEN 3 devem ser especificados em linhas de instruções no início de programas.

No modo gráfico as cores geradas pelo MSX são muito utilizadas.

17.1 TABELA DE CORES DO MSX

0 - TRANSPARENTE	8 - VERMELHO-MÉDIO
1 - PRETO	9 - VERMELHO-CLARO
2 - VERDE	10 - AMARELO-ESCURO
3 - VERDE-CLARO	11 - AMARELO-CLARO
4 - AZUL-ESCURO	12 - VERDE-ESCURO
5 - AZUL-CLARO	13 - ROXO
6 - VERMELHO-ESCURO	14 - CINZA
7 - AZUL-CIANO	15 - BRANCO

INSTRUÇÃO COLOR

Determina as cores a serem utilizadas na tela de vídeo.

Na sua forma plena, a instrução COLOR tem a seguinte configuração:

COLOR n, n', n''

sendo que n, n' e n'' representam os números das cores, conforme tabela apresentada e especificam:

n - cor do primeiro plano.
n' - cor do segundo plano ou de fundo.
n'' - cor da moldura da tela.

17.2 DESENHOS NO MSX

Instruções usadas:

CIRCLE, DRAW, LINE, PAINT, POINT, PRESET, PSET

INSTRUÇÃO PSET - Desenha um ponto no vídeo, na posição de encontro das coordenadas X e Y.

INSTRUÇÃO PRESET - Apaga no vídeo o ponto desenhado por PSET na posição das coordenadas X e Y.

Desenho de um ponto no centro do vídeo:

```
10 SCREEN 2
20 PSET (128,95)
40 GOTO 40
```

Quando usada após PSET, a instrução STEP desloca a posição do ponto pelos valores indicados, em relação a uma posição anterior. Acrescente no programa acima:

```
30 PSET STEP (-20,20)
```

A linha 40 "mantém" o ponto no vídeo. Experimente desativá-la e verifique o que ocorre. Para desativá-la, coloque um REM após o número da linha.

Desenho de uma linha horizontal e outra vertical com PSET:

```
10 COLOR 15,4,11: SCREEN 2
20 FOR A=50 TO 150
30 PSET (A,96)
```

```
40 PSET (128,A)
50 NEXT A
60 GOTO 60
```

Desenho de linhas diagonais com PSET:

```
10 COLOR 15,4,11: SCREEN 2
20 B=1
30 FOR A=128 TO 178
40 PSET (A,96-B)
50 PSET (A,96+B)
60 B=B+1
70 NEXT A
80 B=1
90 FOR A=128 TO 78 STEP -1
100 PSET (A,96+B)
110 PSET (A,96-B)
120 B=B+1
130 NEXT A
140 GOTO 140
```

Desenho de um "céu estrelado" com PSET:

```
10 DEFINT A,B,C: SCREEN 2
20 FOR I=1 TO 100
30 A=256*RND(1)
40 B=193*RND(1)
50 C=16*RND(1)
60 PSET (A,B),C
70 PRESET (C*C,B),C
80 NEXT I
90 GOTO 90
```

O parâmetro C da linha 60 determina a cor do ponto a ser desenhado. Se não for especificado na instrução, é assumida a cor de primeiro plano que estiver ativada.

Na linha 70, o parâmetro C faz com que sejam "abertos" pontos na tela quando a cor desta não coincide com a determinada pelo valor atribuído a C. Se for omitido o parâmetro C, PRESET assume a cor de fundo vigente.

Para que PRESET "apague" um ponto desenhado por PSET é preciso

que as coordenadas de ambas sejam as mesmas:

```
10 SCREEN 2
20 PSET (128,96)
30 FOR C=1 TO 200: NEXT
40 PRESET (128,96)
50 FOR C=1 TO 200: NEXT
60 GOTO 20
```

As linhas 30 e 50 produzem uma pausa para que seja percebido com nitidez o pisca-pisca decorrente do desenha-apaga-desenha do programa.

INSTRUÇÃO LINE - Desenha linnas. Sua estrutura e parâmetros são:

LINE (X,Y)-(X',Y'),C,B ou BF

X e Y determinam as coordenadas da posição inicial da linha. X' e Y' determinam a posição do fim da linha.

As posições indicadas são absolutas em relação à tela.

Se os parâmetros X e Y forem precedidos pela instrução STEP, os mesmos determinarão posições relativas à última posição do cursor após a execução de uma instrução precedente.

C é o parâmetro de cor. Se não houver indicação para C, o computador assumirá a cor vigente para o primeiro plano.

A letra B no fim da instrução faz com que seja desenhado um retângulo, do qual a linha determinada na instrução seria sua diagonal.

A letra F colocada após a letra B determina a pintura do retângulo desenhado.

Exemplos de linhas desenhadas pela instrução LINE:

```
10 COLOR 15,4,11: SCREEN 2
20 LINE (0,96)-(255,96)
30 GOTO 30
```

```
10 COLOR 15,4,11: SCREEN 2
20 LINE (0,0)-(255,191)
30 GOTO 30
```

```
10 COLOR 15,4,11: SCREEN 2
20 LINE (0,191)-(255,0)
30 GOTO 30
```

```
10 COLOR 15,4,11: SCREEN 2
20 LINE (128,0)-(128,191)
30 GOTO 30
```

```
10 COLOR 15,4,11: SCREEN 2
20 LINE (100,80)-(140,120)
30 GOTO 30
```

Exemplos de retângulos com e sem pintura, desenhados por LINE:

```
10 COLOR 15,4,11: SCREEN 2
20 LINE (100,70)-(150,120),,B
30 GOTO 30
```

```
10 COLOR 15,4,11: SCREEN 2
20 LINE (100,70)-(150,120),15,BF
30 GOTO 30
```

Linhas desenhadas com LINE e LINE STEP

```
10 SCREEN 2
20 A=128: B=96
30 C=75: D=C
40 LINE (A,B)-STEP (C,D)
50 LINE (A,B)-STEP (-C,D)
60 LINE STEP (C-D,-D*2)-(A,B)
70 LINE STEP (C,-D)-(A,B)
80 GOTO 80
```

A instrução STEP admite valores positivos e negativos, sendo o deslocamento da posição de LINE executado em conformidade com tais valores em relação à posição do cursor após execução de uma instrução precedente.

INSTRUÇÃO CIRCLE - Desenha circunferências e elipses. Sua estrutura e parâmetros são:

CIRCLE (X,Y),|RAIO|,|COR|,|ANG.INICIAL|,|ANG.FINAL|,|RR|

Se forem omitidos os parâmetros sublinhados, será desenhado um círculo com o raio especificado.

X e Y determinam as coordenadas do centro da circunferência ou da elipse.

RAIO especifica a medida do raio da figura a ser traçada.

COR especifica o número da cor do traçado da figura. Se não for indicada, é assumida a cor vigente para primeiro plano.

ÂNGULO INICIAL e ÂNGULO FINAL são parâmetros expressos em valores radianos e determinam os pontos de início e de fim de um segmento ou arco de circunferência ou de elipse. Uma circunferência tem o ângulo inicial = 0 e o ângulo final = 2*PI. Se os valores determinados para ângulo inicial e ângulo final forem negativos, serão traçados raios dos pontos de início e de fim do arco até seu centro.

RR significa RELAÇÃO DE RAIOS e determina a relação entre raio maior e raio menor de uma elipse a ser desenhada. Se o valor de RR for = 1 será desenhada uma circunferência. RR menor do que 1 desenha uma elipse "achatada" verticalmente. RR maior do 1 produz uma elipse "achatada" horizontalmente.

Desenhos feitos pela instrução CIRCLE:

CIRCUNFERÊNCIA

```
10 COLOR 15,2,9: SCREEN 2
20 CIRCLE (128,96),70
30 GOTO 30
```

ELIPSE VERTICAL

```
10 COLOR 15,5,6: SCREEN 2
20 CIRCLE (128,96),70,15,0,6.28,1.8
30 GOTO 30
```

ELIPSE HORIZONTAL

```
10 COLOR 15,5,6: SCREEN 2
20 CIRCLE (128,96),70,15,0,6.28,1/8
30 GOTO 30
```

CIRCULOS CONCENTRICOS

```
10 COLOR 6,11,13: SCREEN 2
20 FOR C=1 TO 10
30 CIRCLE (128,96),10*C-9,6,,,1
40 NEXT C
50 GOTO 50
```

ARCOS OU SEGMENTOS DE ELIPSES

```
10 COLOR 15,4,8: SCREEN 2
20 FOR C=1 TO 10
30 CIRCLE (128,96),10*C-9,15,0,5,1.5
40 NEXT C
50 GOTO 50
```

SEGMENTO DE CIRCULOS COM RAIOS

```
10 COLOR 6,3,9: SCREEN 2
20 CIRCLE (128,96),70,6,-.01,-5
30 GOTO 30
```

INSTRUÇÃO DRAW - Desenha figuras determinadas pelos comandos da linguagem denominada "macrográfica". A linguagem macrográfica é expressa por "strings" formadas pelos comandos, seguidos das medidas relativas a pontos de imagem da tela de vídeo.

17.3 COMANDOS DA LINGUAGEM "MACROGRÁFICA"

Os comandos da linguagem macrográfica são representados por letras.

Os comandos relacionados a seguir movimentam o cursor, traçando pontos a partir do último ponto de referência. Ponto de referência é a última posição ocupada pelo cursor na execução de uma instrução.

U(n) - PARA CIMA
D(n) - PARA BAIXO
R(n) - PARA DIREITA
L(n) - PARA ESQUERDA
E(n) - PARA CIMA E DIREITA
F(n) - PARA BAIXO E DIREITA

G(n) - PARA BAIXO E ESQUERDA
H(n) - PARA CIMA E ESQUERDA

(n) significa a distância do movimento, correspondendo a n vezes o fator de escala determinado pelo comando S. O comando S será visto pouco mais à frente.

n pode ser representado por um número ou uma variável. Quando representado por um número, é colocado justaposto ao comando. Quando representado por uma variável, esta é separada do comando pelo sinal = (igual), sendo a variável seguida por ; (ponto-e-vírgula). Exemplo: DRAW "U = A;" .

Outros comandos da linguagem macrográfica:

Mx,y - Produz movimentos absolutos ou relativos. Se x e y não tiverem à sua frente sinais + ou - , o movimento será absoluto em relação à tela. Se forem precedidos de tais sinais, o movimento será relativo à última posição ocupada pelo cursor na tela após a execução de um comando precedente.

Todos os comandos já vistos podem ser precedidos pelos comandos seguintes:

B - Movimenta o cursor sem traçar pontos.
N - Movimenta o cursor traçando pontos, mas retorna o cursor ao ponto de origem ou de referência inicial.

Comandos restantes:

S(n) - Fixa o fator de escala para movimentação do cursor, com ou sem traço. Se não for indicado, o computador assume automaticamente o valor 4, que é a escala de distância natural da tela no modo gráfico.

Se for adotado o valor 1, por exemplo, a escala será de 1/4 em relação à de distância natural. Nesse caso, uma linha de 256 pontos será produzida no vídeo com 64 pontos (= 256/4).

A(n) - Determina o ângulo do sistema de coordenadas. n pode variar de 0 a 3, correspodendo respectivamente a 0, 90, 180 e 270 graus. Quando não é especificado, é assumido o valor 0 para n.

C(n) - Determina a cor do traçado. n pode variar de 0 a 15. Se não for especificado, é assumido o valor 15 para n.

X(variável alfanumérica) - Executa os comandos através de uma variável alfanumérica. Sua formatação: DRAW "XA$;", em que A$ poderá ser, por exemplo, "BM128,60F60L120E60", produzindo o traçado de um triângulo no centro da tela. Note que a variável deve ser seguida de ponto-e-vírgula (;).

Desenhos feitos pela instrução DRAW:

DESENHO DOS PONTOS CARDEAIS E COLATERAIS

```
10 A=4: REM Variar A de 1 a 7
20 COLOR 1,5,10: SCREEN 2
30 DRAW "S=A;"
40 DRAW "BM128,96"
50 DRAW "NR50ND50NL50NU50"
60 DRAW "NE35NF35NG35NH35"
70 GOTO 70
```

Os comandos das linhas 30 a 60 podem ser agrupados todos em uma única instrução DRAW. Fizemo-los separados para poder analisar melhor seus efeitos.

Na linha 30, a escala do desenho é determinada pelo valor atribuído à variável A na linha 10. Experimente alterar o valor de A.

Na linha 40, o comando BM posiciona o cursor no centro da tela. Esse será o ponto de referência para os comandos das linhas 50 e 60.

Nas linhas 50 e 60, o comando N, precedendo todos os comandos de movimentos, faz com que o cursor, após executar cada traçado, retorne ao nascedouro, que é o ponto de referência para o início de execução de sua ação.

Experimente eliminar todos os comandos N da linha 50 e rode o programa para perceber melhor como funcionam os outros comandos.

Em seguida, faça o mesmo com a linha 60.

DESENHO DE DEGRAUS EM 4 ESCALAS E 4 ÂNGULOS

```
10 A=0: S=4: COLOR 1,10,5: SCREEN 2
20 DRAW "BM128,96"
30 DRAW "A=A;S=S;"
40 FOR C=0 TO 20 STEP 4
```

```
50 DRAW "U8R8U8R8"
60 NEXT C
70 A=A+1: S=S-1
80 IF A>3 THEN 100
90 GOTO 20
100 GOTO 100
```

Note: na linha 30, os comandos A e S estão em configuração para assumir valores variáveis a cada execução dos quatro ciclos principais do programa, sendo que cada ciclo principal executa outros seis ciclos através do laço FOR/NEXT que encerra.

Na linha 50, a instrução DRAW desenha dois degraus, que se multiplicam por seis em função do laço FOR/NEXT.

Na linha 70, as variáveis A e S têm seus valores alterados para determinação de quatro parâmetros diferentes para A (ângulo) e S (escala) da linha 30.

E interessante notar como o computador memoriza as posições finais do cursor depois de desenhar dois degraus por ciclo, para continuar a execução dos mesmos, em seqüência perfeita, no ciclo seguinte. E isso é feito em quatro ângulos e quatro escalas diferentes.

INSTRUÇÃO PAINT - Pinta uma figura gráfica com uma cor que determina. Sua estrutura e parâmetros:

PAINT (X,Y),|COR|,|COR DA LINHA DE CONTORNO|

X e Y são as coordenadas do ponto de início da pintura. Se forem precedidas pela instrução STEP, o cursor será deslocado, para início da pintura, relativamente à sua última posição após a execução de uma instrução precedente.

COR especifica o número da cor para pintura da figura.

COR DA LINHA DE CONTORNO especifica o número da cor do traçado da figura.

No modo SCREEN 2, COR e COR DA LINHA DE CONTORNO devem ser a mesma.

No modo SCREEN 3, devem ser diferentes.

A seguir, são apresentadas figuras "pintadas" por PAINT.

PINTANDO A TELA TODA

```
10 SCREEN 2
20 PAINT (128,96),10
30 GOTO 30
```

CIRCULO

```
10 SCREEN 2
20 CIRCLE (128,96),70,7
30 PAINT (128,96),7
40 GOTO 40
```

QUADRADO

```
10 SCREEN 2
20 LINE (100,68)-(156,124),10,BF
30 GOTO 30
```

ELIPSE VERTICAL

```
10 SCREEN 2
20 CIRCLE (128,96),70,3,,,1.5
30 PAINT (128,96),3
40 GOTO 40
```

ELIPSE HORIZONTAL

```
10 SCREEN 2
20 CIRCLE (128,96),70,3,,,1/2
30 PAINT (128,96),3
40 GOTO 40
```

TRIANGULO

```
10 COLOR 10,6,4: SCREEN 2
20 DRAW "BM128,40S6F60L120E60"
30 PAINT (128,96)
40 GOTO 40
```

POLIGONO

```
10 COLOR 12,10,4: SCREEN 2
20 DRAW"BM100,30S10R22F15D22G15L22H15U2
2E15"
30 PAINT (101,31)
40 GOTO 40
```

CILINDRO

```
10 SCREEN2
20 CIRCLE (128,50),50,3,0,3.14,1/3
30 CIRCLE (128,140),50,3,3.14,6.28,1/3
40 FOR A=50 TO 140
50 PSET (78,A),3: PSET(178,A),3
60 NEXT A
70 PAINT (128,100),3
80 CIRCLE (128,50),50,12,3.14,6.28,1/3
90 GOTO 90
```

CONE

```
10 COLOR 2,9,4: SCREEN 2
20 LINE (128,30)-(66,140)
30 LINE (128,30)-(190,140)
40 CIRCLE (128,140),60,2,3.14,6.28,1/3
50 PAINT (128,100)
60 CIRCLE (128,100),40,6,3.14,6.28,1/3
70 GOTO 70
```

CUBO

```
10 COLOR 6,3,13: SCREEN 2
20 DRAW "BM60,60"
30 DRAW "E20R100D100G20L100U100"
40 PAINT (100,100)
50 DRAW "BM60,60"
60 COLOR 3
70 DRAW "R100NE20ND100"
80 GOTO 80
```

CIRCULO COM RAIOS

```
10 COLOR 6,15,9: SCREEN 2
20 CIRCLE (128,96),70
30 PAINT (128,96)
40 CIRCLE (128,96),70,15,-.01,-1.05
50 CIRCLE (128,96),70,15,-2.1,-3.15
60 CIRCLE (128,96),70,15,-4.2,-5.25
70 GOTO 70
```

INSTRUÇÃO POINT - Fornece o número de código da cor de determinado ponto de imagem no modo gráfico. Sua estrutura:

POINT (X,Y)

em que X e Y representam as coordenadas do ponto. Exemplo:

```
10 SCREEN 2
20 PSET (118,96),11
30 OPEN "GRP:" AS #1
40 DRAW "BM128,96C15"
50 PRINT #1,"Cor nº";POINT (118,96)
60 GOTO 60
```

17.4 TEXTO NO MODO GRÁFICO

Para se escrever no vídeo no modo gráfico é necessária a utilização dos recursos de arquivo(s) do MSX para vídeo.

São usadas as instruções OPEN e CLOSE, e o posicionamento do cursor para início de impressão do texto é feito através das instruções DRAW "BM X,Y" ou PRESET (X,Y). Exemplo:

RETÂNGULO PINTADO COM ESCRITA DENTRO E FORA DELE

```
10 COLOR 6,11,4: SCREEN 2
20 LINE (60,40)-(200,100),2,BF
30 OPEN "GRP:" AS #1
40 DRAW "BM80,60": COLOR 1
50 PRINT #1, "ESTE é UM"
60 PRINT #1,,,TAB(18)"RETÂNGULO"
70 DRAW "BM80,120"
80 PRINT #1,"OS ÂNGULOS DO"
90 PRINT #1,,,TAB(11)"RETÂNGULO SÃO RET
OS"
100 GOTO 100
```

"Sprites" - figuras 18

SPRITE em inglês significa DUENDE ou FANTASMA, entre outras diversas acepções.

No MSX, SPRITES são figuras que seus fabulosos recursos de programação permitem criar. Figuras que podem assumir os mais variados aspectos e que podem movimentar-se por toda a tela e em profundidades ou camadas diferentes!

Isso mesmo! Os "sprites" podem mover-se em 32 planos diferentes na tela de vídeo. Isso quer dizer, por exemplo, que, se forem colocados na tela 3 ou 4 deles em movimento numa mesma linha, mas em planos diferentes, podem cruzar-se sem envolvimento. Durante o cruzamento, será visto sempre o do plano anterior, passando os outros por trás dele.

Os planos são definidos por prioridades numeradas de 0 a 31. A de número 0 define o plano mais próximo do telespectador. Figuradamente, a de número 31 é a do que fica "la atrás" no tubo de imagem.

Os "sprites" podem ser criados em formato pequeno ou grande.

No formato pequeno são constituídos por 8 x 8 pontos de imagem.

No formato grande são formados por 16 x 16 pontos de imagem.

Há dois tamanhos para o formato pequeno, 0 e 1, e dois tamanhos também para o formato grande, 3 e 4.

Os "sprites" são válidos apenas nos modos de tela SCREEN 1, 2 e 3, sendo seu tamanho definido pelo segundo parâmetro da instrução de controle de tela, SCREEN.

Quando os "sprites" são utilizados em programas que contêm ins-

truções do modo gráfico, só podem ser adotadas as modalidades 2 e 3 de SCREEN.

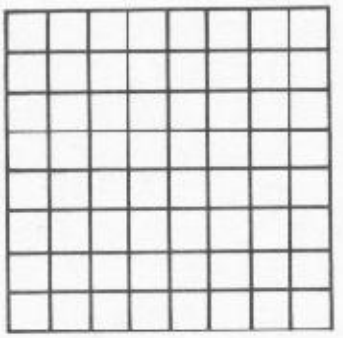
18.1 CRIANDO "SPRITES"

A criação ou formação de "sprites" é feita através da definição em códigos de seus pontos de imagem.

A definição dos pontos pode ser feita por códigos dos três sistemas de números: binário, hexadecimal e decimal e é baseada no "mapa" dos pontos de imagem do "sprite".

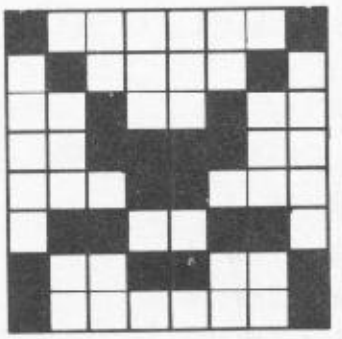
Vamos criar um "sprite" pequeno, começando pelo mapa.

Um mapa de 8 x 8 pontos de imagem corresponde ao seguinte quadriculado:



Cada linha horizontal de um mapa de 8 x 8 pontos corresponde a um "byte". Como cada "byte" tem oito "bits", cada quadradinho corresponde a um "bit".

Se a figura que desejamos criar for um "monstrinho devorador", por exemplo, fazemos seu desenho no mapa:



Nota: deve ter-se em conta que o aspecto da figura feita no papel é muito inferior ao produzido no vídeo pelo MSX.

Cada ponto escurecido no mapa corresponde a um "bit" no estado 1 e cada ponto em branco corresponde a um "bit" no estado 0. Assim, a primeira linha do mapa, que tem o primeiro e o último quadradinhos escurecidos, corresponde a um "byte" com a seguinte codificação binária:

10000001 sendo que a segunda linha corresponde a:
01000010 e assim por diante.

Considerando todas as linhas horizontais do mapa como "bytes" em codificação binária, o "monstrinho devorador" terá os seguintes códigos, que devem ser lidos de cima para baixo:

10000001
01000010
00100100
00111100
00011000
01100110
10011001
10000001

Vamos agora colocar o monstrinho no vídeo. Para tanto, introduzimos no computador o programa montador apresentado a seguir:

18.2 PROGRAMA "MONSTRINHO DEVORADOR"

```
10 COLOR 1,15,4: SCREEN 2,2
20 FOR C=1 TO 8
30 READ R$
40 S$=S$+CHR$(VAL("&B"+R$))
50 NEXT C
60 SPRITE$(0)=S$
100 PUT SPRITE 0, (120,90),1,0
110 GOTO 110
200 DATA 10000001
210 DATA 01000010
220 DATA 00100100
```

```
230 DATA 00111100
240 DATA 00011000
250 DATA 01100110
260 DATA 10011001
270 DATA 10000001
```

Vejamos como funciona o programa, antes de rodá-lo:

18.3 A VARIÁVEL ESPECIAL SPRITE$ E A INSTRUÇÃO PUT SPRITE

Note na linha 60 do programa a variável SPRITE$(N) e na linha 100 a instrução PUT SPRITE P, (X,Y),C,N.

Através da instrução READ RS da linha 30, os códigos binários das linhas de instruções DATA são transformados na "string" de caracteres S$ da linha 40.

Na linha 60, o conteúdo da variável S$ é transferido para a variável SPRITE$(0), que assume a figura do "monstrinho devorador".

A variável SPRITE$, própria do sistema operacional do computador, assume apenas figuras. Pode ter até 256 índices no formato pequeno, que significa que é possível "armazenar" 256 "monstrinhos" ou figuras na memória do computador, a um só tempo. O que acabamos de criar tem apenas o número 0.

No formato de "sprite" grande, a variável SPRITE$ admite 64 índices.

A instrução PUT SPRITE da linha 100 coloca o desenho no vídeo. Seus parâmetros obrigatórios são:

P - Prioridade ou camada do "sprite", podendo variar de 0 a 31.

(X,Y) - Coordenadas de posicionamento do "sprite" na tela, sendo que X pode variar de -32 a 255 e Y de -32 a 191.

C - Cor do "sprite", sendo definida pelo número de código de cor (0 a 15).

N - Número do "sprite", devendo este coincidir com o índice da variável SPRITE$.

Rode agora o programa e verá o "monstrinho" no centro da tela. Note que sua aparência na tela de vídeo é muito melhor do que no mapa.

Mas ele está estático, parece sem vida...

Vamos movimentá-lo! Acrescente as seguintes linhas ao programa montador, que se tornará também movimentador:

```
90 FOR M = 0 TO 255
120 GOTO 90
```

Altere a linha 110 para:

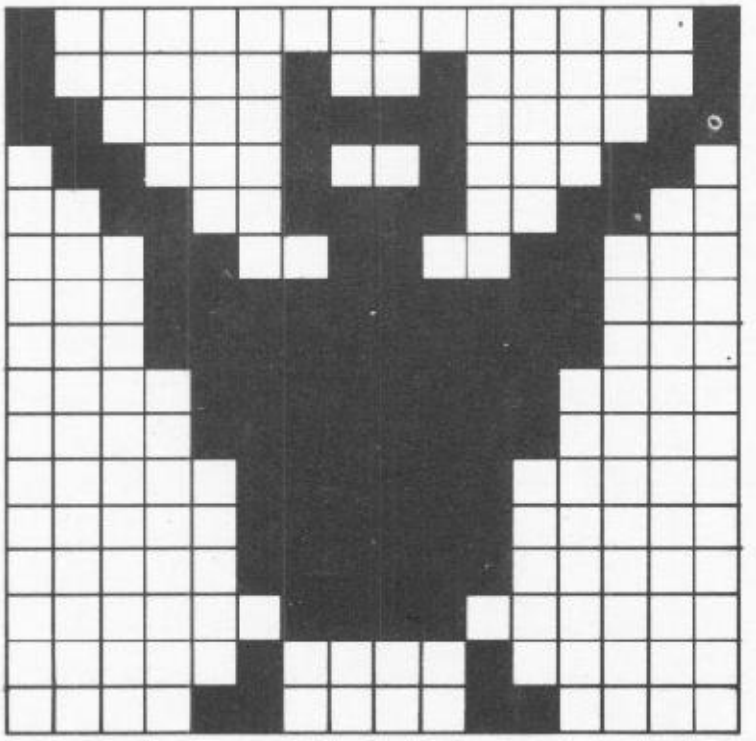
```
110 NEXT M
```

Na linha 100, substitua o número 120 (colocado entre parênteses) por M.

Rode o programa para ver o monstrinho em movimento.

Se quiser vê-lo descendo do alto da tela, inverta as posições de M e 90 na linha 100.

Vamos agora fazer um "sprite" no formato 16 x 16 pontos, começando pelo mapa:



No mapa 16 x 16 pontos, cada linha horizontal corresponde a dois "bytes" e o mapa é considerado em duas alas para efeito de transformar os códigos correspondentes em "strings" de caracteres.

Os códigos da ala esquerda, lidos de cima para baixo, são transformados em caracteres e armazenados numa variável alfanumérica, A$ ou qualquer outra, assim como os da ala direita são armazenados noutra, B$, por exemplo.

Em seguida, os conteúdos de A$ e B$ são transferidos para a variável SPRITE$(N), que passa a ser a figura do mapa. Lembre-se de que N, colocado entre parênteses após a variável, deve ser o número correspondente ao "sprite".

Assim, a figura do mapa será definida pelos seguintes códigos binários:

```
1000000000000001
1000001001000001
1100001111000011
0110001001000110
0011001111001100
0001100110011000
0001111111111000
0001111111111000
0000111111110000
0000111111110000
0000011111100000
0000011111100000
0000011111100000
0000001001000000
0000010000100000
0000110000110000
```

Nota: cada linha codificada acima corresponde a dois "bytes".

O programa montador e movimentador da figura desenhada e codificada será:

18.4 PROGRAMA "MONSTRO ESPACIAL"

```
10 COLOR 1,15,4: SCREEN 2,2
20 FOR C=1 TO 16
30 READ R$
40 A$=A$+CHR$(VAL("&B"+LEFT$(R$,8)))
50 B$=B$+CHR$(VAL("&B"+RIGHT$(R$,8)))
60 NEXT C
70 SPRITE$(0)=A$+B$
80 FOR M=-32 TO 191
90 PUT SPRITE 0, (120,M),1,0
100 NEXT M
110 GOTO 80
200 DATA 1000000000000001
210 DATA 1000001001000001
220 DATA 1100001111000011
230 DATA 0110001001000110
240 DATA 0011001111001100
250 DATA 0001100110011000
260 DATA 0001111111111000
270 DATA 0001111111111000
280 DATA 0000111111110000
290 DATA 0000111111110000
300 DATA 0000011111100000
310 DATA 0000011111100000
320 DATA 0000011111100000
330 DATA 0000001001000000
340 DATA 0000010000100000
350 DATA 0000110000110000
```

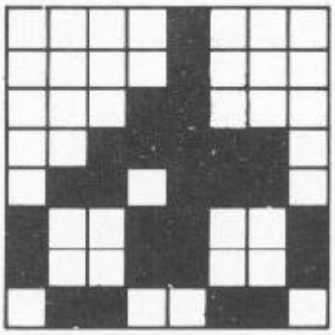
Rode o programa para conferir a aparência do "monstro espacial" com o desenho do mapa. Experimente depois mudar a instrução SCREEN da linha 10 para SCREEN 2,3 e note a diferença.

18.5 "SPRITES" COM CODIFICAÇÃO DECIMAL

Vamos agora criar outro "sprite", usando codificação decimal.

Mas, chega de monstros! Vamos fazer algo diferente.

Comecemos pelo mapa:



Baseados no mapa, anotamos os códigos binários correspondentes a ele e seus equivalentes decimais:

CÓDIGOS BINÁRIOS		CÓDIGOS DECIMAIS
00001000	=	8
00001000	=	8
00011000	=	24
00111110	=	62
01101110	=	110
10011001	=	153
10011001	=	153
01100110	=	102

Para produzir no vídeo o "sprite" correspondente, inserimos no computador o programa que segue (a numeração de linhas do mesmo deverá ser obedecida, para ampliações posteriores):

18.6 PROGRAMA "MOTOCICLETAS"

```
10 COLOR 6,11,7: SCREEN 2,2
60 A$=""
70 FOR C=1 TO 8
80 READ A: A$=A$+CHR$(A)
90 NEXT C
100 SPRITE$(0)=A$
130 FOR M=255 TO -12 STEP -1
140 PUT SPRITE 1,(M,91),1,0
150 NEXT M
160 FOR K=1 TO 500: NEXT K: GOTO 130
170 DATA 8,8,24,62,110,153,153,102
```

Rode o programa e constate como é fácil "produzir" uma motocicleta já com motoqueiro e fazê-la funcionar!

18.7 "SPRITES" E RECURSOS GRÁFICOS

Vamos agora construir uma via para a motocicleta e colocar alguns prédios na paisagem. Detenha a execução do programa e acrescente as seguintes linhas de instruções ao mesmo:

```
30 LINE (0,100)-(255,100)
40 DRAW "BM75,100U50R10D50BM95,100U70R1
5D70"
50 DRAW "BM125,100U30R50D30R20U60R20D60
"
```

Rode novamente o programa e note a compatibilidade entre os recursos gráficos estáticos e os de "sprites".

Observação: deve ser lembrado que, no caso de "sprites" em formato grande com codificação decimal, os 16 códigos correspondentes à ala esquerda do mapa de pontos de imagem, considerados de cima para baixo, devem preceder os correspondentes a ala direita, quando colocados nas linhas de instruções DATA.

18.8 COLISÃO DE "SPRITES" - INSTRUÇÃO ON SPRITE GOSUB

Esse interessante recurso do MSX permite simular colisão entre "sprites" e é largamente utilizado na elaboração de jogos.

Funciona através da instrução

ON SPRITE GOSUB

que desvia a execução do programa para uma sub-rotina quando se encontram num mesmo ponto da tela dois "sprites".

A sub-rotina é utilizada para simular uma explosão, por exemplo, através de recursos gráficos e de som.

A instrução ON SPRITE GOSUB é ativada no programa pela instru-

ção SPRITE ON e desativada por SPRITE OFF. Pode ser desativada apenas temporariamente por SPRITE STOP, a exemplo das outras instruções especiais de desvio, já vistas.

Acrescentando as linhas que seguem ao programa acima, veremos como funciona esse interessante recurso:

```
20 ON SPRITE GOSUB 200
110 PUT SPRITE 0,(50,91),1,0
120 SPRITE ON
200 FOR E=2 TO 12 STEP 2
210 CIRCLE (55,90),E
220 CIRCLE (55,90),E,11
230 NEXT E
240 SPRITE OFF
250 FOR C=90 TO 30 STEP -2
260 PUT SPRITE 0,(50,C),1,0
270 NEXT C
280 PUT SPRITE 0,(50,209),1,0
290 LINE (45,100)-(65,100)
300 RETURN
```

Atenção: alterar a terceira instrução da linha 160 para:

```
GOTO 110
```

COMENTARIOS SOBRE O ACRESCIMO FEITO

LINHA 110: a fim de que ocorra a colisão, uma duplicata do "sprite" é colocada na via principal.

LINHA 120: a instrução SPRITE ON indica ao computador que a linha 20 do programa é "prá valer", isto é, havendo colisão de "sprites", deve ser executada a sub-rotina que começa na linha 200.

LINHAS 200 a 300: simulam um estrondo de colisão através de círculos que se propagam com intermitência.

LINHA 240: a instrução SPRITE OFF desativa a instrução ON SPRITE GOSUB para que o computador não fique executando indefinidamente as linhas 200 a 230. Experimente suprimi-la, a título de experiência.

LINHAS 250 a 270: fazem a moto estacionada ir "pros ares" após a colisão.

LINHA 280: através do parâmetro 209, correspondente à coordenada Y, en-

tre parênteses, a moto trombada é tirada de cena. Experimente colocar um REM nessa instrução e note como a moto fica suspensa no ar depois da colisão.

Depois altere esse parâmetro para 208 e note que as duas motos são tiradas de cena após a colisão. São peculiaridades das instruções relacionadas com os "sprites" e que têm grande validade na elaboração de "videogames".

LINHA 290: Conserta o estrago causado pela colisão no "asfalto" da via principal.

LINHA 300: saída da sub-rotina para continuidade de execução do programa, a partir do ponto em que houve a interrupção.

18.9 CONTROLANDO O MOVIMENTO DOS "SPRITES" - DISPAROS

FUNÇÃO STICK - A função de estrutura STICK(N) tem por finalidade interpretar as direções apontadas pelas teclas de controle do cursor ou pelo "joystick", a fim de movimentar "sprites".

O argumento ou parâmetro N deve ser um número entre 0 e 2, sendo que

0 indica o uso das teclas de controle do cursor,
1 indica o uso de "joystick" ligado ao conector número 1, e
2 indica o uso de "joystick" ligado ao conector número 2.

Quando a alavanca do "joystick" é direcionada ou uma das teclas do cursor é pressionada, a função STICK assume um valor entre 1 e 8, que serve de parâmetro para execução de uma instrução IF... THEN.

Os valores obtidos por STICK baseiam-se no seguinte esquema de direções apontadas pelas teclas do cursor ou pelo "joystick".

FUNÇÃO STRIG - A finalidade da função de estrutura STRIG(N) é verificar se é pressionada a barra espaçadora do teclado ou se é apertado o botão de disparo de um "joystick" conectado ao microcomputador, a fim de produzir efeitos de "disparos" ou similares.

O argumento N deve ser um número entre 0 e 4, sendo que:

0 se relaciona com a barra espaçadora,
1 ou 3 ao botão disparador do "joystick" 1, e
2 ou 4 ao botão disparador do "joystick" 2.

Se for pressionado um botão disparador ou a barra espaçadora, a função STRIG assume o valor -1. Caso contrário, assume o valor 0.

O valor assumido por STRIG serve de parâmetro para execução da instrução comentada a seguir:

INSTRUÇÃO ON STRIG GOSUB N - Indica ao computador o número N de linha de sub-rotina para a qual deve ser desviado o fluxo de execução do programa quando é pressionada a barra espaçadora do teclado ou o botão disparador de um "joystick".

STRIG ON, OFF, STOP

STRIG(N) ON ativa a instrução ON STRIG GOSUB de um programa.
STRIG(N) OFF desativa totalmente a instrução ON STRIG GOSUB.
STRIG(N) STOP desativa temporariamente a instrução em foco.

Exemplos de utilização das instruções e funções mencionadas serão vistos no programa que será apresentado a seguir. Nele, a utilização dos comandos e instruções relacionados com "sprites" é conjugada com:

- recursos de movimentação;
- contagem de tempo;
- desenho e pintura de cenários,

redundando num "videogame" que, além da finalidade didática a que se propõe, servindo de base inicial para estudo e elaboração de programas do gênero, poderá servir de entretenimento.

O jogo tem como enredo o propósito de uma nave extraterrestre - NAVET - destruir com seus poderosos raios "laser" dois lindos prédios de uma cidade.

NAVET dispõe de tempo exíguo para sua missão e ainda sofre com-

bate de três valentes helicópteros que procuram rechaçá-la às alturas.

Use as teclas do cursor para movimentar NAVET e a barra espaçadora para o disparo de raios "laser". Estes só têm efeito sobre os prédios da cidade. Os helicópteros são imunes a eles.

18.10 PROGRAMA "ATAQUE DE NAVET"

```
10 DATA159,112,96,255,255,132,132,255,2
55,113,49,31,31,48,96,192
20 DATA249,14,6,255,255,33,33,255,255,1
42,140,248,248,12,6,3
30 DATA0,255,16,121,255,32,112,0,0,255,
8,156,255,62,4,14
40 COLOR6,11,7:SCREEN2,2
50 A$=""
60 FORC=1TO32
70 READA:A$=A$+CHR$(A)
80 NEXTC
90 SPRITE$(0)=A$
100 FORK=1TO2
110 A$="":A=0
120 FORC=1TO8
130 READA:A$=A$+CHR$(A)
140 NEXTC
150 SPRITE$(K)=A$
160 NEXTK
170 ON SPRITE GOSUB 600
180 ON STRIG GOSUB 520
190 DRAW"BM5,192C4U40R15D35R10U45R20D50
":PAINT(10,160),4
200 DRAW"BM65,192C15U75R15D75":PAINT(70
,160),15
210 FORI=125TO185STEP5
220 PSET(70,I):PSET(75,I)
230 NEXTI
240 DRAW"BM90,192C9U70R20D65R15U55R15D6
0":PAINT(100,160),9
250 DRAW"BM145,192C15U80R25D80":PAINT(1
60,160),15
260 FORI=120TO185STEP5
```

```
270 PSET(150,I):PSET(155,I):PSET(160,I)
:PSET(165,I)
280 NEXTI
290 DRAW"BM185,192C13U65R20D60R15U50R20
D50R10U40R10":PAINT(190,160),13
300 SPRITE ON
310 D=255:E=0:F=5
320 TIME=0
330 STRIG(0) ON
340 PUTSPRITE 1,(X,Y),12,0
350 PUTSPRITE2,(D,M),1,1
360 PUTSPRITE3,(E,N),6,2
370 'PUTSPRITE4,(D+25,0),12,1
380 PUTSPRITE5,(E-32,P),4,2
390 IFX<10 THEN X=10
400 IFX>230 THEN X=230
410 IFY<5 THEN Y=5
420 IFY>150 THEN Y=150
430 IF STICK(0)=1 THEN Y=Y-2
440 IF STICK(0)=5 THEN Y=Y+2
450 IF STICK(0)=3 THEN X=X+2
460 IF STICK(0)=7 THEN X=X-2
470 IFSTRIG(0)THENPLAY"C4L4"
480 D=D-F:E=E+F
490 IF D=0 THEN SWAP D,E:GOSUB580
500 IF TIME>1800 THEN GOSUB 610
510 GOTO 330
520 STRIG(0)OFF
530 FOR I=1TO3
540 LINE(X+7,Y+14)-(X+7,Y+90)
550 NEXT I
560 LINE(X+5,Y+14)-(X+9,Y+90),11,BF
570 RETURN
580 M=INT(RND(-TIME)*160)
590 N=M/2:O=M/3:P=M/4:RETURN
600 X=0:Y=0:RETURN
610 OPEN"GRP:" AS # 1
620 PRESET(6,90):COLOR1:PRINT#1,"Tempo
esgotado. Para jogar nova- mente pressi
one a tecla F5, ok?"
630 FORA=1TO2500:NEXT:END
```

Som musical e efeitos sonoros 19

Vimos que, além dos atributos inerentes a um microcomputador de categoria, o do padrão MSX conta com outros recursos de programação e de desenho gráfico e animado.

Podemos dizer agora, sem receio de exagero, que, além de microcomputador, o MSX é um instrumento musical.

Dotado de poderoso gerador sonoro de três canais, o MSX possibilita ao apreciador produzir e tocar música sofisticada e com características próprias de instrumentos diversos.

Também os mais variados efeitos sonoros e ruídos podem ser produzidos pelo seu gerador de som, de forma a enriquecer jogos e programas.

19.1 MÚSICA

INSTRUÇÃO PLAY - É utilizada para produzir som musical ou música. A instrução PLAY opera os comandos da LINGUAGEM MACROMUSICAL. Esta é constituída pelas letras do alfabeto enunciadas na tabela apresentada a seguir com seus respectivos significados:

19.2 COMANDOS DA LINGUAGEM MACROMUSICAL

C = nota musical DO
D = nota musical RE
E = nota musical MI
F = nota musical FA
G = nota musical SOL
A = nota musical LA
B = nota musical SI
O = oitava musical (1 a 8. Padrão: 4)
L = comprimento ou duração da nota (1 a 64. Padrão: 4)
V = volume (0 a 15. Padrão: 8)
R = pausa (1 a 64)
M = freqüência de modulação de envoltória (1 a 65535. Padrão: 255)
S - forma de envoltória (1 a 15. Padrão: 1)
T = tempo de execução em quartos de nota por minuto (32 a 255. Padrão: 120)
N = numeração absoluta da nota em 8 oitavas (0 a 95)
X = variável usada em lugar do comando.

Observação: os valores indicados como padrões são os assumidos automaticamente pelo computador ao ser ligado.

SINAIS USADOS APÓS OS COMANDOS E SIGNIFICADO

. (ponto) = duração da nota aumentada em 50%.
+ ou # (mais ou cerquilha) = nota em sustenido.
- (menos) = nota em bemol.

A estrutura da instrução PLAY é: PLAY "COMANDOS"

em que "comandos" são as letras da linguagem macromusical. Exemplos:

```
PLAY "O4CDEFGABO5C"  toca a 4ª oitava da escala musical.
PLAY "O5CDEFGABO6CDEFGABO7C"  toca a 5ª e 6ª oitavas.
PLAY "O4C+D+F+G+A+"  toca os sustenidos da 4ª oitava.
PLAY "O4D-E-G-A-B-"  toca os bemóis da 4ª oitava.
```

Na forma desses exemplos, a instrução PLAY gera som apenas através de um canal.

Para tocar nos três canais, devem ser feitos três comandos - separados por vírgulas - por uma mesma instrução PLAY:

```
PLAY "O4DDF","O5EFG","O6CEF"
```

Para tocar acordes:

```
PLAY "D","F","E"
```

Quando os parâmetros referentes aos comandos O, L, V, M, S e T não forem especificados, são automaticamente assumidos pelo GPS (Gerador Programável de Som) os valores padrões respectivos. Depois de especificados uma vez, todavia, permanecem vigentes enquanto não forem alterados ou não for o computador desligado.

Conclui-se ser necessário indicá-los a cada passo, em conformidade com os efeitos ou resultados desejados.

Os comandos O, L e V são normalmente especificados para uma frase musical inteira ou para cada nota específica, em conformidade com a peculiaridade e características da música. Exemplos:

```
10 PLAY "O3L8V12CDEFGABB.A+D-EDC"
20 PLAY "V12O3L8CDEO4L4FGV12O5L2AB"
```

Em vez de Ln, a duração da nota pode também ser indicada apenas por um número colocado após ela. Nesse caso, a frase da linha 20, ficaria na seguinte forma:

```
20 PLAY "V12O3C8D8E8O4F4G4V12O5A2B2"
```

TABELA DE DURAÇÃO DAS NOTAS

L1 = semibreve
L2 = mínima
L4 = semínima
L8 = colcheia
L16 = semicolcheia
L32 = fusa
L64 = semifusa

Um . (ponto) colocado após uma nota aumenta sua duração ou comprimento em 50%. Dois pontos (..) aumentam a duração da nota para 2,25 vezes seu valor original. Três ... para 3,375 vezes. Perceba no programa que segue:

```
10 PLAY "V1204L2C"
20 PLAY "R64"
30 PLAY "V1204L2C..."
40 PLAY "V1204L8FGAB"
50 PLAY "R64"
60 PLAY "V1204L8F...G...A...B..."
```

Os sinais de sustenidos (+ ou #) e bemóis (-) devem sempre ser usados após as letras das notas correspondentes. Exemplo:

```
PLAY "O4V10CC+DD-E-FGG+G-AB"
```

Os comandos M, S e T são geralmente usados para uma música toda ou para períodos da mesma, sendo especificados no início de cada. O comando S determina a forma de modulação das notas, enquanto que M determina a freqüência de modulação. Note sua atuação no exemplo que segue:

```
10 BEEP
20 PLAY "V12L4CDEFGAB"
30 PLAY "L4S5M500CDEFGAB"
40 PLAY "T255L3S8M3000CDEFGAB"
```

Os comandos M e S não devem ser usados simultaneamente com o comando V.

O comando T determina a velocidade de execução da música; pode variar de 32 a 255, sendo a maior velocidade obtida com o valor 255.

O comando BEEP é usado para "resetar" os valores do GPS. No pequeno programa acima, por exemplo, se não for usado, ao ser repetida a execução do programa a linha 20 assumirá os parâmetros dos comandos T e L da linha 40, executada por último.

O comando R é usado para determinar pausas entre notas ou períodos. Seus parâmetros de duração são os mesmos das notas. Exemplo:

```
PLAY "V10L4CDEFGABR64CDEFGABR64...CDEFGAB"
```

O comando N é usado para especificar diretamente as 96 notas musicais (8 oitavas), substituindo os comandos O e C, D, E, F, G, A e B. Exemplo:

```
PLAY "V12L8N36N38N40N41N43N45N47"  corresponde a:
PLAY "V12L8O4CDEFGAB"
```

19.3 O USO DE VARIÁVEIS COM COMANDOS

Variáveis podem ser usadas para especificar os parâmetros dos comandos da linguagem macromusical. Exemplo:

```
10 X = 10: Y = 8
20 PLAY "V=X;L=Y;CDEF"    - em lugar de:
30 PLAY "V10L8CDEF"
```

É importante observar que as variáveis devem sempre ser colocadas entre o sinal de igualdade (=) e o de ponto-e-vírgula (;).

19.4 MODOS DE TOCAR MÚSICA NO MSX

É possível tocar música diretamente através do teclado ou indiretamente, através de programas.

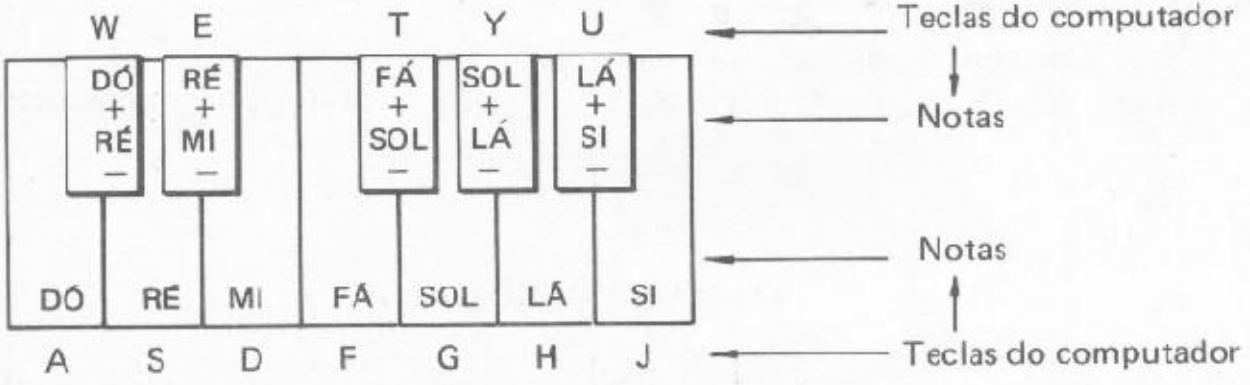
Para tocar diretamente por meio do teclado, digite o programa apresentado a seguir:

19.5 PROGRAMA "MÚSICA DIRETA PELO TECLADO'

```
10 SCREEN,,0
20 L=8: S=1: M=5000
30 PLAY "L=L;S=S;M=M;"
40 R$=INPUT$(1)
50 IF R$="A" THEN PLAY "C"
60 IF R$="W" THEN PLAY "C+"
70 IF R$="S" THEN PLAY" D"
80 IF R$="E" THEN PLAY "D+"
90 IF R$="D" THEN PLAY "E"
```

```
100 IF R$="F" THEN PLAY "F"
110 IF R$="T" THEN PLAY "F+"
120 IF R$="G" THEN PLAY "G"
130 IF R$="Y" THEN PLAY "G+"
140 IF R$="H" THEN PLAY "A"
150 IF R$="U" THEN PLAY "A+"
160 IF R$="J" THEN PLAY "B"
170 IF R$="1" THEN PLAY "01"
180 IF R$="2" THEN PLAY "02"
190 IF R$="3" THEN PLAY "03"
200 IF R$="4" THEN PLAY "04"
210 IF R$="5" THEN PLAY "05"
220 IF R$="6" THEN PLAY "06"
230 IF R$="7" THEN PLAY "07"
240 IF R$="8" THEN PLAY "08"
250 GOTO 30
```

Depois de digitado e conferido o programa, tecle F5 e experimente tocar algumas músicas usando o teclado de acordo com a figura apresentada a seguir:



Ao ser executado o programa, a escala vigente é a quarta (quarta oitava), ASSUMIDA AUTOMATICAMENTE PELO COMPUTADOR, conforme já explicado.

Para mudar para outra escala (ou oitava), ENQUANTO TOCA MÚSICA, aperte uma das teclas de números 1 a 8.

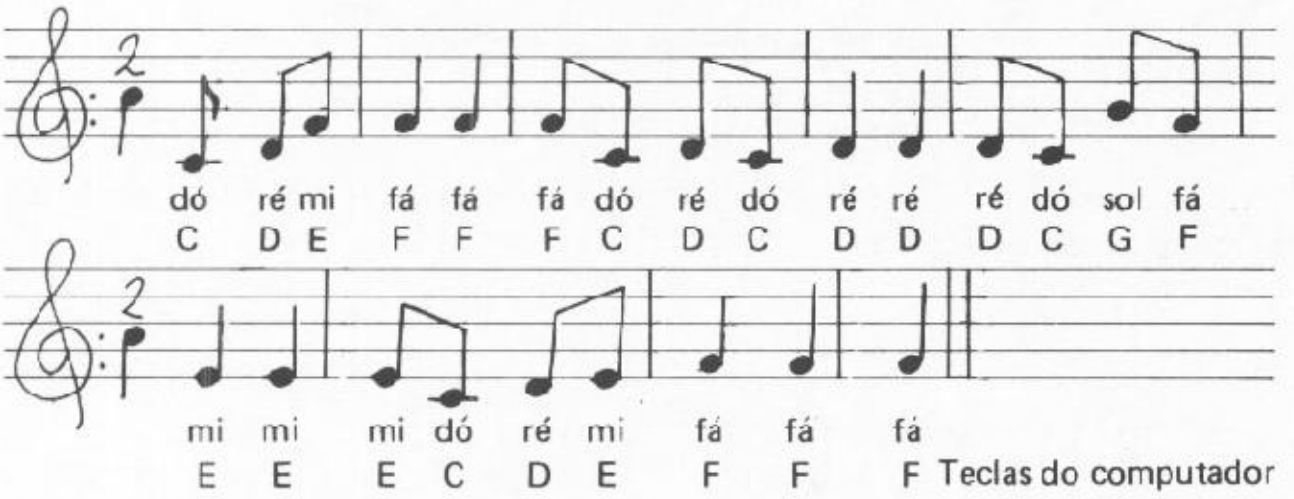
A duração das notas, a forma de onda da envoltória e a freqüência desta estão especificadas nas linhas 20 e 30 do programa, com valores de 8, 1 e 5.000 respectivamente.

Experimente outros valores para L, S e M, de acordo com a tabela de comandos apresentada no início deste capítulo. Para tanto, bastará alterar os valores da variáveis respectivas na linha 20.

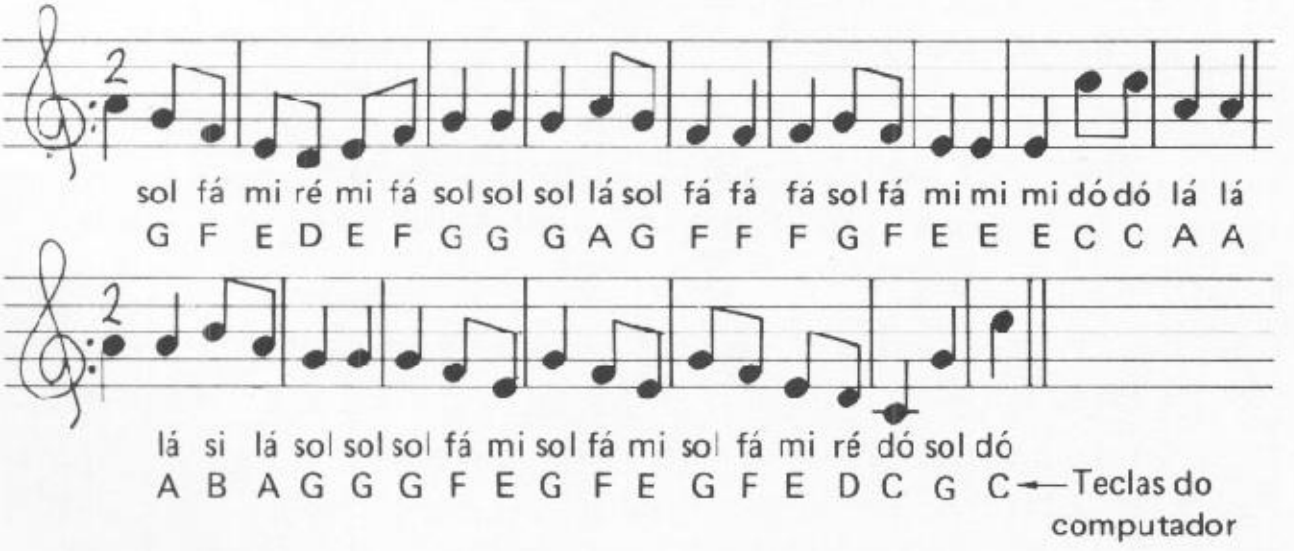
Naturalmente, será possível ampliar os recursos do programa, para permitir, por exemplo, tocar simultaneamente em duas ou três oitavas e com acompanhamento. Deixamos a cargo do leitor essa tarefa, que se constituirá em excelente exercício.

Na falta de lembrança de músicas mais apuradas, experimente tocar as duas popularíssimas que seguem:

DÓ RÉ MI FÁ



ATIREI O PAU NO GATO



19.6 MÚSICA ATRAVÉS DE PROGRAMAS

O MSX pode tocar também sozinho, bastando para tanto escrever em forma de programa a música que se deseja que ele toque.

Fazendo uso da instrução PLAY e dos comandos da linguagem macro musical examinados no início do capítulo, torna-se muito fácil transformar o MSX em invejável virtuoso.

A título de exemplo e a fim de que possa ser constatada a facilidade de conversão, as duas pequenas músicas transcritas são apresentadas a seguir na forma de programas:

19.7 PROGRAMA "DÓ-RÉ-MI-FÁ"

```
10 PLAY "O4S1M5000"
20 PLAY "L8CDEL4FF"
30 PLAY "L8FCDCL4DD"
40 PLAY "L8DCGFL4EE"
50 PLAY "L8ECDEL4FFF"
```

Se desejar que a música seja tocada concomitantemente em dois canais de som do MSX, repita os comandos de cada linha na própria linha, separando-os com vírgula. Mude apenas as oitavas, para poder distinguir os sons dos dois canais. Exemplo:

```
10 PLAY "S1M5000", "S1M5000"
20 PLAY "O3L8CDEL4FF", "O5L8CDEL4FF"
...
```

A título de exercício, use o terceiro canal para produzir acompanhamento:

```
10 PLAY "S1M5000", "S1M5000", "S1M5000"
20 PLAY "O5L8CDEL4FF", "O7L8CDEL4FF", "
R1603L4CR16F"
...
```

Não se esqueça de ajustar o volume do televisor adequadamente, para uma boa reprodução do som produzido.

19.7 PROGRAMA "ATIREI O PAU NO GATO"

```
10 PLAY "O4S1M5000"
20 PLAY "L8GFEDEF"
30 PLAY "L4GGGL8AG"
40 PLAY "L4FFFL8GF"
50 PLAY "L4EEEL8CC"
60 PLAY "L4AAAL8BA"
70 PLAY "L4GGGL8FE"
80 PLAY "L4GL8FEGFED"
90 PLAY "L4CG05C"
```

19.8 EFEITOS SONOROS

Efeitos sonoros, incluindo ruídos, são produzidos atuando-se diretamente sobre 14 dos 16 registros do GPS, o que é feito através da

INSTRUÇÃO SOUND, cuja estrutura é SOUND N, V

em que N é o número de registro e V o valor a atribuir ao mesmo.

A tabela apresentada a seguir indica os registros do GPS, os valores admitidos pelos mesmos e suas funções:

19.9 TABELA DOS REGISTROS DO GPS

REGISTRO	VALORES	FUNÇÃO	OBSERVAÇÃO
0	0 a 255	Determinar freqüência de som do canal A	Ajuste fino
1	0 a 15		Ajuste grosso
2	0 a 255	Determinar freqüência de som do canal B	Ajuste fino
3	0 a 15		Ajuste grosso
4	0 a 255	Determinar freqüência de som do canal C	Ajuste fino
5	0 a 15		Ajuste grosso
6	0 a 31	Determinar freqüência de ruído	
7	1 a 63	Abrir canais para saída de som/ruído	
8	0 a 15/16	Controlar volume canal A	Para uso de envoltória, deve ser adotado o parâmetro 16
9	0 a 15/16	Controlar volume canal B	
10	0 a 15/16	Controlar volume canal C	
11	0 a 255	Determinar freqüência da envoltória	
12	0 a 255		
13	0 a 15	Determinar forma de envoltória	

Para determinar a freqüência de som a ser gerado pelo canal A, por exemplo, divide-se 124.797 pela freqüência desejada:

Exemplo de freqüência a ser gerada: 300

124.797/300 = 416. Divide-se este resultado por 256:

416/256 = 1 com resto = 160.

1 vai para o registro de ajuste grosso do canal escolhido e 160 para o registro de ajuste fino do mesmo canal.

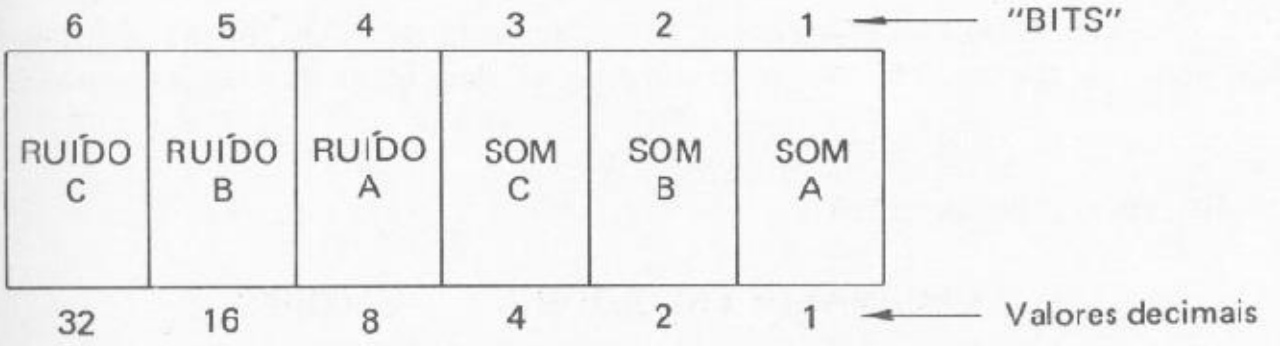
O procedimento é o mesmo para os três canais.

A freqüência máxima que pode ser gerada por um canal é 124.797 ciclos. Todavia, o ouvido humano pode captar (escutar) 15.000 ciclos no máximo.

A freqüência para o canal de ruído também é determinada dividindo-se 124.797 pela freqüência desejada, sendo o quociente atribuído ao registro 6. Todavia, como o valor máximo admissível pelo registro é = 31, conclui-se que a freqüência mínima de ruído que pode ser gerada é de 4025 ciclos.

O registro 7 controla a saída de som e/ou ruído para os canais, através de seus "bits" ativos, que são três para som e três para ruídos, correspondentes aos canais A, B e C em cada modalidade. "Bits" no estado 0 abrem a saída dos canais. "Bits" no estado 1 fecham.

A figura apresentada a seguir facilita a compreensão e o controle do registro 7:



Para determinar saída de SOM pelos canais A, B e C, a título de exemplo, devem ser fechados os demais canais, o que é feito atribuindo-se a soma dos valores de seus "bits" no estado 1 ao registro 7 atra-

vés da instrução SOUND.

Assim, fechamos os canais A, B e C de RUIDO através do seguinte comando:

```
SOUND 7,56
```

pois seus "bits" no estado 1 correspondem a 32 + 16 + 8 = 56, ou 63 - 7 = 56, sendo 7 a soma dos valores dos "bits" dos canais de som no estado 0. 63 é o valor máximo para o registro 7, de acordo com a tabela.

Para determinar a saída de RUIDO pelo canal B, por exemplo, comandamos:

```
SOUND 7,47
```

pois os "bits" dos demais canais no estado 1 correspondem a 32 + 8 + 4 + 2 + 1, ou, inversamente, 63 - 16.

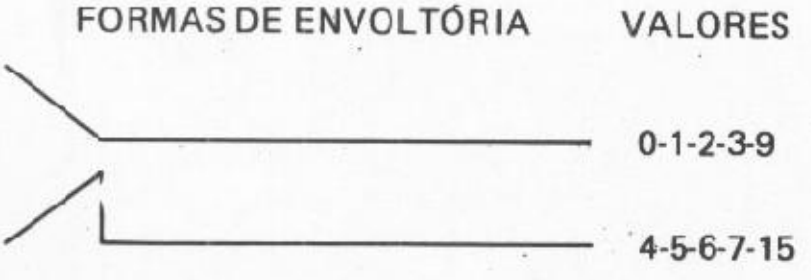
Os registros 8, 9 e 10 controlam o volume de saída de som ou de ruído pelos canais A, B e C. Seus valores variam de 0 a 15 quando controlam apenas volume e não são usados os registros 11, 12 e 13.

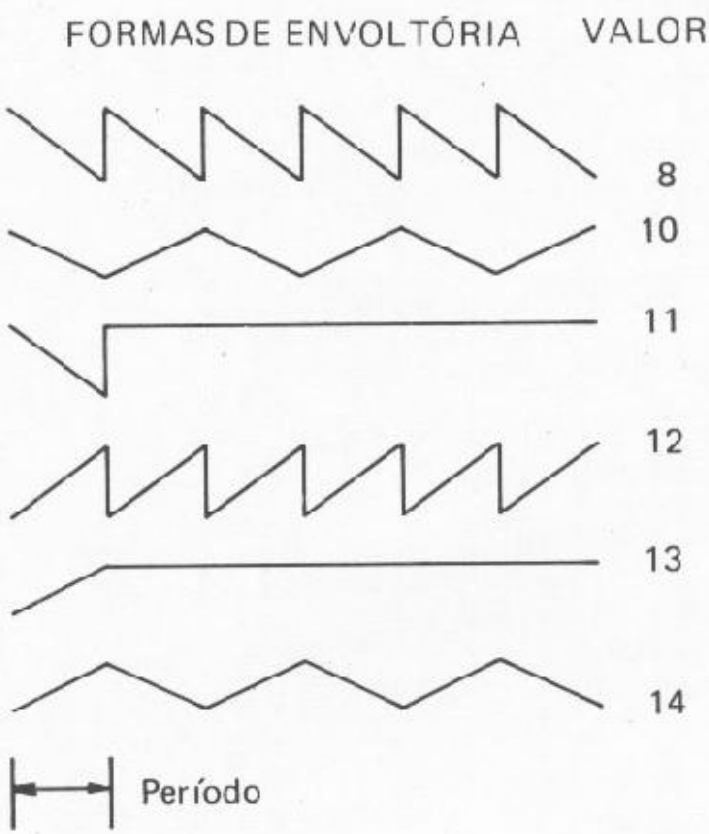
Se forem usados os registros 11, 12 e 13, o canal que for usado em conexão com os mesmos teve ter seu registro de volume fixado com o valor 16. Os três canais podem ser usados em conexão com os registros 11, 12 e 13.

Os registros 11 e 12 determinam a freqüência de envoltória. Para determinar os valores para os registros, divida 7799,8 pela freqüência desejada.

O registro 13 determina a forma de onda da envoltória. Seu valor pode variar de 0 a 15, determinando as seguintes formas de ondas:

19.10 FORMAS DE ENVOLTÓRIA





Já sabemos como determinar os valores a serem atribuídos aos registros do GPS. Façamos agora algumas experiências:

GERAR SOM DE 600 HZ OU CICLOS PELO CANAL A:

- 124.797/600 = 208;
- 208 deverá ser o valor do registro 0;
- abrir canal A para saída de som: 63 - 1 = 62;
- 62 deverá ser o valor do registro 7;
- vamos determinar volume = 12;
- 12 deverá ser o valor do registro 8.

PROGRAMA CORRESPONDENTE:

```
10 SOUND 0, 208
20 SOUND 7, 62
30 SOUND 8, 12
```

GERAR RUIDO DE 5000 HZ PELO CANAL A:

- 124.797/5000 = 20,8
- 21 deverá ser o valor do registro 6;
- abrir canal A para saída de ruído: 63 - 8 = 55;
- 55 deverá ser o valor do registro 7;
- vamos determinar volume = 10;
- 10 deverá ser o valor do registro 8.

PROGRAMA CORRESPONDENTE:

```
10 SOUND 6, 21
20 SOUND 7, 55
30 SOUND 8, 10
```

Vejamos agora como uma mesma freqüência gera SOM e RUÍDO:

19.11 PROGRAMA "SOM E RUÍDO"

```
10 SOUND 0,25
20 SOUND 7,62
30 SOUND 8,12
40 FOR C=1 TO 1000: NEXT: BEEP
50 SOUND 6,25
60 SOUND 7,55
70 SOUND 8,12
80 FOR C=1 TO 1000: NEXT: BEEP
90 GOTO 10
```

COMANDO BEEP - Produz um som tipo "bip". Quando usado depois da instrução SOUND, recompõe o GPS no estado 0, isto é, "limpa" a memória do gerador programável de som. O comando BEEP / RETURN equivale a PRINT CHR$ (7) / RETURN.

19.12 EXEMPLOS DE EFEITOS SONOROS

Os programas apresentados a seguir utilizam os recursos expostos e comentados a partir do item 19.8.

PARTIDA DE TREM A VAPOR

```
10 CLS
20 SOUND 0,255
30 SOUND 7,8
40 SOUND 8,15
50 FOR C=1 TO 600: NEXT C
60 BEEP
70 M=340: V=14
80 SOUND 7,7
90 SOUND 8,V: SOUND 9,V-3
100 FOR C=1 TO 120: NEXT C: SOUND 8,0
110 FOR C=1 TO M: NEXT C
120 M=M-5: IF M<10 THEN M=5
130 IF M<200 THEN V=V-.1
140 IF V<5 THEN 160
150 GOTO 80
160 BEEP: PRINT"O trem partiu..."
```

TRENS PASSANDO

```
10 SOUND 6,30
20 SOUND 7,7
30 SOUND 8,16
40 SOUND 11,50
50 SOUND 12,200
60 SOUND 13,14
70 TIME=0
80 PRINT "Ouça os trens que vão passar.
.."
90 IF TIME<1800 THEN 90
100 BEEP: PRINT,,"Já passaram..."
```

TIRO OU EXPLOSÃO

```
10 SOUND 1,10
20 SOUND 6,20
30 SOUND 7,7
40 SOUND 8,16
50 SOUND 12,45
60 SOUND 13,0
```

VEÍCULO PASSANDO

```
10 V=5: E=V: CLS
20 SOUND 1,10
30 SOUND 7,0
40 SOUND 8,V
50 IF E=15 THEN 90
60 FOR C=1 TO 250: NEXT C
70 V=V+.5: E=E+.5
80 GOTO 20
90 FOR C=1 TO 250: NEXT C
100 V=V-.5: IF V=3 THEN 120
110 GOTO 20
120 BEEP: PRINT "O veículo passou..."
```

Manipulação de vídeo no MSX 20

Já comentamos que o MSX dispõe de um processador próprio de vídeo. Este processador tem memória auxiliar independente de 16 "kbytes" e é acessado diretamente pela UCP (unidade central de processamento), através do BASIC MSX.

A memória própria do processador de vídeo é denominada VRAM. Denominaremos o processador de vídeo PTV (processador de tela de vídeo), como tradução de VDP (video display processor).

O PTV pode ser controlado diretamente pelo usuário, através de VPOKE, VPEEK, VDP e BASE e com auxílio das TABELAS DE CONTROLE DO PTV.

INSTRUÇÃO VPOKE

Insere códigos diretamente na VRAM. Sua sintaxe é:

VPOKE |endereço|,|código|

sendo que o endereço varia de 0 a 16383 e o código de 0 a 255.

FUNÇÃO VPEEK

Fornece os códigos (em números decimais) contidos na VRAM. Sua sintaxe é:

VPEEK (endereço)

sendo endereço um número compreendido entre 0 e 16383, correspondente aos "bytes" de endereçamento do PTV.

VARIÁVEL VDP

Armazena o conteúdo dos registros do PTV e é responsável pela definição de características da tela de vídeo nas suas diversas modalidades, incluindo molduras, coloração, tamanho de "sprites" etc. Sua sintaxe é:

VDP (n)

sendo n um número de 0 a 8, correspondente aos registros do PTV.

Com exceção do registro 8, os conteúdos (ou códigos) dos registros podem ser alterados. Para alterá-los, o comando deve ser:

VDP (n) = |número|

sendo n o número de registro e número um valor entre 0 e 255, que será atribuído ao mesmo.

Observação: em razão de que cada "bit" dos registros do PTV é significativo no processamento de vídeo, a manipulação da variável VDP é bastante complexa e recomendável apenas quando o usuário possui conhecimentos bem avançados sobre o computador de forma geral e, em particular, sobre o PTV.

VARIÁVEL BASE

Fornece as posições iniciais, na VRAM, em que se encontram as tabelas que contêm os códigos dos caracteres e dos símbolos que são exibidos no vídeo, bem como de seus padrões e das cores, em cada modalidade de tela em que são geradas as imagens correspondentes à execução de comandos ou de programas. Sua sintaxe é:

BASE (n)

sendo n o número correspondente a uma das tabelas, de acordo com a lista apresentada abaixo.

Os endereços das tabelas podem ser alterados pelo usuário. O comando deve ser:

BASE (n) = |endereço na VRAM|

sendo n o número da tabela e endereço o novo endereço a ser inserido.

TABELAS DE CONTROLE DO PTV

MODO TEXTO 1 (SCREEN 0)

(0) 0-959 - TABELA DE CÓDIGOS DOS CARACTERES EM TELA
(2) 2048-4095 - TABELA DE FORMAÇÃO DOS CARACTERES

MODO TEXTO 2 (SCREEN 1)

(5) 6144-6911 - TABELA DE CÓDIGOS DOS CARACTERES EM TELA
(6) 8192-8223 - TABELA DE CORES DE CADA 8 CARACTERES
(7) 0-2047 - TABELA DE FORMAÇÃO DOS CARACTERES
(8) 6912-7040 - TABELA DE ATRIBUTOS DE "SPRITES"
(9) 14336-16383 - TABELA DE FORMAÇÃO DE "SPRITES"

MODO GRÁFICO DE ALTA RESOLUÇÃO (SCREEN 2)

(10) 6144-6911 - TABELA DE CÓDIGOS DOS CARACTERES EM TELA
(11) 8192-14335 - TABELA DE CORES PARA CADA 8 PONTOS
(12) 0-6143 - TABELAS (3) DE FORMAÇÃO DE CARACTERES
(13) 6912-7040 - TABELA DE ATRIBUTOS DE "SPRITES"
(14) 14336-16383 - TABELA DE FORMAÇÃO DE "SPRITES"

MODO GRÁFICO DE BAIXA RESOLUÇÃO (SCREEN 3)

(15) 2048-3583 - TABELA DE CÓDIGOS DOS CARACTERES EM TELA
(17) 0-2047 - TABELA DE FORMAÇÃO DOS CARACTERES
(18) 6912-7040 - TABELA DE ATRIBUTOS DE "SPRITES"
(19) 14336-16383 - TABELA DE FORMAÇÃO DE "SPRITES"

Observações: Os números das tabelas estão colocados entre parênteses.

Os endereços ocupados pelas tabelas na VRAM estão transcritos em seguida aos números das mesmas.

Convém observar que a manipulação das tabelas apresentadas, como também do PTV, diretamente, isto é, não através de programação normal pelo BASIC MSX, requer do usuário muita experiência e conhecimentos de recursos bastante avançados no trato com o computador, não sendo recomendável fora dessa condição.

Infelizmente, os manuais que acompanham os MSX nacionais não facilitam essa tarefa. Um dos manuais não contém sequer os dados básicos indispensáveis sobre a matéria.

Por outro lado, inexiste ainda literatura específica em língua portuguesa, de sorte que resta ao usuário, caso deseje explorar toda a potencialidade do PTV em manipulação direta, fazer incursões gradativas no mesmo e ir acumulando experiência até que todas as informações necessárias sejam obtidas.

Antes de fazê-lo, contudo, convém refletir sobre ser ou não conveniente, já que o BASIC MSX oferece condições sobejas para manipular o PTV, embora indiretamente, de maneira a obter do mesmo os mais variados efeitos visuais, alguns dos quais tivemos oportunidade de notar em execuções de programas apresentados neste livro.

Entretanto, visando facilitar ao usuário sua introdução na aquisição da prática indispensável para lidar com o PTV, apresentamos a seguir um programa que, utilizando os principais recursos discutidos, sugere meios de abordar a matéria.

20.1 PROGRAMA "CONHECENDO E REDEFININDO CARACTERES"

```
10 CLS: KEYOFF: PRINT: INPUT "DIGITE O
CARACTER A CONHECER:";C$: IF LEN(C$)=2
AND LEFT$(C$,1)=CHR$(1) THEN C=BASE(2)+
8*(ASC(RIGHT$(C$,1))-64): PRINT: GOTO30
20 C=BASE(2)+8*ASC(C$):PRINT
30 PRINT"FORMAÇÃO GRÁFICA DO CARACTER "
;C$:PRINT
40 FORA=CTOC+7
50 V=VPEEK(A):CD$=RIGHT$("00000000"+BIN
$(V),8)
60 FOR B=1TO8:IF MID$(CD$,B,1)="1" THEN
 PRINT CHR$(219);ELSE PRINT CHR$(32);
```

```
70 NEXTB:PRINT: NEXT A
80 PRINT"DEFINIÇA0 BINÁRIA DO CARACTER
"C$:PRINT:GOSUB 320
90 PRINT: PRINT"APERTE QUALQUER TECLA P
/REDEFINI-LO"
100 R$=INPUT$(1)
110 LOCATE 0,22:PRINT "USE ESTAS LINHAS
 DATA P/A REDEFINIÇÃO:":PRINT:LIST400-4
70
200 INPUT"CONFIRME O CARACTER A REDEFIN
IR:";C$: IF LEN(C$)=2 AND LEFT$(C$,1)=C
HR$(1) THEN C=BASE(2)+8*(ASC(RIGHT$(C$,
1))-64): GOTO220
210 C=BASE(2)+8*ASC(C$)
220 FOR A=C TO C+7
230 READ A$
240 VPOKEA,VAL("&B"+A$)
250 NEXT A: CLS
260 FOR I=BASE(0)+520 TO 879 STEP 2
270 IF LEN(C$)=1 THEN VPOKEI,ASC(C$):VP
OKEI-1,32
280 IF LEN(C$)=2 THEN VPOKEI,ASC(RIGHT$
(C$,1))-64:VPOKEI-1,32
290 NEXT I
300 LOCATE 0,0:GOSUB 310:PRINT,,"NOTE C
OMO FICOU O CARACTER REDEFINIDO":LOCAT
E 0,22:END
310  PRINT"NOVA DEFINIÇA0 BINÁRIA DO CA
RACTER:":PRINT
320 FOR A=CTOC+7
330 V=VPEEK(A)
340 PRINT RIGHT$("00000000"+BIN$(V),8);
" - ENDEREÇO NA VRAM:";A
350 NEXT A
360 RETURN
400 DATA 00000000:'REDEFINA O CARACTER
410 DATA 00000000:'MOSTRADO, FAZENDO O
420 DATA 00000000:'SEU DESENHO COM 1 E
430 DATA 00000000:'0 NAS LINHAS DATA(.
440 DATA 00000000:'DEPOIS, DIGITE GOTO
```

```
450 DATA 00000000:'200 E TECLE RETURN.
460 DATA 00000000:'NOTE NAS LINHAS SE-
470 DATA 00000000:'GUINTES COMO FICOU:
```

O programa é auto-explicativo. Rode-o diversas vezes, redefinindo os caracteres que desejar.

Para retornar aos caracteres normais, comande diretamente:

```
SCREEN 0 / RETURN
```

O programa utiliza as duas tabelas do modo TEXTO 1. A título de prática, adapte-o de forma que utilize outras tabelas do PTV.

Instruções e funções de uso restrito

21

INSTRUÇÃO CALL

Executa rotina ou instrução expandida contida em cartucho ROM utilizável através de "slots". Sua sintaxe é:

CALL |nome da instrução| (argumentos)

sendo que nome da instrução e argumentos devem ser os indicados pelas instruções que acompanham o cartucho.

Em vez da palavra CALL, pode ser usado o sinal _ (travessão).

FUNÇÃO PAD

Fornece dados lidos num "touch pad" (mesa digital ou digitalizadora), ainda inexistente no mercado brasileiro.

Embora os manuais dos MSX nacionais citem a função PAD, haverá conveniência em estudá-la apenas quando estiver disponível o acessório.

FUNÇÃO PDL

Fornece um valor entre 0 e 255, representando o estado do acessório "paddle", caso este seja usado numa das entradas para "joystick".

O acessório "paddle" é uma relíquia dos primórdios dos jogos de

"vídeo", raramente usado na atualidade.

A função PDL tem a seguinte sintaxe:

PDL (n)

sendo n um número entre 1 e 12, correspondendo os números pares ao conector de "joystick" número 2, e os números ímpares ao conector nº 1.

A função STRIG pode ser utilizada para interpretar o estado do botão disparador do "paddle".

FUNÇÃO INP

Recebe um número de código entre 0 e 255 através de uma porta de entrada do computador. Sua sintaxe é:

INP (n)

sendo n o número da porta especificada.

INSTRUÇÃO OUT

Transmite um número de código entre 0 e 255 para uma porta usada para saída de dados do computador. Sua sintaxe é:

OUT |número da porta|,|número de código|

INSTRUÇÃO WAIT

Controla o estado de uma porta de entrada do computador. Sintaxe da mesma:

WAIT |número de porta|,|expressão|,|expressão|

Observação: As funções e instruções citadas neste capítulo são mal ou pouco esclarecidas nos manuais dos MSX nacionais.

Índice remissivo dos comandos, instruções e funções do basic MSX

Este índice cita apenas o número de páginas em que os comandos, instruções ou funções são introduzidos com explanação, embora sejam citados e utilizados em diversas outras páginas.

Não estão incluídos aqui os comandos, instruções e funções do DOS MSX (sistema operacional de discos do MSX) ainda inexistente no mercado brasileiro na data de elaboração deste livro.

O "Manual de Basic" que acompanha o HOTBIT HB-8000 cita os seguintes comandos, instruções e funções do HB-DOS, embora não detalhe suficientemente sua aplicação e não aborde o sistema operacional correspondente:

COPY (I)	FILES (C)	MKI$ (F)
CVI (F)	GET (I)	MKS$ (F)
CVS (F)	KILL (I)	MKD$ (F)
CVD (F)	LOC (F)	NAME (I)
DSKF (F)	LOF (F)	PUT (I)
FIELD (I)	LSET (I)	RSET (I)

C = comando / F = função / I = instrução

Os manuais que acompanham o EXPERT não fazem nenhuma referência ao DOS MSX.

O sistema operacional de discos para o MSX será objeto de outro livro nosso sobre essa linha de computadores.

Bibliografia

ALEPH PUBLICAÇÕES E ASS. PED. LTDA. "Dominando o Expert - Gradiente" e "Linguagem Basic MSX - Gradiente".

EPCOM EQUIPAMENTOS ELETRONICOS DA AMAZONIA LTDA. "Manual do Usuário Home Computer Hotbit HB-8000" e "Manual de Basic Home Computer Hotbit HB-8000".

CASARI, N. "TK 85 - Domínio Rápido". São Paulo, Atlas, 1985.

MAGRI, J. A. "Programação Basic". 2ª ed. São Paulo, Atlas, 1985.

RODRIGUEZ, C. G. F. "Basic - Um Enfoque Profissional". São Paulo, Atlas, 1985.

SHIMIZU, T. "Introdução à Ciência da Computação". São Paulo, Atlas, 1985.

YONG, C. S. "Banco de Dados". São Paulo, Atlas, 1985.

1988

Impressão e acabamento
(com filmes fornecidos):
EDITORA SANTUÁRIO
Fone (0125) 36-2140
APARECIDA - SP

MSX - Prática e Domínio

Nélson Casari

Este livro reúne informações sobre comandos, instruções e funções do BASIC MSX e busca dar solução a muitas questões que surgem no uso do manual que acompanha a máquina. Visa proporcionar ao leitor informações sobre como trabalhar com a máquina e obter dela o máximo proveito possível.

Os microcomputadores do padrão MSX caracterizam-se sobretudo pela intercambialidade de *software* e periféricos, a par de constituir-se numa das mais poderosas máquinas de oito *bits* da atualidade.

Destinado ao novo usuário de computador padrão MSX, o livro propõe-se a ministrar os conhecimentos indispensáveis ao entendimento da máquina, visando seu rápido domínio através da prática intensiva com a mesma, através de programação e do exame de questões inerentes.

MSX – Prática e Domínio apresenta considerável quantidade de programas concisos e comentados, a maioria dos quais com finalidades práticas e aplicativas.

Tratamento especial é dispensado ao processamento de arquivos – banco de dados – em fitas cassetes, permitindo utilização ilimitada do microcomputador sem o concurso de periféricos dispendiosos.

Todo o BASIC MSX – poderoso e dos mais versáteis – é repassado de maneira prática no livro.

Gráficos, som, efeitos sonoros e jogos são abordados de modo objetivo, a fim de permitir ao proprietário ou usuário de um computador padrão MSX proveito também na área de lazer e entretenimento.

APLICAÇÃO

Manual destinado aos usuários de microcomputadores do padrão MSX. Complementa o Manual do Fabricante, permitindo a utilização mais intensiva dos recursos desses equipamentos. Recomendado para o desenvolvimento de cursos de treinamento.

publicação atlas