



Plus

CARACTERÍSTICAS TÉCNICAS

UNIDADE CENTRAL

UPC : Microprocessador 6502, 8 bits, 1 MHz

ROM : 12 Kbytes

RAM : Dinâmica 48 Kbytes

Expansões : Módulos de 128 e 256 Kbytes, etc...

TECLADO : Padrão QWERTY, 69 Teclas, tipo profissional.

VIDEO

PADRÃO : Televisor doméstico preto e branco ou a cores, por Modulação de RF no canal 7 - Padrão PAL/M.

- OPÇÕES : 1. Televisor doméstico preto e branco ou a cores, modificado para Video Composto ou
2. Monitor profissional monocromático ou colorido.

FORMATO DA INFORMAÇÃO

24 linhas de 40 colunas no modo texto, possibilidade de gráficos de baixa resolução de 40 x 40 pixels em 16 cores.

GRÁFICOS DE ALTA RESOLUÇÃO EM 280 x 198 PIXELS DE 4 CORES.

Pode ser misturada informação gráfica e texto (4 linhas inferiores).

DUAS PÁGINAS GRÁFICAS INTERCAMBIÁVEIS POR PROGRAMA.

ACESSO A PERIFÉRICOS :

Saídas independentes para :

Monitor de Video ou televisor comum modificado (video composto)

Televisor comum (RF)

Gravador Cassete comum

Controladores de jogos (PADDLES / JOYSTICKS)

7 Conectores para ligação de periféricos, interfaces e expansões (SLOTS)

MEMÓRIA AUXILIAR

-FITA MAGNÉTICA : Conexão para 1 gravador cassete comum, monaural, velocidade de transmissão de 1500 / Bauds, sem controle de motor.

-DISCOS FLEXÍVEIS: 1 ou 2 Disk-Drives de 5 1/4" por interface, com 140 Kbytes de capacidade cada um.

FONTE DE ALIMENTAÇÃO :

Universal 110/220 V) conforme ajuste da chave de seleção de voltagem ,localizada na parte lateral da fonte (parte interna do gabinete), com proteção contra curtos sobrecargas.

SISTEMAS OPERACIONAIS

PADRÃO : Apple Soft (Basic)
Monitor (linguagem de máquina)

DISCO : Dos 3.3 (APPLE DOS)
PRONTO DOS
SUPER-DOS
CP/M 2.2
OUTROS

LINGUAGEM : BASIC com comando Gráficos, residente em ROM (Apple Soft)

Linguagem de Máquina : (Monitor) do 6502

VERSÃO DISCO :

BASIC EXTENDIDO, INTEGER-BASIC, COBOL ASSEMBLER, FORTH, PASCAL, LOGO, FROTRAN, compiladores e interpretadores, etc. (através de Terceiros)

Acessórios : Modulador de RF, Cabo ,Conector para antena de VHF.

- Cabos para ligação de gravador cassete
- Cabo de alimentação
- Manual do usuário

OBS: Todas as referências à geração de cores neste manual levam em consideração a existência da interface PAL/M, que é um item opcional. Na ausência da mesma não é possível gerar algumas cores.

ÍNDICE

CAPÍTULO 1 - INSTRUÇÕES PARA OPERAÇÕES

.. Instalação do equipamento e seus acessórios.....	1
.. Material necessário.....	1
.. Como ligar o computador na TV.....	2
.. Como ligar os "Joysticks".....	3
.. Como ligar o gravador.....	3
.. Como instalar a unidade de disco.....	4
.. Você já ligou o computador. Agora conheça o teclado.....	4
.. Notação do teclado.....	8
.. Controle e outras características.....	8
.. Ligando o gravador.....	10
.. Procedimento usual para carregar fitas.....	12
.. Usando uma unidade de disco.....	13
.. O "menu".....	15
.. Parando o computador.....	16
.. Colocando cor na TV.....	16

CAPÍTULO 2 - INICIAÇÃO À LINGUAGEM

.. Iniciação à linguagem.....	19
.. Uma primeira noção do comando PRINT.....	19
.. Formato dos números.....	21
.. Explicações sobre o RETURN	23
.. Características simples de adição ou 0 que fazer antes de teclar RETURN.....	23
.. Mensagens de erro nos gráficos.....	29
.. Desenhando linhas.....	30
.. Os controles (joysticks).....	32
.. Arquivos e outras possibilidades de cálculo.....	33
.. Prioridade, ou quem vem primeiro?.....	38
.. Como evitar as prioridades.....	40

CAPÍTULO 3 - PROGRAMAÇÃO ELEMENTAR

.. Execução adiada.....	43
.. Edições elementares.....	47
.. Acrobacias elementares.....	49
.. Algumas outras coisas que tornam a programação mais simples.....	50
.. Eliminando ou copiando o que foi escrito.....	51
.. Comentários sobre o aprendizado.....	53
.. Evitando acidentes.....	54
.. A comparação e a verdade.....	54
.. Ordem ou prioridade para operações.....	59
.. O comando IF.....	59
.. Guardando programas no disco.....	61
.. Guardando programas em um cassete.....	62
.. Outros programas gráficos.....	63
.. "Loops" dos comandos: FOR e NEXT.....	65
.. Um exemplo de programa errado.....	68
.. Mais um exemplo de "loops" internos.....	68
.. Ganhando brilho.....	69
.. Imprimindo com detalhes.....	69

CAPÍTULO 4 - EXECUÇÃO DE GRÁFICOS

Conversando com um programa.....	75
Fora das paredes.....	78
Criando sons.....	80
Barulho para a bola bate e volta.....	81
Notas mais altas e múltiplos Comandos em uma linha.....	82
Notas Aleatórias ("RANDOM").....	83
Simulando um par de dados.....	85
Subrotinas.....	86
Verificações.....	88
Uma subrotina para desenhar um cavalo melhor.....	90
Gráfico de alta- resolução.....	92

CAPÍTULO 5 - ALFANUMÉRICOS E MATRIZES

Dando sequência.....	99
Concatenação.....	104
Mais funções de alfanuméricos.....	105
Introdução aos argumentos (Matrizes).....	107
Mensagem de erros dos agrupamentos.....	110
Conclusão.....	111

APÊNDICE "A"

Resumo dos comandos.....	113
--------------------------	-----

APÊNDICE "B"

Palavras reservadas na linguagem.....	126
---------------------------------------	-----

APÊNDICE "C"

Características de edição.....	129
--------------------------------	-----

APÊNDICE "D"

Mensagem de erro.....	132
-----------------------	-----

GUIA DE REFERÊNCIA.....	136
-------------------------	-----

CAPÍTULO

I

INSTRUÇÕES PARA OPERAÇÃO

instalação do equipamento e seus acessórios

Material necessário

Como ligar o computador na TV

Como ligar os "joysticks"

Como ligar o gravador

Como instalar a unidade de disco

Você já ligou o computador. Agora conheça o teclado

Notação do teclado

Controle e outras características

Ligando o gravador

Procedimento usual para carregar fitas

Usando uma unidade de disco

O "menu"

Parando o computador

Colocando cor na TV

INSTALAÇÃO DO EQUIPAMENTO E SEUS ACESSÓRIOS

Este manual destina-se a ensinar como ligar, operar e programar o seu computador.

Ele pode ajudar tanto a programadores experimentados trazendo recursos que aumentarão muito suas possibilidades de programação, como a iniciantes pois as facilidades que são apresentadas permitirão a você tornar-se um "cobra" em pouco tempo.

Se você, mesmo lendo o manual, encontrar dificuldades, procure o revendedor onde adquiriu seu equipamento e ele terá muita satisfação em lhe dar todas as informações necessárias.

Como todas as atividades de nosso dia-a-dia, programar também é uma atividade que exige exercício; assim, só se aprende a andar, andando, também só se aprende programação, programando.

Instalar o equipamento é uma tarefa tão simples quanto ligar um equipamento de som. Assim, vamos por mãos a obra.

MATERIAL NECESSÁRIO

Antes de iniciar a instalação do seu computador, providencie o material necessário.

Junto com sua unidade APPLE PLUS II, você recebe dois cabos:

- . Um, para conectar o computador a um gravador cassete comum. Esse cabo tem dois plugs em cada uma das pontas.
- . Outro, para conectar o computador a um monitor de vídeo do tipo RCA.

Possui uma caixa de conversor de TV para uso de TV/Computador. Com esta caixa você vai ligar seu computador diretamente no televisor doméstico, sem nenhum prejuízo, modificação, ou alteração em seu televisor a cores. Quando você quiser assistir programas de TV, basta colocar a conexão na posição TV; quando quiser usar o computador, você posicionará a cha-

ve para COMPUTADOR.

Outros equipamentos serão necessários:

1. Gravador cassete comum, para você gerar seus arquivos ou rodar os programas que pretende trabalhar.
2. Opcionalmente, uma unidade de disk drive lhe dará - ao invés do cassete - muito mais capacidade de arquivo e velocidade de pesquisa de dados. Naturalmente se você adquiriu também o disk drive, o gravador cassete poderá ser usado para "Back-up", ou seja, um sistema de segurança do arquivo de dados.
3. Se você é um aficcionado de vídeo jogos, então a aquisição de um par de "joysticks" é muito importante.

Além destes equipamentos, uma série enorme de opcionais estão disponíveis no mercado, principalmente os seguintes:

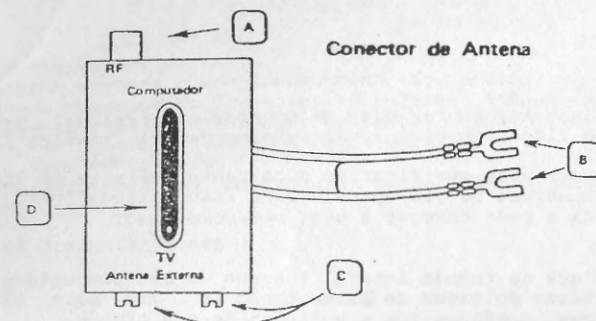
- . Fita de demonstração de cores.
- . Vídeo profissional mono-cromático.
- . Impressora de alta-velocidade.
- . Modem para transmissão telefônica.
- . Diversos interfaces que nossos revendedores terão prazer em lhes explicar e demonstrar.

COMO LIGAR O COMPUTADOR NA TV

Juntamente com seu computador, você recebeu uma caixa metálica (CONVERSOR DE TV) com vários conectores. Esse pequeno aparelho serve para que você use seu televisor doméstico sem modificações, como unidade de vídeo do computador.

Quando você parar de usar o computador, ou chegar no horário da novela, é só mover a chave para a posição TV e a programação volta ao ar, normalmente.

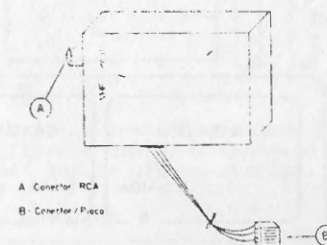
Essa é a primeira peça que você deve instalar no seu TV doméstico para usá-lo com o computador. Se você optou pela aquisição de um monitor de vídeo profissional, leia as instruções que acompanham o próprio monitor para realizar sua ligação.



Para instalar, desligue o televisor, desconecte a antena e ligue-a no conector C da caixinha, agora ligue a ponta dos fios B no lugar onde estava a antena. O conector A é para ligar o cabo que sai do MODULADOR DE RF do seu computador.

A chave D é o seletor: na posição "COMPUTADOR" desliga a antena e liga o sinal do computador, permitindo que o Televisor mostre a imagem gerada pelo computador. Na posição "TV" o sinal de computador é desligado e a antena externa fica conectada, permitindo que o televisor funcione normalmente. Se desejar, cole o conector de antena no televisor, usando a fita adesiva do mesmo.

MODULADOR DE RF



O MODULADOR DE RF vem fixado na parte posterior direita interna do gabinete do computador, e é mostrado na fig. acima.

Ligue o cabo RCA/RCA (isto significa um conector macho tipo RCA em cada extremidade) no conector A de CONECTOR DE ANTENA e no conector A DO MODULADOR DE RF. Faça a ligação com cuidado, o conector do MODULADOR DE RF é frágil.

O MODULADOR DE RF do seu computador opera na faixa de frequência correspondente ao canal 7 portanto coloque o seletor neste canal quando for usar o computador.

Caso o seu televisor esteja modificado para receber sinal de

Vídeo Composto, ou for usar um monitor profissional, use a saída do Vídeo Composto localizada no painel traseiro do computador. Resta agora você verificar se a corrente elétrica de sua cidade é compatível de seu aparelho. Se isto estiver "OK", ligue na tomada e pode começar a usar sem computador.

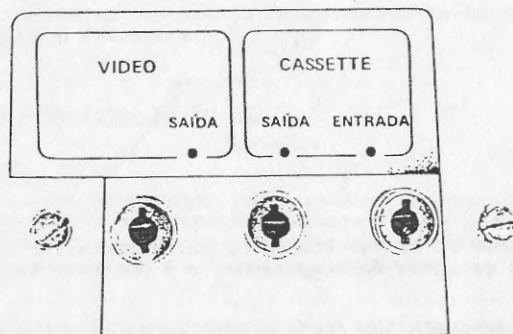
Observe que na tomada lateral interna do seu computador, você pode alterar voltagem de alimentação de 110 para 220 ou vice-versa, conforme for a voltagem de sua cidade.

COMO LIGAR O GRAVADOR

Use para isto os cabos RCA/P2 fornecidos juntamente com o seu computador da seguinte maneira:

No painel traseiro (Fig. abaixo) existem, além da saída de vídeo, os conectores de entrada e saída para gravador, ligue um dos cabos no conector marcado. "Saída" ou "OUT" e a outra extremidade do mesmo na entrada (MIC) de seu gravador, o outro cabo deve ser ligado na entrada do computador e na saída (EAR, ou MONITOR) de seu gravador.

PAINEL TRAZEIRO

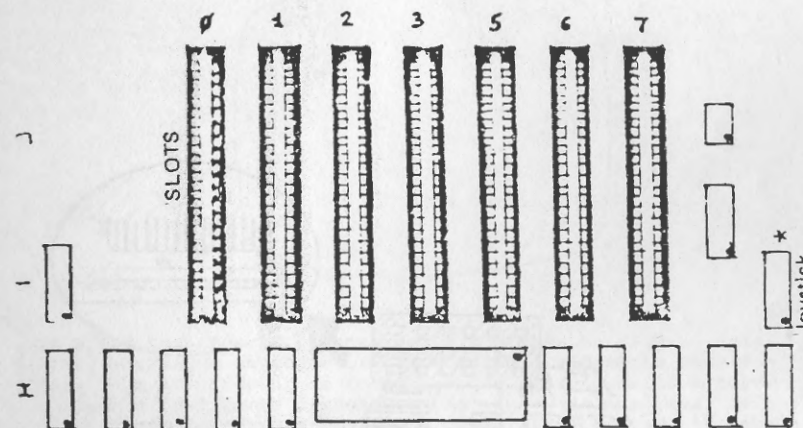


COMO LIGAR OS JOYSTICKS OU PADDLES

Se você gosta de jogos, e adquiriu seu joysticks ou Paddles proceda à ligação da seguinte maneira:

1. Abra o computador
2. Localize o soquete correspondente na fig. abaixo
3. Coloque o conector de Joysticks no soquete, tomando cuidado para não inverte-lo nem dobrar os pinos. O ponto indica o pino 1, coloque-o coincidindo com a marca na placa (virado para o lado do teclado)
4. Passe o cabo para uma das ranhuras do gabinete e recoloque a tampa.

SOQUETE DE JOYSTICKS/SLOTS



IMPORTANTE:- Nunca instalar ou retirar qualquer placa, expansão ou periférico com o equipamento ligado, pois o mesmo pode sofrer graves avarias.

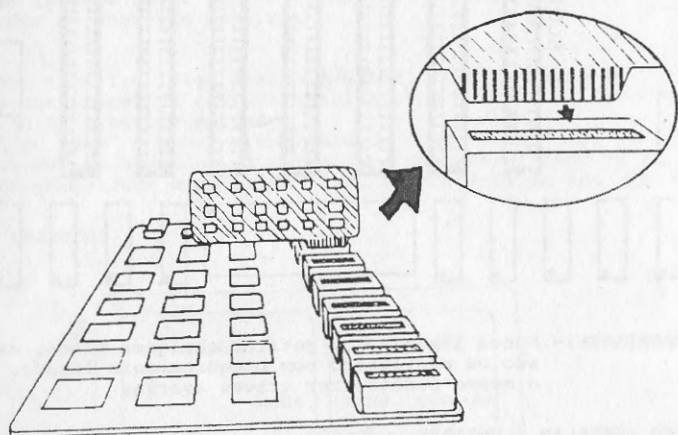
COMO INSTALAR A UNIDADE DO DISCO

A unidade do disco é fornecida normalmente em embalagem isolada, cujas instruções de ligação acompanham o produto. Ali você terá todas informações necessárias.

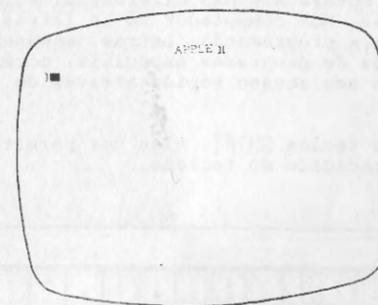
Igualmente todos os demais acessórios e equipamentos que você comprar para seu computador, virão acompanhados de instruções apropriadas para instalação e uso.

Você já verificou se a voltagem estava correta, já ligou o CONVERSOR DE TV e ligou o computador já acoplado com o gravador cassete ou com a unidade do disco.

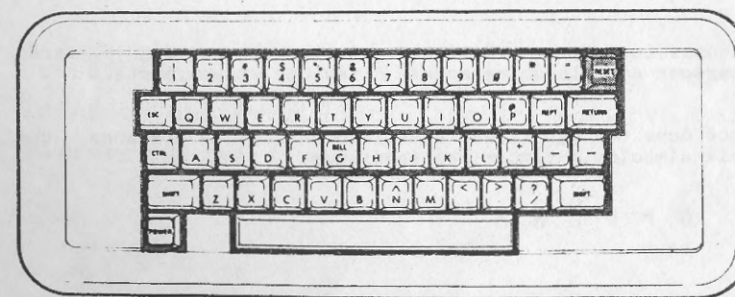
Na tomada do computador você vai observar que tem um pino destinado a ligação "TERRA". É importante sempre ter essa ligação para evitar danos irreparáveis no equipamento.



Quando você liga o computador, acende-se uma luz no teclado, à esquerda, abaixo da tecla SHIFT. Na tela do televisor deve aparecer a palavra APPLE, no canto superior esquerdo, acompanhada do símbolo  e um quadradinho piscando que denominamos "CURSOR".



Se a sua tela não se parecer com a mostrada na figura anterior, verifique se todos os cabos estão devidamente conectados, se o interruptor de força está na posição certa e torne a ligar e desligar o computador. Se ainda assim a sua tela não estiver correta, pressione a tecla RESET situada no canto superior direito do teclado. O computador deverá fazer um "bip" quando esta tecla é pressionada.

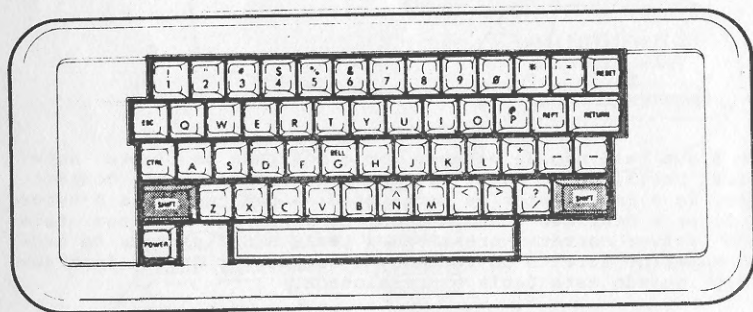


Se você tem uma unidade de disco, quando ligar o seu computador alguns ruídos característicos serão ouvidos, seguidos por

um som mais suave. Uma luz vermelha indicando "EM USO" irá se acender. A unidade de disco irá girar indefinidamente até que seja acionada a tecla RESET. Nessa hora o APPLE PLUS II irá desaparecer do vídeo e o "CURSOR" aparecerá no canto inferior da tela.

Verifique o teclado. Se você conhece o teclado de uma máquina de escrever, notará algumas diferenças. Primeiro, não há letras minúsculas. No computador só as letras maiúsculas são necessárias para a programação. Letras minúsculas poderão parecer em textos de programas especiais, porém, normalmente não é possível o seu acesso rápido através do teclado.

Localize as duas teclas SHIFT, elas nos permitem, praticamente, dobrar a capacidade do teclado.



Algumas teclas, se pressionadas juntamente com a SHIFT farão aparecer na tela um caractere diferente. Experimente.

Você deve ter verificado esse fato nas teclas gravadas com dois símbolos e também nas seguintes:

O P K L N M

Essas teclas quando pressionadas farão aparecer na tela, respectivamente:

- @ [\ ^]

ERRATA

O seu APPLE II PLUS possui uma nova característica. Agora ele é capaz de gerar caracteres em português e utiliza caracteres maiúsculos e minúsculos. Para isso é necessário seguir as seguintes instruções.

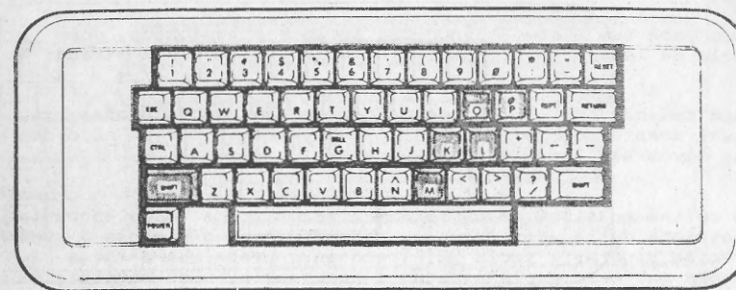
1. As teclas C.A.O. pressionadas juntamente com a tecla SHIFT gerarão caracteres em português.

SHIFT + O = Õ

SHIFT + C = Ç

SHIFT + A = Â

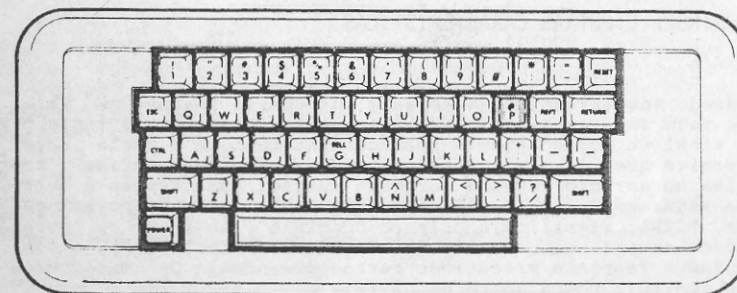
2. Pressionando a tecla LOCK, localizada no canto inferior esquerdo do teclado você poderá gerar / caracteres maiúsculos e minúsculos.

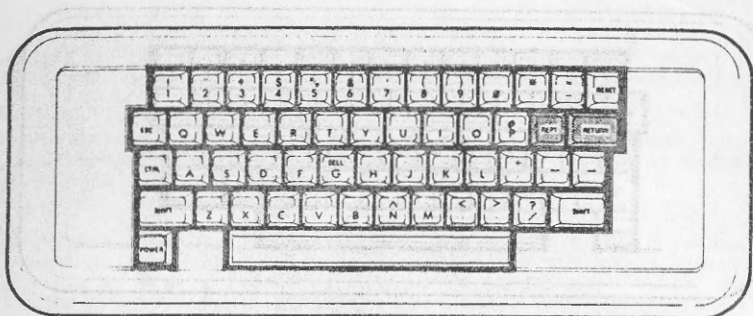


Uma observação importante é que o computador precisa saber distinguir o "zero" da letra "O". O método para isto é cortar o "zero" com uma barra. Tanto o teclado como a tela fazem uma distinção bastante clara entre eles. Experimente. Note também que a letra L não pode ser usada como o número "1".

Note que a tecla ESC ao contrário da SHIFT não precisa e não deve ser pressionada enquanto pressionamos outra tecla.

Para limpar a tela temos que apertar três teclas. Primeiro aperte ESC e solte. Então, segurando a tecla SHIFT, aperte P. O conteúdo da tela irá desaparecer instantaneamente.





LIGANDO O GRAVADOR

Agora pressione a tecla RETURN. O caracter } e o "CURSOR" piscando aparecerão no canto esquerdo da tela. Agora você está pronto para fixar o volume de controle do seu gravador.

Quando você toca uma fita cassete, é usual que seu som possa ser escutado normalmente. Se o volume está muito baixo, você perde algumas palavras da música. Se está muito alto, é irritante.

Quando você liga um gravador ao computador, é com a intenção de transferir a ele informação da fita. Se o volume estiver muito baixo, o computador perderá algumas informações, dando uma mensagem de erro. Se o volume estiver muito alto o seu computador também reclamará.

O volume adequado deve ser achado pelo método das tentativas. Ligue a fita num volume baixo e verifique se a informação foi captada corretamente. Se não funcionar, experimente aumentar um pouco mais o volume, e assim sucessivamente. Quando o volume estiver adequado para o computador ele responderá com um "bip".

Para limpar a tela tecle:

SHIFT

ESC P

Coloque a fita de demonstração no gravador. Para cada posição do volume de controle, faça o seguinte:

PASSO 1 - Volte a fita até o começo,

PASSO 2 - Ligue o gravador e ponha a fita para rodar,

PASSO 3 - Tecle: L O A D RETURN

Quando assim fizer, o cursor irá desaparecer e talvez demore uns 15 segundos antes que algo aconteça. Teremos então as seguintes possibilidades:

A. Aparece a mensagem: ?SINTAX ERROR,

B. Nada acontecerá,

C. A mensagem ERR ou ERRERR aparecerá (com ou sem "bip"),

D. O computador faz um "bip" e nada aparece.

No caso A, não mude o volume de controle, e sim volte a fita para o início, dando as instruções novamente (PASSO 1).

Nos casos B e C, após os 15 segundos, se não tiver nem um caracter nem o cursor e se o computador não responder ao teclado, aperte RESET, aumente um pouco o volume de controle e volte para o PASSO 1. Se o cursor aparecer na tela antes da fita ser rodada, simplesmente desligue o seu computador e experimente carregá-la (LOAD) novamente.

No caso D, você está no caminho certo. Após o "bip", espere mais 15 segundos. Ou você receberá uma mensagem de erro (caso C), ou o caracter } e o cursor piscando reaparecerão. Se isto acontecer, pare e volte a fita. Marque a posição do volume de controle do gravador para que você possa usá-la cada vez que utilizar esse gravador. Depois escreva:

R U N RETURN

A tela deverá ficar assim:

PROGRAMAS DE DEMONSTRAÇÃO APPLE PLUS II

PARA VER UMA DEMONSTRAÇÃO, TECLE O NÚMERO DESTA E ENTÃO PRESSIONE A TECLA MARCADA "RETURN" NO LADO DIREITO DO TECLADO. PARA PARAR QUALQUER DELAS PRESSIONE NOVAMENTE "RETURN".

1. NOMES PADRÕES DAS CORES
2. NÚMEROS PADRÕES DAS CORES
3. CALEIDOSCÓPIO
4. TELA DESENHADA

QUAL DELAS VOCÊ DESEJA VER? ■

PROCEDIMENTO USUAL PARA CARREGAR FITAS

Uma vez que o volume de controle já está na posição certa,

- PASSO 1 - Retorne a fita ao ponto de início. Anote sempre o ponto de início e fim marcados pelo numerador de seu gravador,
- PASSO 2 - Ligue o gravador e ponha a fita para rodar,
- PASSO 3 - No teclado do computador, digite:
L O A D RETURN
Após teclar RETURN o "CURSOR" irá desaparecer. Na da acontecerá durante 5 a 20 segundos, então o computador emitirá um "bip". Isso significa que a informação contida na fita começou a entrar no computador. Após mais um tempo, dependendo da quantidade de informação contida na fita (normalmente alguns minutos), o computador emitirá um novo "bip" e o "CURSOR" e o caractere], reaparecerão.
- PASSO 4 - Pare o gravador e volte a fita. A informação foi transferida e por ora você não mais usará o gravador.
- PASSO 5 - Digite RUN, tecla RETURN e o programa começará a ser executado.

Várias palavras são usadas para descrever o processo de in-

serir e retirar informações do computador. O ato de ler a informação contida numa fita é conhecido em computação por "loading" (que em inglês significa "carregando"), ou seja, quando você carrega um computador com informações contidas numa fita, você executa um "LOAD" da fita para o computador.

SUGESTÃO UTIL

Quando você tentar ouvir o que está gravado numa fita contendo programas, note que a informação começa com um som constante, seguido de um pequeno "bip" e novamente um som constante (o som está numa frequência de 1000 Hz). Após o som, um acumulado de barulhos aparecerão.

USANDO UMA UNIDADE DE DISCO

A Unidade de Disco ("Disk-Drive") é muito mais prática e rápida que um gravador cassete, entretanto, discos flexíveis ("disketes") e unidades de disco são objetos muito delicados. Alguns cuidados terão que ser tomados para protegê-los. Informações sobre estes cuidados estão no manual do DOS (Sistema de Operação do Disco).

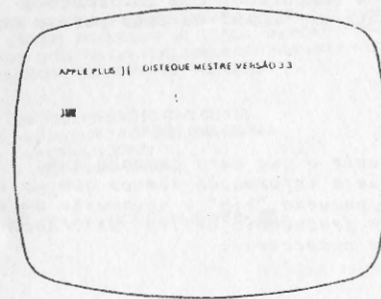
Coloque o disco "DOS" na unidade com a etiqueta voltada para cima e corte oval no verso, conforme descrito no manual do DOS.

Uma das razões que tornam a unidade de disco tão fácil de se usar é a possibilidade de inserir e ler diversos grupos de informações. Esses grupos de informações estão arquivados no disco com seus respectivos nomes. O nome de um programa é escolhido na hora da gravação e servirá para identificar um grupo de informações ali arquivadas (por exemplo, ENDEREÇOS).

Os programas que armazenam a relação de arquivos, guardam-nos, escrevem-nos e fazem muitas outras rotinas de processamento, são os componentes do DOS (Sistema de Operação do Disco).

Há várias maneiras de introduzir no computador as instruções do DOS. Uma delas é simplesmente desligar e ligar o seu computador. A luz vermelha do painel da unidade de disco irá acender novamente, e os mesmos barulhos característicos de quando nós ligamos o computador pela primeira vez, irão aparecer. Desta vez o disco irá parar de girar sozinho.

Quando os barulhos cessarem e a luz vermelha do painel se apagar, o título APPLE irá desaparecer e uma mensagem aparecerá na tela.



Quando você obtiver esta mensagem saberá que o DOS foi inserido. Um outro meio de inserir o DOS é escrever: P R # 6

SHIFT

&

P R 3 6 RETURN

Se o cartão de controle (interface) não estiver ligado ao conector 6 e sim no conector 3, escreva: P R SHIFT seguido do número do conector que o cartão estiver conectando. Em seguida, tecle RETURN

SHIFT

#

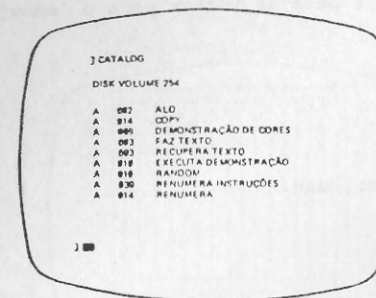
P R 3

O "DOS" é um disco muito especial. Ele tem programas bastante úteis para o seu aprendizado. Para saber quais programas ele contém, use o comando CATALOG. Digite:

C A T A L O G RETURN

e uma lista de nomes de arquivos ou programas aparecerá na

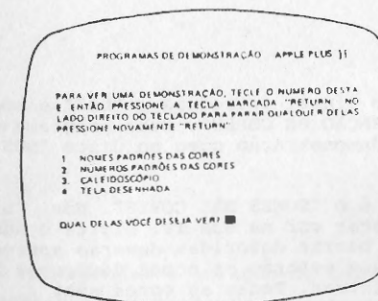
tela.



O primeiro programa que você irá necessitar é chamado DEMONSTRAÇÃO DE CORES. Localize esse nome na lista (catálogo) e escreva:

RUN DEMONSTRAÇÃO DE CORES

Depois tecle RETURN. A tela deverá se parecer com esta:



O "MENU"

Costuma-se chamar esta lista de descrições numeradas de

"menu". Ela funciona como um menu (cardápio) de restaurante. Experimente selecionar uma das demonstrações de cores escrevendo o seu número seguido da tecla RETURN. Quando você estiver em uma das opções e desejar voltar para o "menu", simplesmente tecle RETURN.

PARANDO O COMPUTADOR

Para parar o computador, use:

CRTL

C

Imediatamente aparecerá na tela o caracter] e o "CURSOR", indicando que o computador está aguardando novas instruções. Se você parou um programa, a seguinte mensagem aparecerá no vídeo: BREAK IN 380. O número 380 serve apenas como exemplo, e variará de programa para programa e da linha em que parou a execução, como veremos mais adiante.

Se você quiser que o programa rode novamente, tecle RUN seguido, é claro, do RETURN.

COLOCANDO COR NA TV

Se o "menu" não está no vídeo, introduza o "DOS" e rode o programa chamado "DEMONSTRAÇÃO DE CORES". Esse programa está tanto na fita cassete de demonstração como no disco "DOS".

Um dos itens do "menu" é o "NOMES DAS CORES". Nós usaremos este programa para colocar cor na sua TV. Digite o número 1 e tecle RETURN. Várias barras coloridas deverão aparecer na tela. Embaixo de cada uma estarão os nomes das cores básicas, abreviados por quatro letras. Todas as cores são:

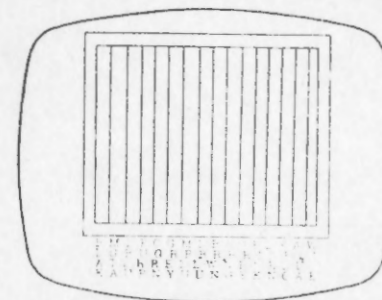
0 Preto	6 Azul	11 Rosa
1 Vermelho	7 Azul claro	12 Verde
2 Azul escuro	8 Marrom	13 Amarelo
3 Magenta (lilás)	9 Laranja	14 Água (azul piscina)

4 Verde escuro

10 Cinza

15 Branco

5 Cinza



Ajuste o contraste, brilho e matriz no vídeo, de modo a conseguir a melhor definição.

A regulagem de sua TV a cores dependerá de uma série de fatores característicos dela. Vários controles são importantes. Utilize sempre o programa "DEMONSTRAÇÃO DE CORES" para essa regulagem.

OBS: Todas as referências à geração de cores neste manual levam em consideração a existência da interface PAL/M, que é um item opcional. Na ausência da mesma não é possível gerar algumas cores.

CAPÍTULO

2

INICIAÇÃO À LINGUAGEM

- . Iniciação à linguagem
- . Uma primeira noção do comando PRINT
- . Formato dos números
- . Explicações sobre o RETURN
- . Características simples de adição ou 0 que fazer antes de teclar RETURN
- . Colocando cores na tela
- . Mensagens de erro nos gráficos
- . Desenhando linhas
- . Os controles (joysticks)
- . Arquivos e Outras possibilidades de cálculo
- . Prioridade, ou Quem vem primeiro?
- . Como evitar as prioridades

INICIAÇÃO À LINGUAGEM

O seu computador possui uma linguagem própria para programação que é o APPLESOFT. Antes de rodar um programa é necessário que você verifique em qual linguagem ele foi escrito. No caso de você desejar usar outra linguagem como INTBASIC, PASCAL, FORTRAN, COBOL, etc., é necessário a introdução de cartões específicos de cada linguagem.

UMA PRIMEIRA NOÇÃO DO COMANDO PRINT

Agora que o caracter] e o cursor estão na tela e caso você tenha uma unidade de disco e já tenha providenciado a instalação do DOS, tudo estará pronto para o uso da linguagem. Escreva:

```
PRINT "INICIO"
```

e o computador imprimirá na linha seguinte, do vídeo, a palavra:

```
INICIO
```

Se não o fizer, verifique se não esqueceu de teclar RETURN. Se você escrever errado a palavra PRINT, a mensagem ?SNTAX ERROR aparecerá na tela. Se você esquecer as duas aspas ou só a primeira o computador imprimirá um zero (você saberá que é zero e não a letra O pelo traço):

```
0
```

Se a segunda aspa é o último caracter antes do RETURN, não é necessário colocá-la: a palavra INICIO será impressa com ou sem ela. De qualquer maneira é útil colocá-la. O hábito de colocar a aspa no final será importante mais tarde.

O comando PRINT "INICIO" é uma instrução para o computador imprimir na tela todos os caracteres entre as aspas. Você pode usar o comando PRINT para imprimir qualquer mensagem que desejar. No entanto, se você teclar mais que 240 caracteres, o computador começará a fazer "bip", depois colocará uma barra no final da linha e deixará você começar tudo outra vez.

ESTE TEXTO MOSTRA AO OPERADOR INICIANTE
 , QUE UMA LINHA DE TEXTO NÃO DEVE TER MA
 IS QUE 240 CARACTERES. ESTA DISTRAÇÃO DO
 OPERADOR NÃO SERÁ PERDOADA PELO COMPUTA
 DOR, O QUAL BARRARÁ A ENTRADA DOS PROXIM
 OS CARACTERES, O QUE, AS VEZES, PODE SER
 DEVERES INCONVENI \
 JENTE. PORTANTO ATENÇÃO!

Agora experimente o comando:

PRINT "150 "

O computador imediatamente imprime o número 150 na linha se-
 guinte, como esperado. Agora tecle:

PRINT 150

e o computador novamente imprimirá o número, sem nenhuma ob-
 jeção quanto a falta das aspas. De fato, o seu computador
 permite imprimir qualquer número sem aspas.

Sem muito esforço, o computador pode ser usado como uma sim-
 ples máquina de calcular. Experimente isto:

PRINT 3 + 4

e a resposta 7 aparece na próxima linha.

O seu computador pode fazer cinco operações aritméticas bási-
 cas:

1. ADIÇÃO - Indicada pelo sinal mais +
2. SUBTRAÇÃO - Mostrada pelo sinal menos -

3. MULTIPLICAÇÃO - Muitas pessoas usam um "x" para repre-
 sentar a multiplicação; mas este símbo-
 lo poderia ser confundido com a letra
 "x". Outras pessoas usam um ponto (.),
 mas este poderia ser confundido com a
 indicação de números decimais. O seu com-
 putador - para evitar confusões -, usa
 um asterisco (*) para representá-la.

Para saber quanto vale 7 vezes 8, sim-
 plesmente tecle:

PRINT 7 * 8

4. DIVISÃO - Use uma barra (/) para indicar a divi-
 são.

Para dividir 63 por 7, escreva:

PRINT 63 / 7

e a resposta correta aparecerá na tela.
 Experimente dividir 3 por 2. A resposta
 é um e meio. O computador lhe dá a res-
 posta na forma decimal: 1.5. Note que
 para a separação da parte inteira das
 casas decimais, seu computador utiliza
 um ponto (.) no lugar da vírgula usual.

5. POTENCIAÇÃO - É bastante frequente a necessidade da
 multiplicação de um número por ele mes-
 mo. Ao invés de usarmos a instrução:

PRINT 4 * 4 * 4 * 4 * 4, devemos usar:

PRINT 4 ^ 5 o que simplifica bastante

O caractere ^ é escrito teclando:

SHIFT

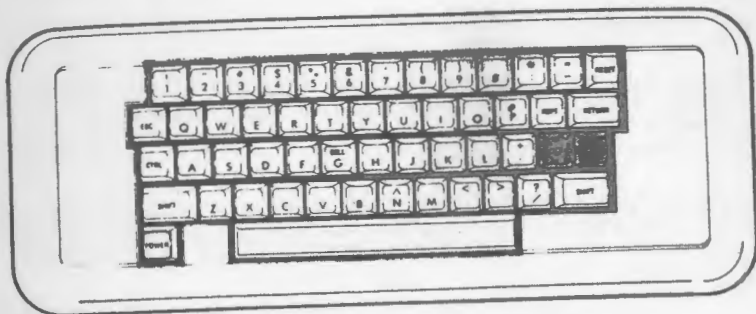
N

Não há nada de especial sobre a poten-
 ciação. Ela é simplesmente a abreviação
 de repetidas multiplicações.

FORMATO DOS NÚMEROS

Digite:

Esse é o ponto onde as setas + e -, são bastante usadas.



A tecla + é semelhante ao retrocesso da máquina de escrever. Algumas aplicações tornarão isto bastante claro. Escreve (exatamente igual) o comando:

```
PRINT APPLE
```

e como de costume, tecla RETURN. O computador responderá com:

```
?SINTAX ERROR
```

porque está faltando uma aspa. Se tivéssemos digitando:

```
PRINT "APPLE"
```

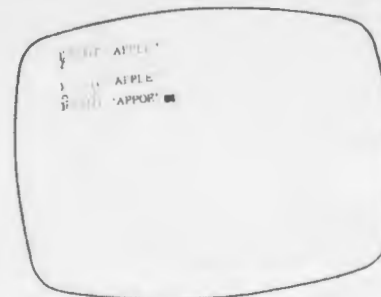
o computador responderia escrevendo APPLE. Experimente.

Agora, sem teclar RETURN, escreva a instrução errada:

```
PRINT "APPOE"
```

Desde que você não tenha pressionado RETURN, nada aconteceu ainda.

Como mostra a figura abaixo, o "CURSOR" está parado à direita da segunda aspa.



Para mudar APPOE

Para APPLE

nós podemos usar a tecla +. Note que cada vez que você pressiona esta tecla, o cursor piscando move um espaço para trás (para a esquerda).

Retroceda o cursor até o 0: Digite L. Como você vê, o L se coloca no lugar do 0.

Agora tecla RETURN. Você obteve APPL no lugar de APPLE. Isto é porque você retrocedeu sobre OE. Qualquer caracter da linha em que você está escrevendo sob o qual o cursor retrocedeu, não é enviado ao computador quando você tecla RETURN. Uma solução seria corrigir o O através do retrocesso e então teclar:

```
SHIFT
L E " 2 RETURN. Experimente.
```

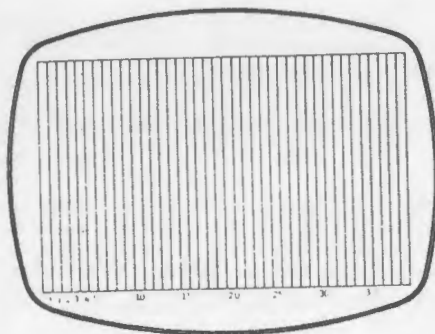
Funciona. No entanto, há um método mais simples. Quando você pressiona a tecla +, o "CURSOR" se move para a direita. Assim que o "CURSOR" se mover para a direita, passando por cima do caracter, produzirá o mesmo efeito que reescrever o caracter. A tecla + é conhecida por tecla de "reescrever". Novamente, digite:

```
PRINT " COMPUTADOR "
```

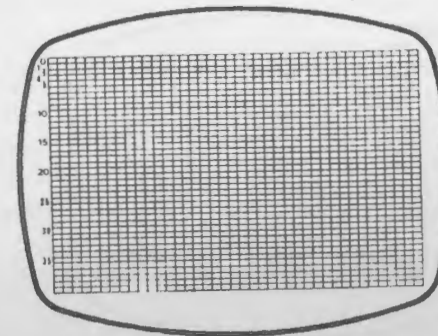
então, retroceda até o F e mude-o para P. Para completar a correção, simplesmente pressione a tecla de reescrever cinco vezes e depois tecle RETURN. Tudo funciona? O uso das teclas retrocesso e reescrever lhe pouparão bastante tempo. Cometa alguns erros e use estas teclas para se familiarizar com elas.

COLOCANDO CORES NA TELA

Para colocar gráficos coloridos na tela, nós precisamos de um meio para descrever qual das 16 cores possíveis nós desejamos e onde as queremos. Para especificar aonde o cor deve ir, nós dividimos a tela em 40 colunas verticais numeradas de 0 a 39. A coluna 0 está bem à esquerda da tela e os números aumentam para a direita. Você deve estar pensando porque os números não vão de 1 a 40 e sim de 0 a 39. Assim que você tiver mais experiência em programação você descobrirá que a escolha que nós fizemos é mais cômoda para se trabalhar mesmo que não pareça assim à primeira vista.



A tela é também dividida em 40 linhas horizontais, novamente numeradas de 0 a 39. As linhas horizontais começam com 0 no topo da tela, indo até a linha 39 na parte inferior da tela. Estas linhas se cruzam com as colunas, partindo cada coluna em 40 "quadrados" numerados de 0 (quadrado superior) a 39 (quadrado inferior). Para aqueles que gostam da terminologia formal, reconhecerão que isto é um mero sistema de coordenadas cartesianas. Para aqueles que não gostam, apenas pensem em termos de colunas e de quadrados.



Para obter cores na tela, inicie escrevendo a seguinte instrução:

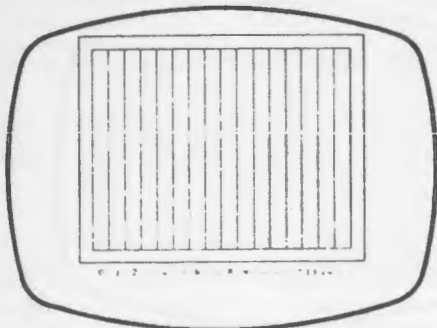
GR

Você se lembrou do RETURN, sem dúvida. Quando você usa este comando a tela se limpa sozinha, deixando apenas 4 linhas para texto, em sua extremidade inferior. O GR significa GRÁFICO. Para retornar ao que estava (antes de teclar GR) use o comando:

TEXT

Quando você escrever este comando a tela mudará rapidamente para muitos sinais @. Isso é normal. Experimente escrever a instrução TEXT e depois voltar para gráfico escrevendo a instrução GR.

Antes de você poder colocar um ponto colorido na tela, você precisa dizer ao computador qual cor deseja. Há disponibilidade de 16 cores. Você já viu antes: elas são numeradas de 0 a 15, como mostrado na "DEMONSTRAÇÃO DE CORES".



Suponha que você queira colocar um ponto verde em algum lugar. Você precisa primeiro teclar o comando GR e então escrever:

```
COLOR = 12
```

Isto significa que qualquer ponto (ou mancha, ou quadrado) colorido que você faça, será verde. De fato, até que ele receba uma outra instrução, tudo o que se colocar na tela será verde, exceto, é claro, da pequena área no final da tela reservada para suas instruções.

Para colocar um ponto no canto superior esquerdo da tela (coluna 0, em cima do quadrado zero) você deve escrever:

```
PLOT 0,0
```

Para colocar um ponto da mesma cor no canto superior direito, você precisa especificar: coluna 39, quadrado 0. Então escreva:

```
PLOT 39,0
```

Note que você sempre especifica a coluna primeiro. Agora ponha um quadrado laranja no canto inferior esquerdo. Primeiro mude a cor. Lembre-se, você deve realmente fazer estes exercícios, não apenas pensar sobre eles. Então escreva:

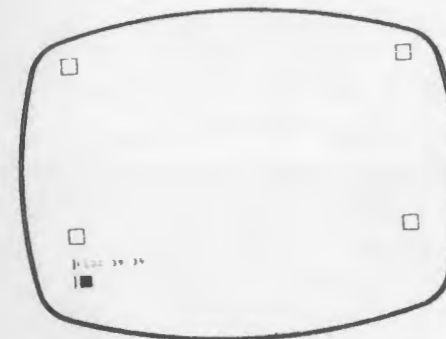
```
COLOR = 9
```

Nada aconteceu com o que estava na tela (mesmo que você tenha se lembrado de teclar RETURN). Mas o computador lembrará que quando você colocar algo na tela, este será laranja e não verde. Agora que você já escolheu a cor, pode colocar um ponto no canto inferior esquerdo da tela, ou seja, na coluna 0 e quadrado 39:

```
PLOT 0,39
```

Funcionou? Você não esqueceu de teclar RETURN?

Agora ponha um ponto magenta no canto inferior direito da tela. Veja se sua tela está como a da figura a seguir:



MENSAGENS DE ERRO NOS GRÁFICOS

Há duas mensagens de erro que são bastante comuns quando se está usando o comando PLOT. Você já sabe que se escrever:

```
PLAT, ou
```

```
PLOC em vez de
```

```
PLOT, obterá a mensagem de erro
```

```
?SINTAX ERROR
```


Uma nova mensagem de erro aparece quando você quer colocar na tela um ponto cujas coordenadas são inferiores ou superiores às permitidas no comando PLOT. Digite:

```
PLOT 13,85
```

e você obterá a mensagem:

```
?ILLEGAL QUANTITY ERROR
```

Esta mensagem significa que você tentou marcar um ponto fora da área permitida e fora da tela. Os maiores números que você pode usar no comando PLOT são 39 para a primeira coordenada, e 47 para a segunda. O uso de números maiores que 39 para a segunda coordenada, como no caso abaixo:

```
PLOT 20,45
```

apenas gerará um caracter peculiar na área de texto, no final da tela.

Se tentar usar números negativos no comando PLOT, também receberá a mensagem:

```
?ILLEGAL QUANTITY ERROR
```

DESENHANDO LINHAS

Suponhamos que você queira desenhar uma linha horizontal azul claro da coluna 5 até a coluna 9 do nível do quadrado 14. Você poderia escrever:

```
COLOR = 7
PLOT 5,14
PLOT 6,14
PLOT 7,14
PLOT 8,14
PLOT 9,14
```

Note que as ligações entre quadrados adjacentes não aparecem, elas formam uma linha contínua. Entretanto, há um método bem mais simples de se desenhar linhas horizontais. Suponhamos que você queira desenhar uma linha horizontal verde escuro no meio da tela. Usando o caminho mais longo teríamos que es-

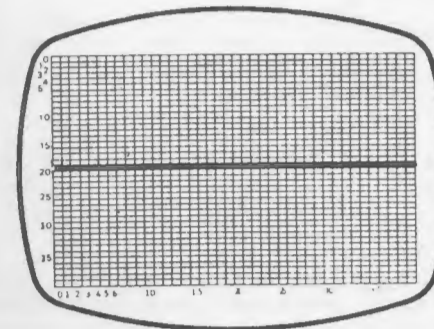
crever 40 comandos:

```
COLOR = 4
PLOT 0,20
PLOT 1,20
PLOT 2,20 e assim sucessivamente, até ...
...
PLOT 39,20
```

O método mais simples é este. Apenas escreva:

```
COLOR = 4
HLIN 0,39 AT 20
```

Pressione a tecla RETURN, então você terá a linha horizontal da coluna 0 a 39, ao nível do quadrado 20. Compare com a figura abaixo.



Agora tente colocar uma linha magenta desde a coluna 19 até a 28, no nível do quadrado 18. Tente outras também.

Faça pelo menos 6 linhas horizontais diferentes para treinar.

Note que quando você coloca um ponto ou uma linha colorida sobre alguma linha ou ponto já existente, a cor antiga desaparece dando lugar à nova. Para limpar totalmente a tela use o comando GR.

Há também uma maneira similar de desenhar linhas verticais na tela. Para desenhar uma linha vertical laranja na coluna 7 do quadrado 12 ao 33, escrevemos:

```
COLOR = 9
VLIN 12,33 AT 7
```

Experimente este comando.

Pratique desenhando várias linhas verticais. Você pode também testar a sua habilidade com as linhas verticais e horizontais simultaneamente desenhando uma borda magenta em torno da tela, apenas com cinco instruções.

Depois ponha um "x" verde na tela. Tente desenhar algumas linhas com a cor 0. Brinque com PLOT, HLIN e VLIN antes de passar adiante.

OS CONTROLES (JOYSTICK)

Antes de iniciarmos, verifique se os controladores foram ligados conforme instruções do Capítulo 1.

Digite o comando que "liga" um dos controladores de jogos:

```
PRINT PDL(0)
```

Um número deverá aparecer. Mova os controladores e verifique qual deles atua. Marque o número zero nesse controlador. Agora escreva novamente:

```
PRINT PDL(0)
```

Experimente ficar mudando o controle e escrevendo PRINT PDL(0). Verifique quais os maiores e os menores números que você pode obter e qual a maior e menor mudança de posição (números) possível.

Você pode descobrir a posição do outro controle digitando: PDL(1). O comando "PDL" vem do inglês "PADDLE" (raquete), uma vez que os controles são frequentemente usados como raquetes,

nos jogos. Esses controles também são usados de várias outras maneiras.

PDL é uma função. A função em linguagem, é uma instrução que usa um ou mais números e executa uma operação sobre eles fornecendo um único valor. Os números que a função usa são chamados de argumentos e são sempre colocados entre parênteses após o nome da função. PDL é uma função que tem apenas um argumento. O número que a função encontra deve retornar ao programa.

ARQUIVOS E OUTRAS POSSIBILIDADES DE CÁLCULO

Em inúmeras calculadoras, você pode guardar um número para mais tarde referir-se a ele ou até mesmo usá-lo. Para isto, você coloca o número num lugar que chamaremos, por ora, arquivo.

Normalmente isso é feito pressionando a tecla "M" (memória). No computador você pode fazer a mesma coisa. Por exemplo, para guardar o valor 13, escreva:

```
M = 13
```

O valor 13 não é impresso, mas é introduzido no arquivo chamado M. Se agora você escrever:

```
PRINT M
```

O computador irá imprimir o valor de M. Experimente escrever os dois comandos.

Agora, escreva:

```
M = 333
```

e imprima o valor de M. É 333, certo? O que aconteceu com o 13? Ele se foi para sempre. O arquivo só pode guardar um valor de cada vez. Quando você coloca um novo valor em M, o valor antigo é apagado.

Escreva:

```
PRINT "M"
```

O que acontece? Há uma grande diferença entre:

M e "M"

É como a diferença entre essas duas instruções:

GATO TEM QUATRO PATAS

"GATO" TEM QUATRO LETRAS

No primeiro caso estamos nos referindo ao animal GATO. E no segundo estamos nos referindo à palavra GATO. Assim que as aspas são usadas em computadores. Quando nós digitamos:

PRINT "M"

nós queremos imprimir a letra "M". Quando digitamos:

PRINT M

nós queremos imprimir o conteúdo da letra M.

Você pode guardar o resultado de uma operação aritmética num arquivo. Por exemplo:

M = 4 + 5

Você pode verificar que o resultado foi guardado mandando imprimir o valor de M.

Você também pode usar o valor de M em uma operação posterior. Por exemplo:

PRINT M = 2

A resposta é a esperada? Experimente outros cálculos usando M.

Uma calculadora simples tem apenas um arquivo; computadores têm centenas de arquivos (o seu computador tem 936). Utilizaremos o termo variável para designar um arquivo, apesar delas não se comportarem como variáveis matemáticas. Elas são

muito mais simples, representando apenas o lugar onde o valor foi guardado. O valor de uma variável será 0, até que nela seja inserido algum outro valor. Veja que isso contraria a matemática que você aprendeu, porém será a nossa convenção.

O arquivo, ou variável, pode ter quase todos os nomes que você desejar, desde que comece por uma letra. Por exemplo:

SOMA = 56 + 34 + 1523 + 8

BOLA = 45

JOGADA = 9

Alguns nomes não são permitidos. São os que incluem alguma palavra que tenha um significado especial para o computador (palavras reservadas). Uma destas palavras é "COLOR". Então, nenhuma variável pode ter um nome que inclua a palavra "COLOR". Experimente escrever:

INCOLOR = 6

ou

TRICOLOR = 9

Tudo o que você obtém é uma mensagem de erro. Sempre que o nome de uma variável provocar a mensagem ?SINTAX ERROR, significa que você introduziu uma das palavras reservadas no nome. Não se preocupe. Escolha um outro nome.

A lista das palavras reservadas que não podem ser usadas como variáveis, pode ser encontrada no APÊNDICE, no fim deste manual.

Quando você estiver escolhendo nomes, tente fazê-los exprimir o significado que você deseja dar à variável. Isto tornará os nomes mais fáceis de serem lembrados.

Experimente escrever

AVE = 11

e então

PRINT AVE

Você obteve o que esperava? Agora escreva:

```
PRINT AVEIA
```

O que aconteceu? Experimente agora:

```
PRINT AVO'
```

```
e
```

```
PRINT AVES
```

Se você observar os nomes, notará que todos começam por "AV". O computador usa apenas os dois primeiros caracteres de cada nome de variável para distinguir uma variável da outra. Então o nome:

```
AVE
```

Se refere a mesma variável que:

```
AVEIA
```

```
e
```

```
AVO' e assim por diante.
```

Aqui temos uma regra útil. Digamos que você tenha algum valor na variável CUSTO e quer incrementar este valor em 5 unidades. Uma das maneiras de fazer isto é imprimir o valor de "CUSTO", somar 5 a este valor e finalmente inserir o resultado na variável CUSTO, assim:

```
CUSTO = 28
```

```
PRINT CUSTO
```

```
PRINT 28 + 5
```

```
CUSTO = 33
```

Outra forma bem mais fácil de escrever a mesma operação é:

```
CUSTO = 28
```

```
CUSTO = CUSTO + 5
```

Experimente agora escrever os seguintes comandos, em ordem:

```
CUSTO = 2
```

```
PRINT CUSTO
```

```
CUSTO = CUSTO + 3
```

```
PRINT CUSTO
```

```
CUSTO = CUSTO * 6
```

```
PRINT CUSTO
```

```
CUSTO = CUSTO / 10
```

```
PRINT CUSTO
```

Ao fim destes comandos, você provavelmente obterá o valor 3. Está correto?

Experimente a sequência:

```
MACAS = 55
```

```
BANANAS = 11
```

```
QUOCIENTE = MACAS / BANANAS
```

```
PRINT QUOCIENTE
```

Primeiro pense na resposta, depois veja se está correta.

Experimente agora os comandos:

```
HOJE = 128
```

```
PRINT = "HOJE"
```

```
HOJE = HOJE/2
```

```
PRINT "HOJE"
```

```
HOJE = HOJE/2
```

```
PRINT HOJE
```

O que você esperava? O que obteve?

PRIORIDADE, OU QUEM VEM PRIMEIRO?

Em certos banquetes formais, as pessoas são servidas de acordo com um planejamento rigoroso: primeiro o convidado de honra, depois as mulheres, depois os homens e finalmente o anfitrião. Não importa onde eles estão sentados, o garçom anda e escolhe quem deve ser servido primeiro. Nesse caso, podemos dizer que há uma certa prioridade entre os participantes do jantar. Num cálculo simples:

```
PRINT 4 + 8 / 2
```

Você não pode dizer se a resposta deve ser 6 ou 8, até que saiba qual é a ordem (ou prioridade) das operações aritméticas. Se você primeiro somar 4 com 8, obtém 12. Se então dividir 12 por 2 a resposta será 6. Esta é uma resposta possível. No entanto, se você somar 4 com 8 dividido por 2, a resposta será 8. Esta é outra resposta possível. Oito é a resposta que seu computador dará. Aqui está como o computador escolhe a ordem das operações:

1. Quando o sinal de menos é usado para indicar negativo, por exemplo:

```
-3 + 2
```

O computador primeiro levará em conta o sinal de menos da variável. Então $-3 + 2$ resulta -1 . Se o computador efetuasse a adição primeiro, $-3 + 2$, resultaria em -5 . Mas isto não ocorre. Um outro exemplo é:

```
JOSE' = 6  
PRINT -JOSE'+10
```

A resposta é 4. Confere?

2. Após os números negativos, o computador executa a potenciação. A expressão:

```
4 + 3 ^ 2
```

é efetuada elevando 3 ao quadrado (3 vezes 3 = 9) e, depois, somando 4, dando um total de 13. Quando há várias potências, elas são efetuadas da esquerda para a direita; então:

```
2 - 3 ^ 2
```

é calculada pela multiplicação de 2 por ele mesmo três vezes ($2*2*2$) = 8 e então multiplicando este valor por ele

mesmo. A resposta é 64.

3. Depois que todas as potências tiverem sido calculadas, todas as multiplicações e divisões são feitas da esquerda para a direita. Operadores aritméticos de mesma prioridade são sempre calculados da esquerda para a direita. Multiplicação (*) e divisão (/) têm a mesma prioridade.

4. Por último, todas as adições e subtrações são feitas, da esquerda para a direita. Adição (+) e subtração (-) têm a mesma prioridade.

Vamos fazer um resumo das prioridades das operações aritméticas no computador:

PRIMEIRO: - (usado para indicar números negativos)
SEGUNDO: ^ (potenciação, da esquerda para a direita)
TERCEIRO: * e / (multiplicações e divisões, da esquerda para a direita)
QUARTO: + e - (adições e subtrações, da esquerda para a direita)

A seguir, você encontrará algumas expressões aritméticas para calcular. Execute-as, fazendo a conta fora do computador (na cabeça, no papel ou na sua calculadora) e depois calcule as nele. Se a sua resposta não conferir com a do computador, procure descobrir porque. Nós aqui, só apresentaremos as expressões. Você terá que colocar um PRINT na frente de cada uma para saber a resposta do computador.

A menos que você tenha muita experiência com cálculos em computadores é aconselhável que realmente faça estes exemplos. Primeiro faça a conta a mão e confira seu resultado com o computador. Faça isso, conta por conta. Não tente resolvê-las todas de uma só vez e depois compará-las com o computador. Verifique no que errou, antes de passar para a seguinte.

```
3 + 5  
8 + 6 - 3 + 2  
8 * 5  
3 - 2 + 4  
12 / 8 + 2  
6 - 5 / 2  
4 / 2 - 2  
6 * -2 + 6 / 3 + 8
```

$4 + -2$
 $2 \wedge 2 \wedge 3 + 1$
 $2 * 2 * 3 + 1$
 $2 * 2 + 1 * 3$
 $8 / 2 / 2 / 1$
 $8 * 2 / 2 + 3 * 2 \wedge 2 * 1$
 $20 / 2 * 5$

Nenhuma resposta é dada neste manual. O seu computador lhe dará as respostas corretas.

COMO EVITAR AS PRIORIDADES

Suponhamos que você queira dividir 12 por $4 + 2$. Se você escrever:

$$12 / 4 + 2$$

Você obterá 12 dividido por 4, e depois somado com dois. Mas isto não é o que você queria. Para conseguir o que deseja, você pode escrever:

$$12 / (4+2)$$

Os parênteses modificam a precedência. A regra que o computador segue é simples: faça primeiro o que está entre parênteses. Se houver parênteses dentro de parênteses, faça a operação dos internos primeiro. Eis aqui um exemplo:

$$12 / (3 + (1+2)^2)$$

Neste caso, fazendo os parênteses internos, você primeiro soma 1 com 2. Agora a expressão fica efetivamente:

$$12 / (3+3^2)$$

Mas você sabe que $3+3^2$ é $3+9$ ou 12, ficando então a nossa simplificada para $12/12$, cujo resultado é 1.

Num caso como $(9-4)*(1+2)$, quando há mais de um jogo de parênteses, não agrupados um dentro do outro, você simplesmente trabalha da esquerda para a direita. Esta expressão se torna $13*3$, ou 39.

Aqui temos algumas expressões para calcular. Lembre-se, é muito importante que você realmente faça estas operações. Estas regras de prioridade e parênteses são válidas para quase todos os sistemas de computadores do mundo, não apenas para o seu computador.

$$\begin{aligned}
 &44 / (2+2) \\
 &(44/2) + 2 \\
 &3 + (-2*2) \\
 &(3+-2)*2 \\
 &100 / (200 / (1*(9-5))) \\
 &32 / (1 + (7/3) + (5/4))
 \end{aligned}$$

CAPÍTULO

3

PROGRAMAÇÃO ELEMENTAR

- . Execução adiada
- . Edições elementares
- . Acrobacias elementares
- . Algumas outras coisas que tornam a programação mais simples
- . Eliminando ou copiando o que foi escrito
- . Comentários sobre o aprendizado
- . Evitando acidentes
- . A comparação e a verdade
- . Ordem ou prioridade para operações
- . O comando IF
- . Guardando programas no disco
- . Guardando programas em um "cassette"
- . Outros programas gráficos
- . "Loops" dos comandos: FOR e NEXT
- . Um exemplo de programa errado
- . Mais um exemplo de "loops" internos
- . Ganhando brilho
- . Imprimindo com detalhes

EXECUÇÃO ADIADA

Até aqui, quando você escreveu

```
PRINT 3 + 4
```

e pressionou RETURN, o computador fez o que foi mandado fazer, imediatamente. Quando um computador age de acordo com o comando que você emitiu, significa que ele entendeu que tem de executar aquele comando. Este caso é conhecido como EXECUÇÃO IMEDIATA dos comandos a ele enviados.

Você agora vai aprender como guardar comandos para serem executados posteriormente (EXECUÇÃO ADIADA). Para garantir que a memória do computador não contenha programas anteriores, informe-a que você está entrando com novas instruções, escrevendo:

```
NEW
```

Como tudo o que já foi visto, NEW tem que ser seguido do RETURN. Para mandar o computador guardar um comando, apenas escreva um número antes do comando. Por exemplo, se você escrever:

```
100 PRINT 3+4
```

nada parece acontecer, mesmo que você tecle RETURN. O computador guardou o comando. Para comprovar se o comando foi guardado, escreva a instrução:

```
LIST
```

Experimente. Se o comando foi digitado corretamente (não apareceu a mensagem ?SINTAX ERROR), aparecerá no vídeo o seguinte:

```
100 PRINT 3+4
```

Agora mande executar a instrução, escrevendo o comando:

```
RUN
```

e a resposta 7 aparece na tela.

Ao teclar RUN a instrução guardada é executada e armazenada para ser rodada quantas vezes você quiser. Experimente digitar RUN novamente.

Além disso, seu computador não esquece a instrução guardada mesmo que você limpe a tela. Aqui temos um novo modo de limpar a tela:

HOME

O comando HOME tem o mesmo efeito de:

SHIFT

ESC P

que você aprendeu anteriormente, só que ele pode ser usado em execução adiada tão bem como em execução imediata. Experimente isto escrevendo:

90 HOME

Agora quando você digitar RUN o computador executará a instrução anterior, limpando a tela antes de exibir o resultado.

Digite agora esses dois comandos:

NEW e depois,

LIST e veja o que acontece.

Ao teclar NEW, a instrução armazenada é perdida permanentemente. Escreva:

RUN

e nada aparece na tela. Isto se deve ao fato da instrução antiga ter sido eliminada pelo comando NEW.

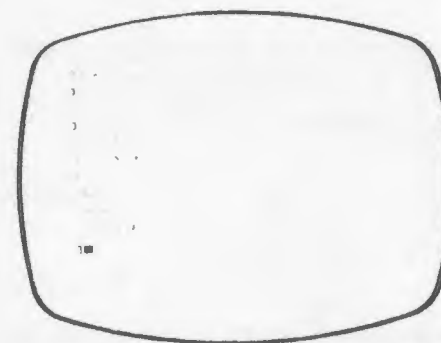
É possível guardar várias instruções, dando a cada uma delas um número diferente. Experimente escrever isto:

```
1 PRINT "ALO"
2 PRINT 4^5
3 PRINT 67/12
```

Nada aconteceu até aqui. Agora escreva:

RUN

e observe as respostas aparecerem.



Os números que pusemos na frente das instruções, para dizer ao computador para guardá-las, são chamados números de linhas. O computador armazena e executa as instruções na ordem crescente dos números de linha. Para verificar isso, apague as instruções anteriores, escrevendo:

NEW

e depois escreva estes comandos:

```
1 PRINT "P"
0 PRINT "A"
3 PRINT "E"
2 PRINT "L"
```

Note que 0 é um número de linha permitido. O maior número de linha que você pode usar é 63999. Agora rode estas instruções com o comando RUN. O resultado deverá ser este:

A
P
L
E

Para ver o que aconteceu dentro do computador, digite:

LIST

Note que você não precisa listar as instruções antes de rodá-las, mas é uma boa idéia fazê-lo.

O conjunto de instruções que são executados quando você tecla RUN são chamados de programa.

O programa deveria imprimir:

A
P
P
L
E

Mas parece que um comando PRINT foi esquecido. Como você pode inseri-lo no programa? A única maneira é reescrever as instruções da linha 4 e colocar uma nova linha 5. Para fazer as correções, tecla isto:

```
2 PRINT "P"
3 PRINT "L"
4 PRINT "E"
```

Para verificar o que ocorreu, liste o programa, utilizando o comando LIST.

Note que qualquer que seja a ordem em que os comandos são inseridos, o computador guarda-os em ordem crescente de número da linha. Agora rode este programa, utilizando o comando RUN.

Não foi muito agradável ter que reescrever os comandos para inserir um outro no meio. É, no entanto, usual em programação não escrever programas com linhas consecutivas, deixando espaço antes da primeira linha. Escreva:

NEW

para eliminar o outro programa e colocar este:

```
100 PRINT "G"
110 PRINT "T"
120 PRINT "O"
```

Quando você rodar este programa ele não imprimirá a palavra GATO verticalmente. Mas agora você pode voltar atrás e escrever:

```
105 PRINT "A"
```

Liste e rode este programa. Daqui por diante este manual irá começar todos os programas com números razoavelmente altos, e deixará bastante espaço entre as linhas consecutivas para que seja possível inserir instruções intermediárias.

EDIÇÕES ELEMENTARES

Cedo você descobriu que a instrução

```
PRINT PDL(0)
```

imprimia um número correspondente à posição de um dos controles manuais ("paddle"). Vários comandos PRINT foram necessários para descobrir muito sobre os controles. Agora que você pode escrever programas, tudo ficou mais fácil. Limpe a memória do computador com NEW e escreva:

```
100 PRINT PDL(0)
```

Agora cada vez que você escrever RUN, este curto programa é executado e você pode constatar a posição do controlador manual.

Este pequeno programa já está lhe poupando bastante trabalho. Antes, você tinha que reescrever todo o comando ou um grupo de comandos. Agora, você simplesmente reescreve:

RUN

Execução adiada tem ainda outra vantagem. Você pode modifi-

car parte de um programa e manter o resto como estava, sem ter que reescrever tudo. Por exemplo:

```
NEW
200 P=PDL(0)
210 PRINT P
220 PRINT "MOVA O CONTROLE"
230 PRINT "PARA UMA NOVA POSIÇÃO"
```

Agora rode este programa algumas vezes mudando a posição dos controles cada vez que rodá-lo. Verifique se o programa responde aos dois controles. Ele deverá funcionar somente para um deles. Você deve aproveitar e marcar este controlador com o número zero.

Este mesmo programa pode ser usado, com uma pequena mudança, para observar o outro controle. Liste o programa como ele está e escreva:

```
200 P=PDL(1)
```

Quando você escreve uma instrução com o mesmo número de linha de alguma já existente no programa, a nova linha se coloca no lugar da antiga. Rode o programa algumas vezes e veja o que acontece. Mova o outro controle entre os comandos RUN. Este programa responde aos dois controles? Marque um número 1 no controle a que o programa responde.

Modificar um programa desta maneira é um exemplo de Edição de um programa. Uma outra maneira é apagar as linhas que você não desejar mais. Se você desejasse eliminar a linha 230 do programa anterior, escreveria:

```
230 e depois RETURN
```

Você também poderia ter usado a instrução DEL. Para apagar a linha 230 do seu programa, você escreveria:

```
DEL 230,230
```

O comando DEL vem do inglês DElete (delete = apagar) e só é vantajoso quando você deseja apagar trechos de programas como no exemplo abaixo:

```
DEL 200,230
```

que apaga todos os comandos situados entre as linhas 200 e 230. Experimente estes comandos e liste o programa para verificar o que ocorreu. A possibilidade de apagar (DEL) trechos de programas será bastante útil quando estiver escrevendo programas longos.

Como você já viu, existem vários comandos que o ajudam a lidar com programas inteiros. São eles:

NEW

que apaga programas,

LIST

que lista os programas e

RUN

que executa os programas começando da instrução cujo número de linha é o menor. Também é possível começar a execução em outra linha ou listar apenas parte do programa. Estas possibilidades serão explicadas mais adiante.

ACROBACIAS ELEMENTARES

A partir desse ponto você irá começar a evoluir, esta seção irá discutir "loops", isto é, rotinas fechadas de programa - ção.

A melhor maneira de constatar como se comporta a função PDL é de compreender os "loops", e usar um comando que ainda não discutimos. Ele é muito simples. Escreva as linhas que se seguem (após escrever NEW, para apagar qualquer programa que ainda esteja na memória).

```
110 PRINT PDL(0)
120 GOTO 110
```

A linha 110 deste programa imprime o número que representa o atual valor do controle. A linha 120 faz o programa voltar para a linha 110. O que acontece então? O programa imprime o valor correspondente à posição do controle. Depois ele executa a linha 120 a qual manda executar a linha 110 novamente, e assim por diante. Isto é um "loop". O "loop" é uma estrutura

ra de programa que existe quando o programa inclui uma ordem para retornar a um comando executado previamente. Rode o programa. Mexa com o controle. Na próxima seção nós lhe diremos como parar este programa. Por enquanto, observe que em poucos segundos, centenas de números já apareceram na tela

Você deve ter notado como os números cintilam no vídeo, assim que um dos controladores é movimentado. Isso se deve pelo fato dos números serem impressos inicialmente na parte inferior da tela e assim que uma posição do controlador gerar um novo número, todos os outros são movidos para cima. Essa característica é conhecida em computação por "scrolling".

ALGUMAS OUTRAS COISAS QUE TORNAM A PROGRAMAÇÃO MAIS SIMPLES

Primeiro, antes de iniciarmos novo assunto, você deve estar imaginando como parar o programa dos controladores. Para parar o programa, simplesmente tecle:

```
CTRL
C
```

O comando CTRL C quando acionado, além de parar o programa, exibe no vídeo, o número da linha em que a execução foi interrompida. Por exemplo:

```
BREAK IN 110
```

A propósito, esta é uma exceção à regra de teclar RETURN após todos os comandos. Normalmente não há necessidade de teclar RETURN quando um programa é interrompido com CTRL C. Você também pode usar a tecla RESET para interromper programas; mas, com RESET você não obterá a mensagem que indica em que linha a execução do programa foi interrompida.

Quando você interrompe um programa com CTRL C ou RESET, você pode continuar a execução através da instrução:

```
CONT
```

que significa "CONTinuar". Experimente e depois digite este

programa:

```
NEW
100 X=PDL(0)
110 PRINT "CONTROLE Ø ESTA EM "
120 PRINT X
130 Y=PDL(1)
140 PRINT"CONTROLE UM ESTA EM "
150 PRINT Y
```

Anteriormente dissemos que quando você digita RUN, a execução do programa começa na linha cujo número é o menor. Entretanto, se você desejar que o programa comece a rodar a partir de outra linha, como por exemplo a 130, simplesmente escreva:

```
RUN 130
```

Você também pode especificar linhas no comando LIST. Se você escrever:

```
LIST 130
```

seu computador irá listar a linha cujo número é 130. Se agora você alterar o comando para:

```
LIST 110,130
```

seu computador irá listar todas as linhas do programa, desde a 110 até a 130. Para o comando RUN não é possível determinar apenas uma parte das linhas.

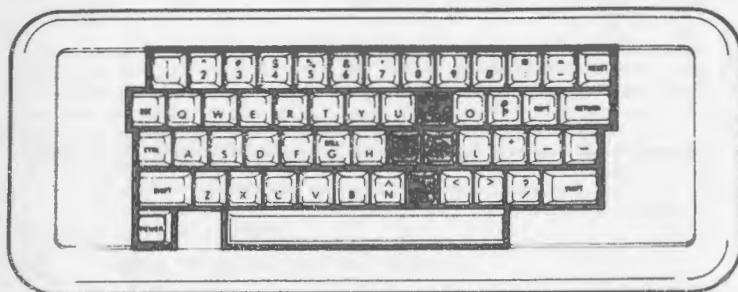
ELIMINANDO OU COPIANDO O QUE FOI ESCRITO

Quando o retrocesso ou as teclas de reescrever são pressionadas, o CURSOR é movimentado. Além de movimentar o cursor, elas também eliminam ou reescrevem caracteres, como você já aprendeu no Capítulo 2. Entretanto, é possível mover o CURSOR sem afetar nada, exceto a sua posição.

Cinco teclas são usadas para executar estes movimentos. Elas são:

```
ESC I J K M
```

Veja como usá-las:



Primeiro você coloca o computador em modo de edição, pressionando e soltando a tecla **ESC**, então usa **I** para mover o cursor para cima, **J** para movê-lo para a esquerda, **K** para movê-lo para a direita, **M** para movê-lo para baixo. Para mover o cursor repetidamente, pressiona uma das teclas indicativas da direção do cursor (**I**, **J**, **K**, ou **M**) ao mesmo tempo que pressiona a tecla **RPT**. O cursor irá andar rapidamente enquanto as duas teclas estiverem pressionadas. Se o cursor atingir a base do vídeo, ele irá parar, e a tela irá subir uma linha de cada vez. Se ele atingir o canto direito da tela, irá desaparecer, reaparecendo novamente no início da linha seguinte. Pratique movendo o cursor ao redor da tela com estas cinco teclas.

Quando você se cansar do simples movimento do cursor, pressione a barra de espaço e tudo voltará ao normal.

O simples movimento de cursor causado pelas teclas **J** e **K**, quando vistos na tela, são iguais aos movimentos causados pelas teclas retrocesso e reescrever, mas os efeitos são diferentes como você verá quando listar os resultados. O movimento simples do cursor não causa nenhuma modificação no texto sob o qual passa, enquanto que as teclas retrocesso e reescrever apagam ou reescrevem os caracteres sob os quais passam.

Estes movimentos simples do cursor têm uma aplicação, não tão simples. Por exemplo, escreva:

```
130 PRINT "TESTE DO MOVIMENTO"
140 PRINT "      DO CURSOR"
```

teclando **RETURN** após cada linha. O seu computador parece ter aceitado os comandos, mas quando você tentar rodar o programa obterá a mensagem:

```
?SINTAX ERROR IN 130
```

Para fazer a correção necessária você pode usar um artifício evitando ter de reescrever a linha toda. Primeiro liste o programa e tecle **ESC**. Agora tecle **I** tantas vezes quantas necessárias para mover o cursor até a linha incorreta. Pressione a tecla **J** para mover o cursor até o início da linha e use a tecla de reescrever para passar o cursor sobre todos os caracteres que precedam a letra **"U"** no comando **"PRINT"**. Tecle um **"I"** sobre o **"U"** e continue com a tecla até o final da linha. Então, tecle **RETURN** e liste o programa para verificar se a linha foi corrigida devidamente.

Quando você precisa corrigir um pedaço de linha que aparece em algum lugar da tela, o movimento simples do cursor, as teclas retrocesso + e reescrever +, se usadas corrigirão com maior rapidez. Se você perder alguns minutos trabalhando com estas teclas agora, você irá economizar muito tempo mais tarde.

A tecla retrocesso só funciona na linha que você está escrevendo. Se você escrever uma linha de programa e então executar um movimento simples de cursor antes de teclear **RETURN**, a linha de programa que você acabou de escrever (não os caracteres sob os quais o retrocesso passou) serão afetados pelos movimentos da tecla retrocesso. A tecla reescrever, no entanto, irá reescrever só o carácter sob o qual ela passou.

COMENTÁRIOS SOBRE O APRENDIZADO

Muitas vezes podem ocorrer dúvidas sobre a linguagem que não são respondidas diretamente neste manual. Uma delas poderia ser: No comando

```
PRINT "MANUAL"
```

você precisa dar um espaço após a palavra **PRINT**? Ao invés de lhe darmos uma resposta, nós recomendamos que você ligue o seu computador e experimente os dois modos. Normalmente, uma simples experiência responderá a sua pergunta e desde que você tenha experimentado sozinho, nós lembramos que foi muito

melhor do que se você tivesse apenas lido a solução.

EVITANDO ACIDENTES

Anteriormente nesse capítulo você aprendeu a apagar uma linha escrevendo o seu número e pressionando RETURN. Este é o método favorito para a introdução de erros no seu programa. Suponhamos que você queira eliminar a linha 1100 do seu programa, mas se enganou e escreveu:

```
110 RETURN
```

Infelizmente você acaba de eliminar a linha 110. Outro exemplo seria o de você querer consertar a linha 450, escreve 450, pensa melhor e decide não mudar mais a linha. Não tecla RETURN. Acione a tecla de retrocesso, passando sobre o 450, ou tecla o comando especial:

```
CRTL
```

```
X
```

Usando CTRL X você coloca uma barra no final da linha que você está escrevendo e será como se nunca tivesse escrito esta linha. Esse comando é conhecido por "esqueça esta linha".

A COMPARAÇÃO E A VERDADE

O seu computador pode distinguir o que é verdadeiro e o que não é. Para que isto seja possível, é necessário a introdução de alguns símbolos. O símbolo > significa "maior que". A afirmação $6 > 2$ (que é lida seis maior que dois) é certamente verdadeira. O computador usa o número 1 para indicar a verdade.

Se você digitar

```
PRINT 6 > 2
```

seu computador responderá com 1. A afirmação $55 > 78$ é falsa. Seu computador usa o número zero para indicar o que não é verdadeiro. Se você agora digitar:

```
PRINT 55 > 78
```

seu computador responderá com o número zero.

O símbolo < significa "menor que", e você pode fazer proposições usando-o também.

Abaixo temos os símbolos usados para fazer afirmações ou comparações:

```
> maior que
< menor que
= igual a
>= maior ou igual a
<= menor ou igual a
<> diferente de
```

Para escrever os símbolos de "maior ou igual a" e "menor ou igual a" no teclado do seu computador, você primeiro deve digitar < ou > e depois =. Para escrever o símbolo de "diferente de" você deve teclar < e depois > .

Pense sobre elas e teste para ver quais das afirmações abaixo são verdadeiras ou falsas

```
5 <> 5
6 > 2
8 > 8
8 <= 8
9534 - 4359
5 < 8
45 >= -4
-8 < -7
-2 >= -5
9 <> -9
```

As afirmações podem incluir variáveis e expressões, tão bem quanto números. Por exemplo, se introduzirmos no computador a instrução:

```
PRINT (45 * 6) <> (45 + 6)
```


nos imprimimos o valor 1, pois 270 não é igual a 51 (lembre-se que 1 indica que a afirmação é verdadeira). Você tem visto que o computador pode distinguir a verdade da falsidade em afirmações sobre números. Entretanto, uma afirmação como $LEITE > CERVEJA$ pode ser falsa ou verdadeira, dependendo do valor das duas variáveis $LEITE$ e $CERVEJA$. Se:

$LEITE = 5$

e

$CERVEJA = 9$

então a afirmação $LEITE > CERVEJA$ é falsa. Mas se:

$LEITE = -8$

e

$CERVEJA = -15$

então a mesma afirmação anterior passa a ser verdadeira.

As afirmações para o computador têm o valor numérico de "zero" ou "um". Elas podem ser utilizadas em expressões aritméticas no lugar dos "zeros" ou dos números "um". Por exemplo:

$PRINT \ 3+(4>2)$

terá como resposta 4. O comando:

$T=4<>3$

dá a T o valor 1, pois 4 é diferente de 3 e dessa forma a afirmação $4<>3$ assume o valor 1. O comando:

$FRIO=67=19$

parece bastante confuso a primeira vista, mas é facilmente entendido. Sendo 67 diferente de 19, conclui-se que a afirmação $67=19$ é falsa e tem o valor zero. Dessa forma, o comando aparentemente confuso, pode ser traduzido por:

$FRIO=0$

Como nós já vimos, o computador usa 1 para indicar verdadeiro e 0 para falso. Se algo não é verdadeiro, é falso. Para seu computador não existem "meias verdades". Experimento isto:

$PRINT \ NOT \ 1$
e depois experimente
 $PRINT \ NOT \ 0$

O computador entende que não verdadeiro é falso e não falso é verdadeiro.

É evidente que você pode usar expressões no lugar de "zeros" e "uns", como essa:

$PRINT \ NOT \ (45>3)$

A sentença:

$TRIANGULOS \ TEM \ TRES \ LADOS$

é verdadeira. E a sentença:

$ESTE \ MANUAL \ E \ BRASILEIRO$

também é verdadeira. Considere agora a junção das duas:

$TRIANGULOS \ TEM \ TRES \ LADOS$
 $E \ ESTE \ MANUAL \ E \ BRASILEIRO.$

Esta sentença é falsa ou verdadeira? É verdadeira. Considere esta outra:

$TRIANGULOS \ TEM \ OITO \ LADOS$
 $E \ ESTE \ MANUAL \ E \ RUSSO.$

Esta sentença também é falsa. Geralmente, quando você combina duas sentenças ou afirmações com a letra E, você obterá:

- Uma nova sentença verdadeira, se as duas originais o forem, ou,
- Uma nova sentença falsa se pelo menos uma das originais o for.

O computador sabe como determinar se uma afirmação contendo a palavra AND (E) é falsa ou verdadeira. Teste seu computador com as seguintes instruções:

```
PRINT 1 AND 1
PRINT 1 AND 0
PRINT 0 AND 1
PRINT 0 AND 0
PRINT (3>2) AND 0
PRINT (NOT 0) AND (4=5)
```

Esta sentença é falsa ou verdadeira?

```
UM TRIANGULO TEM TRES LADOS
OU ESTE LIVRO E EM LATIM.
```

É verdadeira. Um triângulo tem três lados, mesmo que o livro não seja em latim. Então a sentença como um todo é verdadeira. Em geral, quando você combina duas sentenças com a palavra "OU", você obterá:

- Uma nova sentença verdadeira se pelo menos uma das originais o for, ou,
- Uma nova sentença falsa se ambas as originais o forem.

O computador também pode determinar se uma afirmação contendo OR (OU) é falsa ou verdadeira. Experimente cada uma destas afirmações no seu computador, após pensar qual deverá ser a resposta:

```
PRINT 1 OR 1
PRINT 1 OR 0
PRINT 0 OR 1
PRINT 0 OR 0
PRINT (4<>6) OR (4=5)
PRINT 1 OR (0 AND 1)
PRINT ((3>4) OR (54<337)) AND (NOT 0)
```

AND, OR e NOT serão bastante úteis na próxima seção.

Você já descobriu que no comando:

```
PRINT 1 OR 0
```

o computador considera 1 como verdade e 0 como falsidade. Agora experimente:

```
PRINT 23 OR 0 e isto:
PRINT -247 AND 32707.61
```

Em afirmações, como essas, seu computador não considera como verdade apenas o número 1, mas qualquer outro que não seja 0. Entretanto, quando o computador nos dá o valor de uma afirmação, este valor sempre será 0 ou 1.

Enquanto você não tiver conhecimento das regras de prioridades dos comandos AND, OR e NOT, treine com exemplos onde contem parênteses.

ORDEM OU PRIORIDADE PARA OPERAÇÕES

1. Parênteses ()
2. NOT é valores negativos -
3. Exponenciação ^
4. Multiplicação *
5. Soma +
6. Comparações > < ; - ; <= ; <>
7. AND
8. OR

O COMANDO IF

Suponhamos que você queira imprimir números inteiros de 1 a 10, sendo um número em cada linha. Um método óbvio de fazer, isto é:

```
NEW
210 PRINT 1
220 PRINT 2
230 PRINT 3
... e assim por diante.
```

Mas dessa forma você iria necessitar de 10 comandos para 10 números e se você quisesse imprimir inteiros de 1 a 200 com este método, 200 comandos seriam necessários. Usando um "loop" que você já aprendeu, é possível imprimir inteiros de 1 em diante, com apenas 4 comandos:

```
NEW
200 N=1
210 PRINT N
220 N=N+1
230 GOTO 210
```

Há um meio de controlar quando o "loop" deve ser executado. O que você precisa é um comando que faça um GOTO se N for, por exemplo, menor do que 11, mas não execute o GOTO se N for maior que 11. A solução para essa execução é o comando IF. Se uma condição é satisfeita, o computador irá ignorar a instrução GOTO e executará a instrução da linha seguinte. Se não houver próxima linha, o programa parará.

Eis aqui um programa que conta de 1 a 10 e depois pára:

```
NEW
200 N=1
210 PRINT N
220 N=N+1
230 IF N<10 THEN GOTO 210
```

Geralmente o comando IF funciona assim:

IF (expressão aritmética) THEN (algum comando)

Primeiro a expressão aritmética é calculada. Se ela resulta em zero (falsa) todo o resto daquela linha de programa é ignorado, e o computador vai para a próxima linha. Se o resultado da expressão aritmética não for zero (verdadeira), o resto daquela linha de programa é executado.

O comando IF é muito poderoso, e ele aparecerá em quase todos os programas que você escrever. Para treinar, experimente este programa:

```
NEW
400 GR
410 LINHA = 1
420 COLOR = LINHA
430 HLIN 0,39 AT LINHA
440 LINHA = LINHA + 1
450 IF LINHA<16 THEN GOTO 420
```

GUARDANDO PROGRAMAS NO DISCO

(pule esta seção se você não está usando unidades de disco)

Neste ponto, talvez você deseje guardar alguns destes programas que vêm usando, na sua unidade de disco flexível ("diskette"). Simplesmente faça seu programa (não esquecendo na hora da digitação, de começar por NEW) e depois de terminado digite:

SAVE

seguido do nome que você quer usar quando se referir ao programa, e então, tecle RETURN. Por exemplo, se você quisesse guardar o programa acima dando a ele o nome EXEMPLO, você escreveria:

SAVE EXEMPLO

Uma vez que o programa está guardado, tecle:

CATALOG

seguido de RETURN, para ver o nome do seu programa listado com os outros, no "diskette". Você poderá então rodar (RUN) o seu programa chamado EXEMPLO, sempre que desejar, escrevendo:

RUN EXEMPLO

Experimente guardar (SAVE) algum programa que escolher. Se você acidentalmente errar o comando, provavelmente obterá a mensagem:

?SINTAX ERROR

Simplesmente reescreva o comando corretamente e tecle RETURN.

Algumas vezes é desejável colocar um programa que está gravado no disco, na memória do computador, sem ter que rodá-lo. Por exemplo, você deseja modificar o programa antes de rodá-lo (RUN). O comando LOAD é útil neste caso. Para usá-lo, simplesmente digite LOAD, seguido do nome do programa que você deseja. Imagine que você deseja carregar o programa chamado

EXEMPLO. Escreva:

LOAD EXEMPLO

Se você modificar o programa e então desejar rodar (RUN) a nova versão que está na memória do computador e não a que está gravada no disco, lembre-se de escrever somente:

RUN

Caso você esqueça e digite

RUN EXEMPLO

a versão antiga que estava guardada no disco será recarregada, apagando assim a nova versão da memória.

Você pode usar os comandos LOAD e SAVE para transferir programas de um "diskette" para outro, carregando (LOAD) a memória com o programa de um e guardando (SAVE) o programa na outra. Pratique usando os comandos LOAD e SAVE.

GUARDANDO PROGRAMAS EM UM "CASSETTE"

(Pule esta seção se você não estiver usando um gravador casete)

Para guardar (SAVE) um programa que você deseja usar mais tarde, numa fita cassete, primeiro coloque um cassete virgem no seu gravador e volte-a até o começo, onde o seu programa gravado será facilmente encontrado. No gravador segure a tecla PLAY enquanto abaixa a tecla RECORD. Ambas devem permanecer abaixadas. De volta ao teclado escreva:

SAVE

Quando você teclar RETURN, o cursor piscante irá desaparecer. Após 10 ou 15 segundos, o computador dará um "bip" indicando que a gravação terminou e o cursor reaparecerá. Aperte a tecla STOP no gravador, volte a fita para o começo e estará pronto para voltar à programação. O seu programa que estava no computador não foi afetado em nada pela gravação.

OUTROS PROGRAMAS GRÁFICOS

Anteriormente, você colocou quatro pontos coloridos nos cantos na tela. Agora escreva este programa:

```
NEW
190 GR
200 COLOR = 9
210 PLOT 0,0
220 PLOT 0,39
230 PLOT 39,39
240 PLOT 39,0
```

Liste (LIST) o programa para conferir se você o escreveu corretamente, e rode-o (RUN). Rápido, não é? Para mudar as cores, apenas mude a linha 200 e rode o programa novamente. Experimente listar o programa. Note que a listagem passa através da estreita janela de texto no fim da tela. Isto continuará a acontecer até que você escreva:

TEXT

para sair do modo gráfico e retornar a textos. Depois liste-o novamente.

O programa seguinte torna toda a tela de uma só cor.

```
NEW
200 GR
210 COLOR = 9
220 COLUNA = 0
230 VLIN 0,39 AT COLUNA
240 COLUNA = COLUNA + 1
250 IF COLUNA < 40 THEN GOTO 230
```

Aqui temos uma rápida explicação sobre o que acontece quando você roda (RUN) este programa. A linha 200 aciona o modo gráfico (GR). A cor é selecionada na linha 210. O programa deve começar na coluna 0 da tela, seguindo até a coluna 39. A linha 220 garante que o programa comece na coluna 0. Agora que a coluna 0 está preenchida com a cor desejada, a linha 240 incrementa a coluna em uma unidade. O valor da coluna é 1 agora.

A linha 250 verifica se o novo valor de COLUNA é menor do que 40. Se o for, o programa retorna para a linha 230 para desenhar uma nova linha vertical naquela coluna. No entanto, quan

do o valor de coluna atingir 40 (na tela, a máxima coluna é 39), o programa chegou ao fim.

Para eliminar a necessidade de escrever RUN toda vez que você desejar encher a tela de cor, escreva:

```
260 GOTO 210
```

Observe o que acontece. Quando este programa irá parar? Liste (LIST) o programa e certifique-se que você entendeu o que ele faz antes de prosseguir neste manual.

Quando você terminar de usar o programa de tela colorida, pare o computador (CTRL C), volte ao modo de texto (TEXT), limpe a tela (HOME) e experimente o programa que apresentamos a seguir. Ele usa uma instrução nova e muito importante: o comando REM. "REM" vem do inglês "REMark" e significa COMENTÁRIO. Este comando permite que você coloque comentários em programas. O computador ignora qualquer comando REM, eles são feitos exclusivamente para beneficiar o operador. Veja como é fácil acompanhar este programa onde REM (comentários) são usados:

```
200 REM ACIONAR MODO GRAFICO
210 GR
220 REM ESCOLHER A COR
230 COLOR = 1
240 REM LER CONTROLE ZERO
250 X=PDL(0)
260 REM DIVIDIR POR 7 PARA 0
    MAXIMO VALOR DE X SER 36
270 X=X/7
280 REM LER CONTROLE UM
290 Y=PDL(1)
300 REM LIMITAR A VARIAÇÃO PARA
    MANTER Y NA TELA TAMBEM
310 Y=Y/7
320 REM MARCAR O PONTO
330 PLOT X,Y
340 GOTO 250
```

A divisão por 7 é necessária uma vez que a função PDL nos dá valores entre 0 e 255, e a tela só aceita linhas e colunas com valores entre 0 e 39. A divisão por sete resulta em valores deste $(0/7)=0$ até $(255/7)=36.4285715$. O modo gráfico automaticamente arredonda valores de coordenadas para o inteiro mais próximo cujo valor seja menor ou igual ao valor dado. Em outras palavras, as coordenadas X e Y são arredondadas para baixo, para os inteiros de 0 a 36, podendo dessa forma ser marcadas (PLOT). Este método não utiliza a tela toda. Para

ra utilizar toda a largura da tela, no lugar de:

```
270 X=X/7
```

você pode usar essas duas linhas:

```
270 IF X>239 THEN X=239
275 X=X/6
```

para completar toda a altura da tela, você poderia fazer a mesma coisa usando a coordenada Y.

O comando IF limita o valor de X em 239. No modo gráfico de baixa resolução do computador, $239/6$ é arredondada de 39.83333 para 39. Este uso do comando IF para limitar a variação de uma variável é bastante comum.

"LOOPS" DOS COMANDOS: FOR E NEXT

Voltas, do tipo "loop", sejam elas executadas por aviões ou por programas de computadores, têm sempre um começo (topo) e um fim. No programa:

```
NEW
100 NUMERO=0
110 PRINT NUMERO
120 NUMERO=NUMERO+1
130 IF NUMERO <= 12 THEN GOTO 110
```

a linha 110 é o topo do "loop", e a 130 é o fim. O programa imprime os inteiros de 0 a 12 inclusive. O número 12 é o limite do "loop". Uma outra maneira de escrever um "loop" é usar um comando que nós ainda não discutimos: o comando FOR. Nós podemos usar este comando para reescrever o programa anterior:

```
200 FOR NUMERO=0 TO 12
210 PRINT NUMERO
220 NEXT NUMERO
```

Para rodar esse novo programa, use:

```
RUN 200
```

Se você apenas escrever RUN, o programa que começa na linha

100 (o menor número da linha na memória) será executado.

A linha 200 contém o novo comando FOR. Ele começa colocando em NÚMERO o valor 0. Este é exatamente o que a linha 100 faz. Então, a linha 210 é executada. O fim do "loop" está na linha 220. A variável NÚMERO é acrescida de 1 e então comparada com o limite superior especificado no comando FOR 12. Se NÚMERO não é superior ao limite, a execução continua no comando imediatamente seguinte ao FOR. Se a variável ultrapassar o limite, o programa sai do "loop" e vai para o comando após NEXT. Neste caso, o programa sai do "loop" e como não há linhas após o comando NEXT, ele termina.

A vantagem mais óbvia do método de construção de "loops" FOR/NEXT é que ele economiza um comando. Mas a mais importante é que ele não precisa pensar muito para escrever um "loop". Se você quisesse desenhar uma série de linhas horizontais na tela usando cada uma das 15 cores, você poderia escrever:

```
3000 GR
3010 FOR N=0 TO 15
3020 COLOR = N
3030 HLIN 0,39 AT N
3040 NEXT N
```

Uma outra vantagem é que é muito mais fácil ler um simples comando FOR do que procurar três comandos para visualizar o que um "loop" está fazendo. Para encontrar o fim de um "loop" FOR/NEXT, tudo o que você tem que fazer é procurar por um NEXT que tenha a mesma variável do FOR.

Talvez seja bom mencionar que, ao menos que você saiba como o comando FOR funciona, você não precisa usá-lo. Ele não acrescenta nenhum conhecimento a aqueles que você já tem. Ele apenas torna alguns programas mais fáceis de serem escritos. A opção é sua.

Neste ponto, se você tem seguido no seu computador, deverá remover uma parte do programa situada entre as linhas 3000 e 3040, inclusive, assim:

```
DEL 3000,3040
```

e então liste (LIST) o programa para conferir os resultados.

Para imprimir apenas os números pares de 0 a 12, você poderia usar o programa:

```
100 TAL = 0
110 PRINT TAL
120 TAL = TAL + 2
130 IF TAL <= 12 THEN GOTO 110
```

O segredo está na linha 120, onde 2 é adicionado ao TAL. Nós dizemos que o "loop" tem passo 2. Para ter um "loop" FOR com passo 2, escreva:

```
200 FOR TAL=0 TO 12 STEP 2
```

O resto do programa será como as linhas 210 a 220 da página anterior, exceto pelo NÚMERO que deverá ser mudado para TAL, sempre que aparecer 0. O passo (STEP) pode ser qualquer número dentro da faixa de trabalho de seu computador. O passo (STEP) também pode ser negativo, por exemplo:

```
200 FOR TAL=39 TO 15 STEP -3
```

Escreva esta linha e experimente digitando:

```
RUN 200
```

Pratique o comando FOR por uns tempos, se você deseja aprender a usá-lo. Numerosos exemplos daqui por diante usarão o comando FOR.

Juntamente com as conveniências do comando FOR temos algumas limitações. Por exemplo, "loops" FOR/NEXT talvez sejam interessantes, mas não devem se cruzar. Eis alguns exemplos que demonstram a idéia:

```
NEW
300 GR
310 FOR C=1 TO 15
320 COLOR = C
330 FOR LINHA=0 TO 39
340 HLIN 0,39 AT LINHA
350 NEXT LINHA
360 COLOR = C-1
370 FOR COLUMA=0 TO 39
380 VLIN 0,39 AT COLUMA
390 NEXT COLUMA
400 NEXT C
```

Este programa é um exemplo de dois níveis internos. Pense so

bre isto e rode (RUN) este programa antes de ir para o próximo. Lembre-se, quando estiver escrevendo programas usando comandos FOR, cada FOR deve ter um NEXT correspondente.

UM EXEMPLO DE PROGRAMA ERRADO

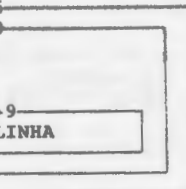
```
NEW
500 FOR N=10 TO 20
510 PRINT N
520 FOR J=30 TO 40
530 PRINT J
540 NEXT N
550 NEXT J
```



Este programa não funcionará. Seus "loops" estão cruzados e você receberá apenas uma mensagem de erro. Se você quiser cruzar "loops" de forma intencional, use comandos IF. Com o comando IF não haverá qualquer problema de cruzamento.

MAIS UM EXEMPLO DE "LOOPS" INTERNOS

```
NEW
300 GR
310 C=0
320 FOR COLUNA=0 TO 35 STEP 5
330 FOR TRAÇO=0 TO 30 STEP 10
340 C=C+1
350 IF C>15 THEN C=0
360 COLOR=C
370 FOR LINHA=TRAÇO TO TRAÇO+9
380 HLIN COLUNA,COLUNA+4 AT LINHA
390 NEXT LINHA
400 NEXT TRAÇO
410 NEXT COLUNA
```



Este programa utiliza três níveis de "loops" internos para desenhar um xadrez. Note que COLOR não pode ser usada como uma variável de um FOR/NEXT. COLOR é uma palavra reservada na linguagem BASIC que você está utilizando. Tente agora rodar o programa retirando a linha 300. O que acontece?

GANHANDO BRILHO

Se você está cansado de imprimir caracteres brancos ao fundo preto, gostará bastante desta seção. Digite:

I N V E R S E

e olhe para o colchete e o cursor no vídeo. O colchete deverá estar preto num fundo branco. Agora escreva um simples programa como:

```
NEW
100 PRINT "PRETO E BRANCO EM CORES"
```

e rode-o. Não é bem diferente do que você já estava acostumado? Agora digite:

F L A S H

e rode o programa novamente. Você verá a mesma frase "piscando".

Note que INVERSE e FLASH afetam somente a forma da saída do computador para o vídeo. Os caracteres que aparecem na tela continuam os mesmos, apenas com outra forma visual. Estes comandos podem ser usados tanto em execução imediata quanto em programas. Experimente-os. Após usar os comandos INVERSE e FLASH, se quiser retornar ao "branco no preto", digite:

N O R M A L

IMPRIMINDO COM DETALHES

Como referência, escreva este programa e veja o que ele faz quando você o rodar:

```
NEW
100 PRINT "OLA AMIGO"
110 GOTO 100
```

Páre o programa com CTRL C. Então acrescente na linha 100 uma vírgula:

```
100 PRINT "OLA AMIGO",
```

e rode o programa novamente. Como você vê, a adição desta permite que você imprima a palavra em colunas. Agora substitua a vírgula (,) pelo ponto e vírgula (;):

```
100 PRINT "OLA AMIGO";
```

e rode o programa novamente. Desta vez a saída é "empacotada"; isto é, não sobram espaços entre o que você mandou o computador imprimir. Ele imprime OLÁ AMIGO após OLÁ AMIGO, até que a tela esteja rapidamente completa.

Para parar a execução, digite:

```
CTRL  
C
```

Mude o programa acrescentando estes comandos:

```
90 V=99  
100 PRINT V
```

depois, rode-o (RUN). Agora mude a linha 100 para:

```
100 PRINT V,
```

e rode novamente. Então, mude a linha 100 para:

```
100 PRINT V;
```

e observe que tanto a vírgula (,) como o ponto e vírgula (;) também podem ser usados com valores numéricos. A habilidade de colocar números um após o outro sem espaços de separação muitas vezes é bastante útil. Vírgulas e ponto e vírgulas podem ser usados num comando PRINT.

Apague o programa antigo com o comando NEW, e digite:

```
100 TIROS=2  
110 BOLAS=3  
120 PRINT TIROS,BOLAS
```

Você pode tornar a saída mais clara incluindo mensagens no comando PRINT. Por exemplo, mude a linha 120 para:

```
120 PRINT "OS TIROS E AS BOLAS SAO ";  
TIROS,BOLAS
```

Note que um espaço foi deixado logo após SAO, para que o número de tiros não ficasse "grudado" à palavra. Outro modo, até preferível de se fazer a mesma coisa, é trocar a linha 120 por:

```
120 PRINT "OS TIROS E AS BOLAS SAO ";  
TIROS;" ";BOLAS
```

Nesta versão, um espaço representado pelas aspas foi deixado entre o número de tiros e as bolas. Entretanto, a melhor forma de se ordenar essa linha é a seguinte:

```
120 PRINT "TIROS ";TIROS;  
" BOLAS ";BOLAS
```

Essa nova forma da linha 120, mostrará no vídeo algo similar a um placar.

Digamos que você queira imprimir a palavra AQUI começando na décima coluna (a tela tem 40 colunas), você poderia usar a seguinte instrução:

```
120 PRINT" AQUI"
```

Deixamos 9 espaços em branco antes da palavra AQUI. Teria sido mais simples se tivéssemos usando o comando TAB. Como nas máquinas de escrever, você pode tabular no computador, da seguinte forma:

```
120 PRINT TAB(10)"AQUI"
```

Experimente, verifique como produz o mesmo efeito que colocar 9 espaços entre as aspas como nós fizemos acima.

Através da combinação de TAB e do "loop" FOR, você pode programar alguns bonitos efeitos visuais. Por exemplo:

```
NEW  
200 FOR N=1 TO 24  
210 PRINT TAB(N)"X"  
220 NEXT N
```


Nossa tela é composta por 24 (não 40) linhas horizontais de texto. Por isto, o limite do "loop" no programa anterior é 24.

TAB não pode ser usado para mover para trás (para a esquerda) numa linha. Somente movimentos para frente são levados em conta. Para imprimir em uma linha particular, você pode usar o comando VTAB. A linha na parte superior do vídeo é a 1 e a inferior, 24. VTAB, ao contrário do TAB, não é usada num comando PRINT.

Você pode tabular horizontalmente com o comando HTAB se não quiser usar o PRINT. HTAB funciona como TAB, exceto que não é usado num comando PRINT. Além disso, HTAB pode fazer a impressão começar à esquerda ou à direita da posição em que estava sendo feita. O caractere mais à esquerda em uma linha está na posição 1, enquanto que o mais à direita está na 40. A seguir apresentaremos um curto programa que demonstra o uso de HTAB e VTAB:

```
NEW
590 HOME
600 FOR X=1 TO 24
610 FOR Y=1 TO X
620 HTAB X
630 VTAB Y
640 PRINT "BANANA"
650 NEXT Y
660 NEXT X
670 GOTO 600
```

Antes de rodar este programa, experimente visualizar o que ele fará.

TAB funciona em execução imediata, mas você só pode usar VTAB e HTAB em programas. Os comandos TAB, HTAB e VTAB agem como as coordenadas no PLOT, entretanto, algumas diferenças devem ser consideradas. As 40 colunas para o comando TAB são numeradas de 1 a 40, como seriam numa máquina de escrever, enquanto que as coordenadas da instrução PLOT variam de 0 a 39, o que é bem mais conveniente para programas gráficos. Desde que caracteres são maiores do que "quadrados", nós construímos gráficos no lugar onde poderíamos imprimir 24 linhas na tela. Então os limites de VTAB são 1 e 24.

O zero ou um número que seja muito grande ou muito pequeno para TAB, VTAB ou HTAB, quando usados, darão a seguinte mensagem no vídeo:

?ILLEGAL QUANTITY ERROR

O maior valor de VTAB é 24, mas o maior valor para TAB e HTAB é 255. Ambos TAB e HTAB ultrapassam a dimensão da linha e pulam para a seguinte. Para ver isso em ação, digite:

```
NEW
300 FOR K=1 TO 255
310 PRINT TAB(K)K
320 NEXT K
```

Então experimente mudar as linhas 310 e 320 por:

```
310 HTABK
320 PRINT K
```

e adicione esta:

```
330 NEXT K
```

O que acontece ao programa quando você coloca VTAB no lugar de HTAB?

CAPÍTULO

4

EXECUÇÃO DE GRÁFICOS

- . Conversando com um programa
- . Fora das paredes
- . Criando sons
- . Barulho para a bola bate e volta
- . Notas mais altas e múltiplos comandos em uma linha
- . Notas aleatórias ("RANDOM")
- . Simulando um par de dados
- . Subrotinas
- . Verificações
- . Uma subrotina para desenhar um cavalo melhor
- . Gráfico de alta-resolução

CONVERSANDO COM UM PROGRAMA

Aqui temos um programa que faz um ponto colorido se mover através da tela, pulando de um lado para outro:

```

NEW
400 REM ESCOLHER A COR DA BOLA
420 BOLA=9
440 REM ACIONAR MODO GRAFICO
460 GR
480 REM POSICAO INICIAL
500 XANT=20
520 REM MOVER A BOLA PARA
    FRENTE E PARA TRAS
540 XMOV=1
560 REM NOVA POSICAO DE X
580 XNOVO = XANT + XMOV
600 REM A BOLA ESTA' NA TELA?
620 IF (XNOVO>= 0) AND
    (XNOVO< 40) THEN GOTO 720
640 REM MUDAR A DIRECAO DE XMOV
660 XMOV = -1*XMOV
680 GOTO 580
700 REM MARCAR A NOVA BOLA
720 COLOR = BOLA
740 PLOT XNOVO,20
760 REM APAGAR A BOLA ANTIGA
780 COLOR = 0
800 PLOT XANT,20
820 REM GUARDAR A NOVA BOLA
840 XANT = XNOVO
860 REM MOVER NOVAMENTE
880 GOTO 580

```

Nós sempre devemos nomear as variáveis com certa coerência. Nunca se esqueça de verificar se o nome da sua variável não contém nenhuma palavra reservada da linguagem. A razão pela qual a variável XMOV foi chamada de "XMOV" será evidente se você mudar o seu valor. Experimente XMOV=2, por ex. Se você colocar XMOV muito alto, a bola começará a pular violentamente através da tela, sem deixar nenhuma marca entre as posições.

Este tipo de programa é básico para muitos jogos de TV. É aconselhável gastar algum tempo estudando e fazendo mudanças no programa para verificar o que se pode fazer com ele. Seria uma boa idéia guardar (SAVE) este programa para você não ter que reescrevê-lo novamente se fizer alguma mudança fatal.

Quando você listar este programa, observará que ele não cabe inteiro na tela e as linhas de programa passam tão rapidamente que você não consegue ler facilmente. Uma das maneiras de

ver todas as linhas do programa é listar (LIST) o programa por partes. Por exemplo, você poderia escrever:

```
LIST 400,680
```

e só as linhas maiores que 400 e menores que 680 seriam mostradas. Então você poderia listar o resto do programa. Você também pode usar o comando

```
CTRL  
S
```

para interromper a listagem do programa. Experimente listar o programa e rapidamente escrever:

```
CTRL  
S
```

antes que as linhas deslizem para o topo da tela.

Para continuar a listagem, digite CTRL S novamente. CTRL C também pode ser usado para interromper a listagem. No entanto, o CTRL C impede que a listagem continue a partir do ponto em que estava.

Suponha que você não goste da cor da bola mostrada pelo seu programa e queira mudá-la. Uma forma seria a alteração da linha 420 toda vez que desejasse uma nova cor. É uma forma bem trabalhosa que pode ser simplificada se você fizer uma interação do programa com sua vontade. Para realização dessa interação, existe o comando INPUT. Mude a linha 420 para:

```
420 INPUT BOLA
```

Quando o programa executar este comando, uma interrogação (?) aparecerá na tela seguida do cursor piscando. O computador irá esperar até que alguém escreva um número e tecle RETURN. O número digitado será o valor de BOLA e o programa continuará a execução. Talvez seja uma boa idéia que o computador diga o que ele está esperando, na hora em que aparecer o ponto de interrogação. Você pode adicionar essas linhas ao programa:

```
280 REM ACIONAR MODO DE TEXTO
300 TEXT
320 PRINT "PARA SELECIONAR UMA
    COR PARA A BOLA,"
```

```
340 PRINT "ESCREVA UM NUMERO
    DE 1 A 15"
360 PRINT "APOS O PONTO DE
    INTERROGAÇÃO."
380 PRINT "ENTAO PRESSIONE A
    TECLA 'RETURN'."
```

Você também pode incorporar uma mensagem no comando INPUT:

```
420 INPUT "QUE COR DE BOLA
    VOCE DESEJA (1 A 15)?"; BOLA
```

Note que num comando INPUT a mensagem deve estar entre aspas e um ponto e vírgula (;) deve aparecer entre a mensagem e o nome da variável. Quando um comando INPUT contém uma mensagem, nenhuma interrogação aparece após o comando. Se você quiser que uma interrogação apareça, deve incluí-la na mensagem do INPUT.

Você poderá usar as teclas de retrocesso + e reescrever + para corrigir erros que cometer ao teclar o valor solicitado após o comando INPUT, mas lembre-se: se cometer erro e teclear RETURN, receberá uma mensagem de erro. Se o caractere inserido não for um número,

? REENTER

aparecerá na tela. Se for inserido um número muito grande ou muito pequeno, ou o programa deixará a bola se mover para o lado direito da tela e depois parará, ou a mensagem

ILLEGAL QUANTITY ERROR IN 720

aparecerá na tela e o programa parará. A maioria dos usuários não saberá como agir para recomeçar a utilização prática de seu programa. Por esta razão, você deverá fazer o programa verificar se todos os números escritos pelo usuário são corretos, dessa forma:

```
424 REM BOLA ENTRE 1 E 15
428 IF (BOLA >.0) AND (BOLA < 16)
    THEN GOTO 460
432 PRINT "ESTE VALOR NAO ESTA'
    ENTRE 1 E 15."
436 GOTO 420
```

Você está começando a perceber porque nós o avisamos para deixar bastante espaço entre uma linha e a outra?

É uma boa prática de programação fazer um programa tão "a prova de erros" quanto possível. Você avançou até o ponto em que está escrevendo mensagens de erro para outros lerem. É importante que os usuários de seu programa, na grande maioria leigos, nunca recebam mensagens de erro do tipo ?SINTAX ERROR, pois para eles não terá o menor sentido.

Toda vez que você usa um comando INPUT, seu programa deve verificar se o que o usuário escreveu está dentro de certos limites para evitar que o programa se perca ou falhe. O uso de sentenças em português claro e a verificação cuidadosa de tudo o que o usuário escreve são sempre necessários.

A propósito, você pode inserir vários valores em um único comando INPUT. O comando

```
3000 INPUT X,Y,Z
```

irá mostrar uma interrogação como de costume e então irá aguardar até que três números sejam inseridos. O primeiro número será inserido na variável X, o segundo na variável Y, e o terceiro na Z. Os três números podem ser separados por RETURN ou vírgulas, porém o último número deverá ser sempre seguido de um RETURN.

Guarde (SAVE) a sua melhor versão do programa BOLA SALTITANTE, só por precaução. Então, se você ainda não o fez, tente adicionar movimento vertical à bola. Use as novas variáveis Y1, Y2 e YMOV. A solução é dada a seguir, porém tente fazer isto sozinho antes de olhar.

Quando este programa estiver funcionando do jeito que você quer, guarde-o (SAVE) no seu disco ou gravador cassete. Nós o usaremos novamente mais tarde.

FORA DAS PAREDES

Aqui temos um jeito de fazer a bola saltar fora das quatro paredes. As instruções com um asterisco à sua frente, são aquelas que foram adicionadas ou alteradas no programa "BOLA SALTITANTE".

```
280 REM ACIONAR MODO DE TEXTO
300 TEXT
320 PRINT "PARA SELECIONAR UMA COR PARA A BOLA,"
340 PRINT "ESCREVA UM NUMERO DE 1 A 15"
```

```
360 PRINT "APOS O PONTO DE INTERROGACAO."
380 PRINT "ENTAO PRESSIONE A TECLA 'RETURN'."
400 INPUT "QUE COR VOCE DESEJA? "; BOLA
424 REM COR DA BOLA 1 A 15?
428 IF (BOLA > 0) AND (BOLA<16) THEN GOTO 460
432 PRINT "ESTE VALOR NAO ESTA' ENTRE 1 E 15."
436 GOTO 400
440 REM ACIONAR MODO GRAFICO
460 GR
480 REM POSICAO INICIAL
500 XANT = 20
* 510 YANT = 38
520 REM MOVER A VOLA PARA FRENTE E PARA TRAS
540 XMOV = 1
* 545 REM MOVER A BOLA PARA CIMA E PARA BAIXO
* 550 YMOV = 1
560 REM NOVA POSICAO DE X
580 XNOVO = XANT + XMOV
600 REM BOLA NA TELA?
620 IF (XNOVO >= 0) AND (XNOVO<40) THEN GOTO 686
640 REM MUDAR A DIRECAO DE XMOV
660 XMOV = -1*XMOV
680 GOTO 580
* 684 REM NOVA POSICAO DE Y
* 686 YNOVO = YANT + YMOV
* 689 REM A BOLA ESTA' NA TELA?
* 690 IF (YNOVO >= 0) AND (YNOVO<40) THEN GOTO 720
* 692 REM MOVER A BOLA PARA CIMA
* 694 YMOV = -1*YMOV
* 699 GOTO 686
700 REM MARCAR A NOVA BOLA
720 COLOR = BOLA
* 740 PLOT XNOVO,YNOVO
760 REM APAGAR A BOLA ANTIGA
780 COLOR = 0
* 800 PLOT XANT,YANT
820 REM GUARDAR A NOVA BOLA
840 XANT = XNOVO
* 850 YANT = YNOVO
860 REM MOVER NOVAMENTE
880 GOTO 580
```

Como você verá quando rodar (RUN) este programa, o resultado é um pouco repetitivo. Você pode alterar o "modelo de oscilação" mudando os valores iniciais de XANT e YANT (linhas 500 e 510), mas aqui temos uma mudança que talvez você prefira:

```
580 XNOVO = XANT+XMOV*PDL(0)/70
686 YNOVO = YANT+YMOV*PDL(1)/70
```

Para ver o que ele faz, mexa nos controladores de jogos.

Mais uma sugestão. Porque não ter um outro INPUT dando um valor à variável chamada FUNDO? Encha a tela uma vez com a cor FUNDO no começo do seu programa (após GR). Então para apagar a posição antiga da bola use:

```
780 COLOR = FUNDO
ou até mesmo
780 COLOR = FUNDO + 3
```

Arquive (SAVE) a sua versão favorita deste programa.

CRIANDO SONS

Clicks, ticks, tocks, e vários outros zumbidos são facilmente gerados. Você pode fazer sons no seu computador gerados por programas. Vá para um lugar quieto e tente trabalhar nesta seção.

Para conseguir produzir qualquer som no seu computador, você necessitará desta instrução:

```
150 SOM = PEEK (-16336)
```

Não há nenhuma explicação simples para esta instrução. O número -16336 é relacionado com o "endereço de memória" do auto-falante do computador e foi construído dentro da eletrônica do computador.

Você só vai ter que procurar este número quando precisar dele. O comando PEEK manda o computador ler alguma memória. Por exemplo: PEEK (15000) é uma instrução que manda o computador ler a memória 15000.

Na maioria das situações a utilização do comando PEEK somente nos fornece um valor numérico, mas algumas situações como PEEK-16336 resultam em alguma ação diferente de um simples número.

Toda vez que o programa executa este comando, o computador irá produzir um minúsculo "click". Rode o programa e escute bem de perto de seu computador.

Agora adicione esta linha e experimente seu novo programa:

```
160 GOTO 150
```

Para fazer o seu programa emitir "bips" por um limitado período de tempo, adicione os comandos:

```
140 FOR BIP = 1 TO 100
160 NEXT BIP
```

Um tom é gerado por uma rápida sequência de clicks. Qualquer programa que usa PEEK(-16336) repetidamente irá gerar algum tipo de barulho. Dado que -16336 é incômodo de se escrever, nós iremos inserir um novo comando que nos permitirá substituir -16336 por um símbolo mais fácil de ser escrito. Introduza o comando:

```
100 S = -16336
```

Para produzir um bonito "click" ressonante, mude a linha 150 para:

```
150 SOM=PEEK(S)-PEEK(S)+PEEK(S)-PEEK(S)+PEEK(S)-PEEK(S)
```

Diferentes números de PEEKs num comando produzirão diferentes qualidades de "clicks". Experimente rodar (RUN) algumas variações. Para obter zumbidos, ponha uma das variações num "loop". Geralmente, quanto mais rápido é o "loop", mais alto é o tom.

Agora, para usar estes sons, carregue (LOAD) o programa da bola bate e volta chamado FORA DAS PAREDES, na memória do seu computador. Experimente adicionar um som cada vez que a bola bater na parede.

Uma solução possível é dada na página seguinte, mas primeiro tente fazê-lo sozinho. (Obs: o som ocorre sempre que XMOV ou YMOV mudam de valor).

BARULHO PARA A BOLA BATE E VOLTA

Aqui temos um meio de tornar a bola audível. Adicione estas linhas ao programa FORA DAS PAREDES:

```

240 REM DAR A "S" O ENDEREÇO DO ALTO-FALANTE
260 S= -16336
663 REM SOM DE BATIDA
665 FOR B=1 TO 5
670 SOM=PEEK(S)-PEEK(S)+PEEK(S)-PEEK(S)
675 NEXT B
695 REM SOM DE BATIDA
696 FOR B=1 TO 5
697 SOM=PEEK(S)-PEEK(S)+PEEK(S)-PEEK(S)
698 NEXT B

```

Agora experimente os seus próprios sons. Por que não fazer um som diferente para cada parede?

NOTAS MAIS ALTAS E MÚLTIPLOS COMANDOS EM UMA LINHA

Para conseguir tons ainda mais altos, uma outra característica da linguagem pode ser introduzida. É possível colocar mais de um comando na mesma linha. Experimente esta linha de programa:

```

NEW
50 S=PEEK(-16336) : GOTO 50

```

Os dois pontos (:) podem ser usados para separar comandos em qualquer programa onde você desejar ter mais de um comando na mesma linha.

No entanto, só o primeiro comando da linha tem um número de comando, dessa forma, com um comando GOTO você vai sempre se referir a primeira instrução da linha.

Agora acrescente:

```

40 FOR PAUSA=1 TO 2500 : NEXT PAUSA

```

As vantagens de múltiplos comandos com um número de linha comum são as seguintes:

1. Os comandos são executados com maior rapidez. (Isso será uma vantagem somente se você necessitar de maior rapidez.
2. Mais linhas do seu programa cabem na tela.

3. Você pode escrever menos.
4. Você pode agrupar comandos que juntos executam uma função como a PAUSA na linha 40 acima.
5. Requer menos memória. (Isso será uma vantagem somente se você está ultrapassando o espaço de memória e o computador lhe dá a mensagem:

?OUT OF MEMORY

ou a mensagem

?PROGRAM TOO LARGE

enquanto você estiver inserindo o programa).

Há também algumas desvantagens:

1. A leitura do programa é mais difícil.
2. A correção ou modificação do programa também é mais difícil.
3. Você só pode se referir ao primeiro comando da linha.
4. É bastante desencorajador escrever um longo múltiplo comando para depois receber uma mensagem de erro como ?SIN-TAX ERROR quando o programa for rodado, sendo necessário, então, reescrever novamente toda a linha.

NOTAS ALEATÓRIAS ("RANDOM")

Experimente este curto programa no seu computador:

```

NEW
100 PRINT RND(1)
110 GOTO 100
RUN

```

O RND na linha 100 significa aleatório. A função RND fornece números aleatórios. Pare o programa com:

```

CTRL
C

```

Os números gerados por este programa são frações decimais a-

leatórias entre 0 (zero) e 1 (um). Mude a linha 100 para:

```
100 PRINT RND(0)
```

e rode. Pare o programa e compare o que você escreveu com o que está agora na tela. RND(0) fornece o último número aleatório que foi gerado.

Frações decimais aleatórias entre zero e um podem ser um pouco usuais. Frequentemente inteiros (números como 3, 6 e 10) são mais fáceis de se usar. Para obter inteiros aleatórios de 0 a 9 nós temos que adicionar algumas linhas ao programa. Digite:

```
NEW
90 REM DESIGNAR NUMEROS ALEATORIOS (RND) POR X
100 X = RND
110 REM MULTIPLICAR X POR 10
120 X = X * 10
130 REM ELIMINAR PARTE FRACIONARIA
140 X = INT (X)
150 PRINT X
160 GOTO 100
```

A linha 140 introduz a função INT. O comando INT(X) fornece o maior inteiro que é menor ou igual ao valor de X. Por exemplo, se o valor de X é 3.6754, então INT(X) é igual a 3. Os parênteses que se segue ao INT pode conter qualquer expressão aritmética ou variável numérica.

Agora rode (RUN) este programa. Funcionou como você esperava? Para fazer o programa gerar números de 1 a 10 no lugar de 0 a 9, simplesmente acrescente uma unidade ao valor de X através desta linha:

```
145 X = X + 1
```

O programa talvez pareça um pouco complicado à primeira vista. Para ver o que acontece passo a passo, você pode adicionar vários comandos PRINT. Modifique seu programa para que ele se pareça com este:

```
90 REM DESIGNAR NUMEROS ALEATORIOS POR X
100 X = RND(1) : PRINT "X=RND(1)",X
105 PRINT
110 REM MULTIPLICAR X POR 10
120 X=X*10 : PRINT "X=X*10 ",X
125 PRINT
```

```
130 REM CORTAR A FRACAO
140 X=INT(X) : PRINT "X=INT(X)",X
145 PRINT
150 REM ADICIONAR 1 AO VALOR DE X
160 X=X+1 : PRINT "X=X+1"
170 PRINT X
175 PRINT
180 FOR PAUSA=1 TO 2000 : NEXT PAUSA
190 GOTO 100
```

Rode este programa e veja o que ele faz.

Se você quer ser caprichoso, pode condensar este programa em apenas uma linha:

```
100 PRINT INT(10*RND(1))+1 : GOTO 100
```

Você sabe como essa linha 100 funciona?

SIMULANDO UM PAR DE DADOS

Você pode usar o que aprendeu sobre números aleatórios para escrever um programa que pretende simular um par de dados:

```
NEW
100 PRINT " DADO BRANCO ",
110 PRINT INT (6 * RND(1)) + 1
120 PRINT "DADO VERMELHO",
130 PRINT INT (6 * RND(1)) + 1
```

Este programa gera inteiros aleatórios de "um" a "seis" para cada dado. Para jogar de novo o dado, rode o programa novamente. Você consegue escrever um programa de jogos que use "dados"? Experimente.

Experimente escrever um programa de uma linha que gere números aleatórios de 1 a 50. De 0 a 25. Faça os seus próprios números. Lembre-se de acrescentar 1 ao número aleatório se não quiser gerar zeros.

Aqui temos um meio colorido de usar inteiros aleatórios:


```

NEW
210 REM ESCOLHER COR ALEATORIA
220 COLOR = INT (16 * RND(1))
230 REM ESCOLHER PONTO ALEATORIO
240 X = INT (40 * RND(1))
250 Y = INT (40 * RND(1))
260 REM MARCAR O PONTO ALEATORIO
270 PLOT X,Y
280 REM COMECAR NOVAMENTE
290 GOTO 220

```

Experimente usar RND em outros programas. Você consegue es-
crever um programa que desenhe linhas através da tela, em co-
res aleatórias?

SUBROTINAS

Imagine que exista um jogo que necessite de uma peça que se
pareça com um cavalo azul com patas cor de laranja e a cara
branca. Aqui temos um programa que desenha essa peça:

```

NEW
1000 REM PROGRAMA PARA DESENHAR UM CAVALO AZUL COM CARA
      BRANCA E PATAS ALARANJADAS
1010 GR
1020 COLOR=7 : REM AZUL CLARO
1030 PLOT 15,15
1040 HLIN 15,17 AT 16
1050 COLOR=9 : REM LARANJA
1060 PLOT 15,17
1070 PLOT 17,17
1080 COLOR=15 : REM BRANCO
1090 PLOT 14,15

```

Não há nada de errado neste programa; ele desenha um cavalo
azul com patas alaranjadas e cara branca. Agora, suponhamos
que você precise desenhar um outro cavalo em algum outro lu-
gar da tela. Você poderia reescrever este programa com novos
valores de X e Y, mas isto não seria agradável. Há um outro
meio de usar o mesmo programa para por a figura em qualquer
lugar da tela sem ter que reescrevê-la novamente.

A chave para fazer isto começa com a observação que você po-
de mover um ponto que está nas coordenadas (A,B) para a di-
reita adicionando algo ao valor da primeira coordenada (A,
neste caso).

Por exemplo, o ponto (4,17) se move 10 colunas para a direi-
ta se você somar 10 a primeira coordenada, criando o ponto
(14,17).

Analogamente, um ponto se move para a esquerda se você sub-
trair algo da primeira coordenada (ou adicionar um número ne-
gativo). Uma simples experiência mostrará que somando a, ou
subtraindo de, a segunda coordenada se moverá para cima e pa-
ra baixo respectivamente.

Com esses fatos em mente, você pode reescrever o seu progra-
ma para "centrar" o cavalo em quase todos os pontos X,Y da
tela. Por que "quase" todos os pontos? Porque se você esco-
lher um ponto central em alguma das laterais, o cavalo irá
sair da tela e isto talvez lhe dê a mensagem

?ILLEGAL QUANTITY ERROR IN 1030 (ou outro número de li-
nha)

Eis aqui um programa aperfeiçoado:

```

NEW
1000 REM COLOCAR UM CAVALO EM QUALQUER LUGAR DA TELA
1010 COLOR=7 : REM AZUL CLARO
1020 PLOT X,Y-1
1030 HLIN X,X+2 AT Y
1040 COLOR=9 : REM LARANJA
1050 PLOT X,Y+1
1060 PLOT X+2,Y+1
1070 COLOR=15 : REM BRANCO
1080 PLOT X-1,Y-1

```

Note que o GR foi deixado de fora. Nós queremos usar esta par-
te do programa para colocar vários cavalos na tela. Um GR
aqui iria limpar a tela antes que cada novo cavalo fosse de-
senhado.

Este programa não pode ser rodado assim como está. Primeiro
você precisa acionar o modo gráfico e escolher X e Y. Uma boa
primeira tentativa de se usar o programa cavalo talvez seja:

```

20 GR
30 REM CENTRAR PRIMEIRO CAVALO
40 X = 12
50 Y = 35

```

Se você tentar rodar este programa, um cavalo irá aparecer

no lugar desejado, mas o programa termina aí. Nós queremos colocar dois cavalos na tela. Você poderia digitar:

```
60
70 REM CENTRAR SEGUNDO CAVALO
80 X = 33
90 Y = 2
100
```

Execute novamente a parte do programa da linha 1000 e depois, termine.

Você sabe que o computador não pode ler aquelas estranhas instruções das linhas 60 e 100. Ele pode, no entanto, ler:

GOSUB 1000

Um programa como aquele começando na linha 1000 é chamado subrotina. GOSUB 1000 diz ao computador para executar a subrotina começando na linha 1000. Este comando também diz ao computador para voltar para linha seguinte ao GOSUB quando ele terminar de executar a subrotina. O computador sabe que a subrotina terminou quando ele encontra o comando RETURN. Para fazer a parte do seu programa que desenha um cavalo na tela ser uma completa subrotina, adicione a linha:

1090 RETURN

Agora rode (RUN) este programa. Você obterá a mensagem de erro:

?RETURN WITHOUT GOSUB IN 1090

mas de qualquer maneira, o programa parece funcionar perfeitamente. De fato, você acrescentou um novo comando à nossa linguagem: um comando de desenha um cavalo! Agora você pode usar o comando

GOSUB 1000

para desenha esse cavalo especial em qualquer posição X, Y que desejar.

VERIFICAÇÕES

A parte do programa da linha 1000 é chamada subrotina ou sub

programa. A parte do programa da linha 20 a linha 100 é chamada programa principal.

Para ver o programa rodar, ou o curso da execução, você pode se utilizar de um comando especial chamado TRACE. Esse comando especial pode lhe mostrar porque o computador deu uma mensagem de erro quando o programa de desenha um cavalo foi executado. Adicione esta linha ao programa especial:

10 TRACE

e, por ora, apague a linha 20. Ponha o computador em modo de TEXTO e rode o programa.

Os números que você vê na tela são os números de cada linha que é executada. Você pode observar que o programa começa na linha 10, continua através do programa principal até a chamada da subrotina, executa-a, volta ao programa principal, executa novamente a subrotina e, não encontrando nenhum número de linha menor vai para a linha 1000, executando a subrotina novamente. Aqui é que o problema ocorre. Você compreende agora a mensagem de erro?

Para remediar este problema, acrescente esta nova linha ao programa:

110 END

Quando o programa atingir a linha 110, ele simplesmente fará o que a linha manda: FIM (END). Rode o programa mais uma vez. Não há mais mensagem de erro. Como você acabou de ver, TRACE é muito útil quando você está tendo problemas com um programa. Se você quiser usar o comando TRACE em apenas parte do programa, use uma nova instrução denominada NOTRACE. Acrescente esta linha:

65 NOTRACE

e o programa será verificado somente até a execução da linha 65.

TRACE também pode ser usado em execução imediata. Simplesmente, escreva:

TRACE
RUN

e seu programa será verificado.

Uma vez que você introduziu o comando TRACE, seja em execução imediata ou como uma instrução do seu programa, o seu programa, será verificado toda vez que for rodado. Para parar o TRACE, você precisa introduzir o comando NOTRACE numa linha do seu programa ou em execução imediata.

UMA SUBROTINA PARA DESENHAR UM CAVALO MELHOR

As subrotinas devem ser escritas de forma a evitar que ocorram erros quando o programa é rodado. Um problema com a nossa subrotina de desenhar um cavalo é que alguns valores de X e de Y colocariam o cavalo fora da tela. Isso poderia ser evitado por um grupo de comandos como:

```
1012 IF X<1 THEN X=1
1014 IF X>37 THEN X=37
1016 IF Y<1 THEN Y=1
1018 IF Y>38 THEN Y=38
```

Como exercício, verifique por que o máximo valor de Y deve ser 38, enquanto X é limitado em 37.

Se houver qualquer tentativa de colocar o cavalo fora da tela, o cavalo será deslocado para a lateral mais próxima. Há outras maneiras possíveis, como dar uma mensagem de erro e parar o programa. Entretanto, a nossa escolha tem a vantagem de não parar o programa e você pode observar que algo está acontecendo.

Caso você queira mudar os valores em uma subrotina para diferentes instruções GOSUB, por exemplo, se o segundo jogador quiser colocar um cavalo de cor diferente, uma das maneiras de se fazer isto seria escrever toda a subrotina novamente com cores diferentes. No entanto, vamos experimentar usar variáveis no lugar de números. Na linha 1010, no lugar de COLOR=7, digite:

```
1010 COLOR=C : REM C=CORPO
```

Analogamente, você poderia escrever:

```
1040 COLOR=PE
1070 COLOR=CARA
```

O programa principal poderá ser assim:

```
20 GR
30 REM CAVALO DO PRIMEIRO JOGADOR
40 C=7 : REM AZUL CLARO : REM C=CORPO
50 PE=9 : REM LARANJA
60 CARA=15 : REM BRANCO
70 REM CENTRAR O CAVALO DO PRIMEIRO JOGADOR
80 X=15
90 Y=30
100 GOSUB 1000
```

e, assim por diante (certifique-se de colocar o comando END antes de tentar rodar o programa). Vários comandos têm que ser usados cada vez que você quiser um cavalo, mas assim mesmo são bem menos que se você tivesse que reescrever o programa todas as vezes. Para termos uma facilidade adicional na programação, um truque sutil é criar uma subrotina que designa as cores para o cavalo de cada jogador e que cada uma destas subrotinas chama a subrotina que desenha um cavalo.

```
2000 REM CAVALOS AZUIS COM PES LARANJA E CARA BRANCA
2010 C=7 : REM AZUL CLARO : REM C=CORPO
2020 PE = 9 : REM LARANJA
2030 CARA=15 : REM BRANCO
2040 GOSUB 1000
2050 RETURN
2500 REM CAVALOS LARANJAS COM PES COR DE ROSA E CARA VERDE
2510 C=9 : REM LARANJA:REM C=CORPO
2520 PE=11 : REM COR DE ROSA
2530 CARA=12 : REM VERDE
2540 GOSUB 1000
2550 RETURN
```

Agora tudo o que você precisa para por um cavalo azul com cara branca e pés laranja em (10,11) é:

```
30 REM CAVALO DO PRIMEIRO JOGADOR
40 X = 10
50 Y = 11
60 GOSUB 2000
```

Para colocar um cavalo laranja em (19,2) tudo o que você precisa é:

```
70 REM CAVALO DO SEGUNDO JOGADOR
80 X = 19
90 Y = 2
100 GOSUB 2500
```

As subrotinas de desenhar um cavalo que começam nas linhas

2000 e 2500, chamam uma outra subrotina que começa na linha 1000. As coisas se tornam bastante eficientes a este ponto. Uma vez que você escreveu uma boa subrotina que verifica erros e usa variáveis cujos valores são atribuídos no programa que for chamado (talvez seja o programa principal) ou uma outra subrotina, então você pode sobrepor outras subrotinas sobre ele. Isso torna os programas principais muito mais fáceis de serem escritos. Usando as três subrotinas, é muito fácil colocar na tela uma atrativa exibição de cavalos.

Mas primeiro, escreva uma outra rotina, útil:

```
1000 REM ESCOLHER M.Y. ALEATORIOS
1010 X=INT(RND(41)*137)+1
1020 Y=INT(RND(11)*10)+1
1030 RETURN
```

E agora o programa principal:

```
10 REM ACIONAR MODO GRAFICO
20 GR
30 REM ESCOLHER UM PONTO ALEATORIO
40 GOSUB 1000
50 REM POR UM CAVALO AZUL NAQUELE PONTO
60 GOSUB 2000
70 REM ESCOLHER PONTO ALEATORIO
80 GOSUB 3000
90 REM COLOCAR UM CAVALO LARANJA NESTE PONTO
100 GOSUB 2500
110 REM FAZER TUDO NOVAMENTE
120 GOTO 30
```

Assim é que deve se parecer um programa principal. Todo bom programador inclui vários REMs e GOSUBs. O trabalho deve ser feito em subrotinas relativamente curtas, cada uma delas fácil de ser escrita e, completa. Sinta-se livre para usar TRAC para verificar como este programa trabalha.

GRÁFICO DE ALTA RESOLUÇÃO

Nós chamamos o tipo de gráfico que você tem usado até então de gráficos de baixa-resolução. Nesta seção você vai aprender a lidar com um outro tipo de gráfico chamado gráfico de alta-resolução.

Este novo tipo de gráfico lhe dará condições de desenhar com

mais detalhes do que em baixa resolução. A nova tela de gráficos de alta-resolução é constituída de 280 por 160 divisões. As coordenadas horizontais começam em 0 à esquerda da tela e terminam em 279 à direita. Analogamente, as coordenadas verticais vão de 0 no topo da tela até 159 no final. Gráficos de alta-resolução não são difíceis de se entender. Frequentemente os comandos dos gráficos de alta-resolução são os mesmos que os correspondentes comandos dos de baixa resolução, exceto pela adição da letra H (alta-resolução = "high resolution"). Um verdadeiro conhecimento dos gráficos de baixa resolução será útil nesta seção.

Escreva:

HGR

Este comando limpa a tela, deixando quatro linhas no final para texto.

Assim como em gráficos de baixa-resolução, os gráficos de alta-resolução permitem o uso de coordenadas verticais na área de texto (o máximo é 192), mas estes pontos não são mostrados na tela. Se o cursor não estiver visível, tecle RETURN algumas vezes até ele aparecer no final da tela.

Neste novo tipo de gráfico há menos cores disponíveis. As cores de alta-resolução vão de 0 (preto) até 7 (branco). As cores são:

0 PRETO	1	4 PRETO	2
1 VERDE		5 LARANJA	
2 VIOLETA		6 AZUL	
3 BRANCO	1	7 BRANCO	2

Estas cores poderão variar de TV para TV de acordo com a sua posição na tela. Um ponto de alta-resolução marcado com a cor número 3, por exemplo, será AZUL se a coordenada horizontal for par, verde se a coordenada horizontal for ímpar, e branco somente se ambas coordenadas horizontais, par e ímpar, forem marcadas.

A única instrução para marcação de pontos em gráficos de alta-resolução é HPLLOT. Para experimentar, uma vez que você já inseriu o comando HGR, digite:

```
HCOLOR=3
HPLLOT 130,100
```


A última linha irá marcar um ponto branco de alta-resolução, na posição $X=130, Y=100$.

É mais fácil desenhar em gráficos de alta-resolução. Você simplesmente digita o comando HPLOT para um ponto da tela até (TO) outro ponto. Para desenhar uma linha no topo da tela, digite:

```
HPLOT 0,0 TO 279,0
```

Se você quiser desenhar uma linha do ponto 279,0 até o ponto 279,159 no canto inferior da tela, tudo o que você tem que fazer é escrever:

```
HPLOT TO 279,159
```

e a linha aparece no canto direito da tela. Quando você usa este último comando, a nova linha adota como ponto inicial o último ponto marcado na tela e como cor a deste último ponto (mesmo que você tenha introduzido um novo comando HCOLOR depois que aquele ponto foi marcado).

Você pode encadear estes comandos e desenhar várias linhas em uma só instrução se assim desejar.

Limpe a tela com HGR e experimente isto no seu computador:

```
HPLOT 0,0 TO 279,0 TO 279,159
TO 0,159 TO 0,0
```

Deverá aparecer uma linha ao redor da tela. Se isso não ocorrer, primeiro verifique se as suas instruções estão corretas. Se a linha ainda não estiver lá ou não for contínua, mude HCOLOR e experimente novamente. Algumas partes da tela só aparecem em certas cores em determinadas TVs.

Não é apenas fácil desenhar estas linhas no seu computador, como também desenhar diagonais em gráficos de alta-resolução é bastante simples. Para desenhar uma linha no canto superior esquerdo da tela até o canto inferior direito, simplesmente escreva:

```
HPLOT 0,0 TO 279,159
```

Pratique desenhando linhas de alta-resolução de vários tamanhos e cores.

Aqui temos um programa que torna o seu computador uma tela de alta-resolução:

```
NEW
200 HGR
210 HCOLOR=3
220 X = PDL(0)
230 Y = PDL(1)
240 IF Y>159 THEN Y=159
250 HPLOT X,Y
260 GOTO 220
```

A linha 240 é incluída porque a função PDL varia desde 0 até 255 e a coordenada Y sairia da tela se o seu valor fosse superior a 159. Rode (RUN) este programa.

Este programa funciona, mas ele seria melhor se fosse mais fácil desenhar uma linha sólida. Aquelas lacunas entre os pontos marcados não são sempre desejáveis. Você pode melhorar o programa escrevendo as seguintes linhas:

```
220 GOSUB 1000
230 HPLOT X,Y
240 GOSUB 1000
250 HPLOT TO X,Y
260 GOTO 240
1000 X= PDL(0)/.912
1010 Y= PDL(1)/1.6
1020 RETURN
```

Liste o programa e verifique cuidadosamente se você escreveu tudo corretamente. Eis aqui o que o programa faz:

O gráfico de alta-resolução (HGR) é acionado, a cor HCOLOR é definida e então o programa vai para a subrotina 1000. A subrotina determina o valor das coordenadas X e Y. Os controles

variam até o valor 255. Como gráficos de alta-resolução usam coordenadas horizontais variando de 0 até 279, o valor fornecido pelo PDL(0) é dividido por 0,913 para a sua variação varrer a tela toda. Analogamente, os valores fornecidos pelo PDL(1) são divididos por 1.6 para diminuir a sua variação para o valor das coordenadas verticais dos gráficos de alta-resolução: 0 a 159. Assim como em gráficos de baixa-resolução, a coordenada atualmente marcada é o inteiro mais próximo menor ou igual ao valor dado. A linha 1020 retorna ao programa principal e a 230 marca o ponto X,Y. Depois o programa volta para a subrotina, dá a X e Y valores novos e retorna para a linha 250 no programa principal onde uma linha muito curta é desenhada desde o ponto X,Y antigo até o ponto X,Y novo. O GOTO na linha 260 repete todo o programa exceto as instruções HGR e HCOLOR, buscando assim novos valores para X e Y a través da posição do controle e então marcando a nova posição X,Y. Use CTRL C para parar o programa.

Há uma razão para desenhar linhas no lugar de marcar cada ponto separadamente. Leva um certo tempo marcar um ponto e quando o computador marca um ponto de cada vez ele não pode sempre responder aos controles. Este é o motivo pelo qual há via espaços entre os pontos quando você movia os botões do controle rapidamente no primeiro programa. Desenhando uma linha pequena para cada nova posição especificada pelo botão do controle, podemos consertar isso: ao desenhar uma linha de um ponto a outro, todos os pontos intermediários são automaticamente marcados e muito mais rapidamente do que se fossem marcados um de cada vez. Guarde o programa (SAVE) e rode-o (RUN).

Aqui temos um programa que desenha na sua tela:

```

NEW
90 HOME
100 VTAB 24 : MOVER O CURSOR PARA ULTIMA LINHA
120 HGR : REM ACIONAR GRAFICOS DE ALTA-RESOLUCAO
140 A=RND(1)*279 : REM FIXAR "A" PARA CENTRO
160 B=RND(1)*159 : REM FIXAR "B" PARA CENTRO
180 N=INT(RND(1)*4)+2 : REM FIXAR O TAMANHO DE UM PASSO
200 HTAB 15 : PRINT "PASSO="; N;
220 FOR X=0 TO 278 STEP N : REM INCREMENTANDO OS
    VALORES COM O PASSO N
240 FOR S=0 TO 1 : REM 2 LINHAS DE X E X+1
260 HCOLOR=7*S : REM PRIMEIRA LINHA PRETA, PROXIMA
    BRANCA
280 REM DESENHAR UMA LINHA DO "CENTRO" ATE' O LADO
    OPOSTO
300 HPLOT X+S,0 TO A,B TO 279-X-S,159
320 NEXT S,X
340 FOR Y=0 TO 158 STEP N : REM INCREMENTOS COM PASSO N
360 FOR S=0 TO 1 : REM 2 LINHAS DE B E B+1

```

```

380 HCOLOR=7*S : REM PRIMEIRA LINHA PRETA, PROXIMA
    BRANCA
400 REM : DESENHAR UMA LINHA DO CENTRO ATE O LADO
    OPOSTO
420 HPLOT 279,Y+S TO A,B TO 0,159-Y-S
440 NEXT S,Y
460 FOR PAUSA=1 TO 1500 : NEXT PAUSA : REM ATRASO
480 GOTO 120

```

Este é um programa longo, escreva-o com cuidado e liste-o por partes (LIST 0,320 por exemplo) para verificar se tudo está correto. Quando você tiver certeza que está tudo correto, rode o programa.

Como você viu nas linhas 320 e 440, uma instrução NEXT pode conter variáveis de mais de um FOR.

Tome cuidado de colocar as variáveis no comando NEXT na ordem certa, para evitar loops cruzados.

Para voltar a programação, pare escrevendo:

```

CTRL
C          e depois
TEXT

```

Você pode pensar em maneiras de mudar este programa?

Após guardar (SAVE) esta versão na sua unidade de disco ou gravador cassette, experimente fazer o valor HCOLOR variar aleatoriamente. Experimente desenhar primeiro linhas laranja e depois azuis, ou somente linhas azuis.

CAPÍTULO

5

ALFANUMÉRICOS E MATRIZES

- . Dando sequência
- . Concatenação
- . Mais funções de alfanuméricos
- . Introdução aos argumentos (Matrizes)
- . Mensagens de erro dos agrupamentos
- . Conclusão

DANDO SEQUÊNCIA

Você gostaria de ver o seu nome soletrado de trás para frente? Até agora nós temos manipulado com gráficos e números, mas computadores também manipulam com letras e símbolos. O seu computador pode lidar com um único caracter, ou ele pode lidar com sequências de caracteres, de uma vez.

Variáveis que contenham sequências de caracteres, assim como as variáveis numéricas, recebem nomes próprios. Nomes de variáveis sequenciais seguem as mesmas regras dos nomes de variáveis numéricas, exceto que elas terminam com o símbolo do cifrão (\$). Aqui temos alguns exemplos de nomes de variáveis sequenciais:

```
NOME$
A$
FRASE$
```

A variável A é diferente da variável A\$, e ambas podem ser usadas no mesmo programa.

Se você desejar que a variável sequencial chamada NOME\$ (pronuncia-se: "NOME-cifrão") contenha as letras "MACHADO DE ASSIS", você pode escrever:

```
NOME$ = "MACHADO DE ASSIS"
```

Note que os caracteres que você coloca numa variável sequencial devem estar entre aspas. O comando:

```
PRINT NOME$
```

irá imprimir o conteúdo variável NOME\$; neste caso, o nome de um escritor famoso. Desta forma, quando você tem uma sequência de caracteres de uso frequente, guarde-a numa variável com um nome curto.

Na linguagem, existem muitas outras instruções que manipulam sequências. Suponhamos que você queira saber o comprimento de uma sequência (quantos caracteres ela contém). Você pode escrever:

```
PRINT LEN ("MACHADO DE ASSIS")
```

ou você pode escrever o comando equivalente:


```
PRINT LEN (NOME$)
```

o computador irá imprimir o "comprimento" da sequência. Neste caso, 16. Note que os espaços contam como caracteres.

Em computação, o termo sequência de caracteres é também conhecido como "string".

O número de caracteres em uma sequência deve variar entre 0 e 255. Se você tentar colocar mais de 255 caracteres, obterá a mensagem de erro:

```
?SINTAX ERROR
```

```
ou
```

```
?STRING TOO LONG ERROR
```

Uma sequência com 0 caracteres é chamada sequência nula.

Em algumas ocasiões você talvez deseje imprimir (PRINT) apenas parte do NOME\$. Para fazer isto você pode utilizar três funções úteis:

```
LEFT$, RIGHT$ e MID$
```

que significam esquerda, direita e meio, respectivamente.

Se por exemplo, você quiser imprimir as primeiras 7 letras de NOME\$ pode escrever:

```
PRINT LEFT$ (NOME$,7)
```

MACHADO, deverá aparecer na tela. Se você agora escrever:

```
PRINT RIGHT$ (NOME$,5)
```

ASSIS, aparecerá.

Para cada programa que for escrito usando variáveis sequenciais, deve ser determinado o valor da sequência dentro do programa. Cada vez que você rodar um programa, todas as variáveis numéricas são primeiro colocadas em 0 e todas as variáveis sequenciais são igualadas à nula.

Aqui temos um programa curto que usa as funções LEN e LEFT\$.

```
NEW
90 NOME$= "MACHADO DE ASSIS"
100 FOR N=1 TO LEN (NOME$)
110 PRINT LEFT$ (NOME$,N)
120 NEXT N
```

O comando RIGHT\$ é igual ao comando LEFT\$, exceto que ele usa os caracteres da sequência mais à direita.

Agora escreva um outro programa colocando RIGHT\$ no lugar de LEFT\$. Rode-o e veja o que acontece.

Se você quiser usar caracteres começando do meio da sequência, ao invés de no começo ou no fim, a função MID\$ é utilizada.

Digite:

```
PRINT MID$ (NOME$,8)
```

O seu computador responderá com:

```
DE ASSIS
```

pois "D" é o oitavo caracter da sequência. Agora experimente este programa:

```
NEW
190 NOME$= "MACHADO DE ASSIS"
200 FOR N=1 TO LEN (NOME$)
210 PRINT MID$ (NOME$,N)
220 NEXT N
```

Antes de rodá-lo, tente imaginar o que vai obter.

Suponhamos que você queira imprimir somente "O DE ASSI" da sequência chamada NOME\$. Para fazer isto, adicione um outro argumento à função MID\$:

```
PRINT MID$ (NOME$,7,9)
```

O primeiro número (7) especifica a posição do caracter que o computador deve começar a impressão. O segundo número (9) indica quantos caracteres devem ser impressos. Então, a instrução

ção é interpretada pelo computador como "procure o sétimo caracter de NOME\$ e imprima nove caracteres começando do sétimo e indo para a direita". Mude a linha 210 do programa anterior como segue e rode-o.

```
210 PRINT MID$ (NAME$,N,6)
```

Antes de seguir na leitura deste manual, pratique bastante com as funções LEFT\$, RIGHT\$ e MID\$.

Considere agora este programa:

```
NEW
200 A$="ABCDEFGHIJKLMNOPQRSTUVWXYZ"
210 PRINT
220 PRINT "ESCREVA UM NUMERO DE 1 ATE' ";LEN(A$);" "
230 PRINT "E EU LHE DIREI QUE LETRA DO ALFABETO "
240?"ESTA' NAQUELA POSICAO ";
250 INPUT P
260 IF P>LEN(A$) OR P<1 THEN GOTO 210
270 PRINT
280 PRINT MID$(A$,P,1);" E A LETRA NUMERO";P;" DO
    ALFABETO"
290 PRINT : PRINT
300 PRINT " ESCREVA UMA LETRA E EU LHE DIREI "
310 INPUT " ONDE ELA ESTA' NO ALFABETO ";X$
320 FOR N = 1 TO LEN(A$)
330 IF MID$(A$,N,1) = X$ THEN GOTO 380
340 NEXT N
350 PRINT
360 PRINT " ISTO NAO E' UMA LETRA DO ALFABETO."; PRINT
370 GOTO 300
380 PRINT
390 PRINT X$ " E' A LETRA NUMERO ";N;" DO ALFABETO
400 PRINT
410 GOTO 210
```

Este programa ilustra algumas práticas comuns de programação. Note como ele encontra a posição de um caracter numa sequência. Este método de usar um "loop", para examinar minuciosamente numa sequência uma posição de cada vez é bastante comum. Note também a função dos espaços em branco nos comandos PRINT. O que acontecerá com a saída sem estes espaços em branco? Finalmente, observe que os limites do programa são sempre determinados por LEN(A\$), dimensionado pelo número de caracteres do alfabeto. Isto permite que o programa funcione mesmo que você especifique um alfabeto diferente na linha 200. Experimente e veja.

Você pode substituir uma sequência por outra com um comando

de substituição como:

```
X$ = A$
```

Este comando copia o conteúdo A\$ em X\$. No entanto, você não pode usar as funções de notação parcial no lado esquerdo de um comando de substituição. Por exemplo, o comando:

```
MID$ = (S$,3,3) = "XYZ"
```

é ilegal, mas o comando:

```
X$ = MID$ (A$,24,3) está correto.
```

Somente uma variável pode estar do lado esquerdo em um comando de substituição.

Você ainda quer ver o seu nome soletrado de trás para frente? O programa a seguir fará exatamente isto:

```
NEW
100 REM PROGRAMA PARA SOLETRAR O SEU NOME AO CONTRARIO
110 INPUT "ESCREVA O SEU NOME E EU O MOSTRAREI AO
    CONTRARIO";N$
120 REM INVERTER A ORDEM DAS LETRAS
130 FOR T= LEN(N$) TO 1 STEP -1
140 R$ = R$+(MID$(N$,T,1))
150 NEXT T
160 PRINT : PRINT "O SEU NOME AO CONTRARIO E'";R$
170 PRINT : PRINT
180 GOTO 110
```

Rode este programa, experimente vários nomes diferentes. Após o programa ser executado várias vezes, você notará que há algo errado. A linha 140 é a chave do problema. Se o seu nome é MARIA, por exemplo, você escreverá quando solicitado e colocará MARIA em N\$ e AIRAM em R\$. Talvez você queira ver como é também o nome PEDRO ao contrário. A próxima vez que o programa solicitar um nome, escreva PEDRO em N\$. A linha 140 irá então colocar em R\$ o valor antigo de R\$ mais N\$ ao contrário, em outras palavras, AIRAMORDEP. Há, portanto, a necessidade de se introduzir um comando que torne a colocar zero nas variáveis sequenciais, para que R\$ possa ser recarregado com os novos caracteres após cada GOTO.

Felizmente existe este comando na linguagem. É o comando CLEAR. Este comando coloca as variáveis de todos os tipos, cores e tamanhos em 0. Adicione esta linha ao seu programa:

```
175 CLEAR
```

Agora rode (RUN) o programa novamente.

O comando CLEAR também pode ser usado em execução imediata. Digite:

```
N = 254
PRINT N
```

Agora digite:

```
CLEAR
e então
PRINT N
```

novamente. O seu computador escreveu 0 como o valor de N?

CONCATENAÇÃO

É possível adicionar uma segunda sequência ao final de uma já existente usando o sinal de mais (+). Este processo é denominado concatenação. Experimente no seu computador o seguinte:

```
C$ = "BOM DIA"
D$ = C$ + " " + "RENATO"
PRINT D$
```

O seu computador responderá com:

```
BOM DIA RENATO
```

A concatenação é especialmente útil se você desejar retirar parte de uma sequência e depois recolocar esta parte com pequenas modificações.

Por exemplo, se você quisesse criar uma sequência que fosse igual a D\$ e que nesta nova sequência, traços separariam as palavras (no lugar de espaços em branco), você poderia escrever:

```
E$ = RIGHT$(D$,6) + "-" + LEFT$(D$,3) +
      "-" + MID$(D$,5,3)
PRINT E$
```

e

RENATO-BOM-DIA apareceria no vídeo.

Abaixo, temos um programa que usa concatenação:

```
NEW
100 INPUT " ESCREVA METADE DE
UMA FRASE ";PRIMEIRA$
110 INPUT " AGORA ESCREVA A SEGUNDA PARTE DA FRASE ";
SEGUNDA$
120 S$ = PRIMEIRA$ + SEGUNDA$
130 PRINT
140 PRINT S$
150 PRINT:PRINT:PRINT:GOTO 100
```

Este é o modo de usar corretamente a concatenação.

MAIS FUNÇÕES DE ALFANUMÉRICOS

Sequências podem ser compostas de quase todos os tipos de caracteres, inclusive números. Entretanto, assim como no comando PRINT, os caracteres da sequência que estiverem entre aspas não podem ser interpretados aritmeticamente mesmo que sejam números. Para comprovar essa afirmação, experimente:

```
C$ = "123"
PRINT C$ + 7
```

O seu computador irá imprimir:

```
?TYPE MISMATCH ERROR
```

e não saberá lidar com a última instrução. Precisamos do auxílio da função

VAL

(abreviação de valor) para resolver este problema.

A função VAL retorna o valor dos conteúdos de uma sequência

como oposto aos conteúdos atuais. Digite:

```
PRINT C$
e então digite:
PRINT VAL(C$)
```

Aparentemente as duas instruções dão o mesmo resultado; no entanto, as aparências enganam. Você já sabe que se digitar:

```
PRINT C$ + 5
```

O seu computador responderá com:

```
?TYPE MISMATCH ERROR
```

Experimente digitar:

```
PRINT VAL(C$) + 5
e
128 aparecerá na tela.
```

Note que o nome da variável sequencial que é o argumento da função VAL deve sempre estar entre parênteses.

Se você quiser colocar o valor de C\$ menos 21 em uma variável comum (não sequencial), simplesmente escreva:

```
Q = VAL(C$) - 21
```

Agora escreva:

```
PRINT Q
```

e veja o que você obtém. O conteúdo de Q era o que você esperava? Você também pode usar VAL para adicionar o valor numérico das duas sequências diferentes. Para experimentar, crie uma nova sequência, assim:

```
K$ = "12"
```

e então escreva:

```
P = VAL(C$) + VAL(K$)
PRINT P
```

Experimente VAL com sequências diferentes, incluindo as que comecem ou terminem com letras.

Algumas vezes é necessário transformar um número em uma variável sequencial. A função STR\$, que funciona com o reverso da função VAL, pode ser usada para fazer esta transformação. Suponhamos que você queira transformar a variável numérica P em uma sequencial. Escreva:

```
P$ = STR$(P)
PRINT P$
```

Você verá como STR\$ funciona neste programa que usa STR\$ e VAL:

```
300 INPUT "ESCREVA UM NUMERO DE 1 A 999999999 ";N$
310 N = VAL(N$)
320 IF N<1 OR N>999999999 THEN GOTO 300
330 N$ = STR$(N)
340 FOR T = LEN(N$) TO 1 STEP -1
350 P$ = P$ + MID$(N$,T,1)
360 NEXT T
370 PRINT : PRINT "ORIGINAL",N$
380 PRINT : PRINT "REVERSO",P$
390 P = VAL(P$)
400 PRINT : PRINT "ORIGINAL + REVERSO= ";N + P
410 CLEAR
420 PRINT : PRINT : GOTO 300
```

Você entendeu como o programa funciona?

Porque existem vírgulas (,) nas linhas 370 e 380? Experimente apagar a linha 330 para observar qual o efeito. As primeiras quatro linhas deste programa demonstram os primeiros passos para fazer uma rotina de entrada verdadeiramente "a prova de erros".

Verifique quais entradas ainda podem parar este programa e então procure descobrir meios de detetá-los antes que façam o programa parar.

INTRODUÇÃO AOS AGRUPAMENTOS (MATRIZES)

Nesta seção usaremos os AGRUPAMENTOS em exemplos da matemática

ca recreativa que não necessitarão de nada mais que aritmética elementar.

Os AGRUPAMENTOS lhe darão condições para solucionar qualquer elemento de uma tabela de números e a programação se tornará tão mais precisa que compensará o tempo dispendido no seu aprendizado.

Um AGRUPAMENTO é uma tabela de números. O nome desta tabela pode ser qualquer dos permitidos para variáveis. A, por exemplo, é um nome permitido.

O nome do AGRUPAMENTO A é diferente e separado da simples variável A.

Para criar um AGRUPAMENTO, você primeiro precisa dizer ao computador o número máximo de elementos que o AGRUPAMENTO vai conter. Para fazer isto use o comando DIM (DIM significa DIMENSÃO). Os elementos de um AGRUPAMENTO são numerados a partir de 0. Para dimensionar um AGRUPAMENTO chamado A que terá no máximo 16 elementos, escreva:

```
DIM A(15)
```

O comando DIM acima nos forneceu 16 novas variáveis. Elas se comportam exatamente como as variáveis que você já aprendeu. Elas são:

```
A(0)
A(1)
A(2)
e assim sucessivamente, até
A(15)
```

Embora você possa achá-las desajeitadas para se escrever, elas podem ser usadas da mesma maneira que qualquer outra variável. A instrução:

```
A(9) = 45 + A(12)
```

é perfeitamente correta. O número entre parênteses é chamado subscrito, e a notação A(12) é lida "A-sub-doze". O subscrito pode ser uma expressão aritmética ou uma variável.

O programa seguinte ilustra o uso das variáveis num subscrito

to e imprime um quadro dos conteúdos de cada elemento do AGRUPAMENTO:

```
100 REM DIMENSIONAR AGRUPAMENTO CHAMADO DIAS, PARA
    AGUARDAR 7 NUMEROS
110 DIM DIAS(6)
120 REM PREENCHER O AGRUPAMENTO
130 FOR NUM= 0 TO 6
140 DIAS (NUM)= NUM + 1
150 NEXT NUM
160 REM IMPRIMIR OS ELEMENTOS DO AGRUPAMENTO
170 FOR I= 0 TO 6
180 PRINT "DIAS(";I;")= ";DIAS(I)
190 NEXT I
```

Se um AGRUPAMENTO for usado em um programa antes de ser DIMENSIONADO, o computador reserva espaço para 11 elementos (subscritos 0 até 10). No entanto, é aconselhável sempre dimensionar todos os AGRUPAMENTOS.

Suponhamos que você queira escrever um programa que gere números de um a oito de forma desordenada. Para realizar isso você precisa manipular tabelas de dados. Este é exatamente o lugar onde AGRUPAMENTOS são excelentes. O programa a seguir ilustra bem o uso dos AGRUPAMENTOS:

```
NEW
200 REM DIMENSIONAR O AGRUPAMENTO
210 DIM COPO(8)
220 REM PREENCHER O AGRUPAMENTO
230 FOR I= 1 TO 8
240 COPO(I)= I
250 NEXT I
260 REM DESORDENAR O AGRUPAMENTO E ESCOLHER CADA
    ELEMENTO
270 FOR VINHO= 1 TO 8
280 REM ESCOLHER ALGUM OUTRO ELEMENTO
290 LEITE= INT(RND(1) * 8) + 1
300 REM LEITE E' DIFERENTE DE VINHO?
310 REM SE NAO FOR, EXPERIMENTE NOVAMENTE
320 IF LEITE= VINHO THEN GOTO 280
330 REM ALTERNAR COPO(VINHO) E COPO(LEITE)
340 TEMP= COPO(VINHO) : COPO(VINHO)= COPO(LEITE) :
    COPO(LEITE)= TEMP
350 NEXT VINHO
360 REM IMPRIMIR O CONTEUDO DO AGRUPAMENTO
370 FOR C= 1 TO 8
380 PRINT COPO(C)
390 NEXT C
```

Você compreendeu como este programa funciona? Ele preenche

primeiro um AGRUPAMENTO com números e então mistura-os.

Note que você não precisa começar a preencher o AGRUPAMENTO em zero.

As linhas 230 e 250 preenchem o AGRUPAMENTO e designam a cada seu elemento um número correspondente ao seu "número de AGRUPAMENTO (COPO(1) = 1, etc.) A linha 270 coloca na nova variável VINHO números de 1 a 8. A linha 280 coloca na variável LEITE números aleatórios inteiros de 1 a 8. Então a linha 300 se certifica que o valor de VINHO seja sempre diferente do valor de LEITE. O conteúdo das variáveis COPO(VINHO) e COPO(LEITE) são trocados na linha 310. Finalmente o AGRUPAMENTO é impresso nas linhas 330 e 350.

A troca que ocorre na linha 310 pode ser interpretada da seguinte forma:

Vamos imaginar que temos dois copos, um de vinho (VINHO) e outro de leite (LEITE). Houve um engano. O leite está no copo de vinho e o vinho está no de leite. Felizmente temos um copo extra (TEMP). Podemos colocar o leite no copo extra, então colocar o vinho no copo de vinho e, finalmente, colocar o leite no copo de leite. Agora as bebidas estão nos seus devidos copos.

MENSAGENS DE ERRO DOS AGRUPAMENTOS

Mostraremos abaixo algumas mensagens de erro que você talvez receba quando estiver utilizando AGRUPAMENTOS em programas:

?REDIM'D ERROR

Esta mensagem aparece quando você dimensiona um AGRUPAMENTO mais de uma vez no mesmo programa. Frequentemente este erro ocorre quando deixamos de dimensionar um AGRUPAMENTO (deixamos que o AGRUPAMENTO assuma a dimensão 11) e posteriormente uma instrução para dimensionamento é acionada ao programa.

?BAD SUBSCRIPT ERROR

Se for feita uma tentativa de usar um elemento do AGRUPAMENTO que está fora de sua dimensão, esta mensagem de erro aparecerá. Por exemplo, se A for dimensionado para 25 através

do comando DIM A(25), caso você se refira ao elemento A(52) ou qualquer outro que não esteja entre A(0) e A(25), a mensagem acima irá aparecer.

?ILLEGAL QUANTITY ERROR

Você obterá esta mensagem se tentar usar números negativos como subscrito de um AGRUPAMENTO.

Estas são algumas maneiras de como você pode usar AGRUPAMENTOS. Os AGRUPAMENTOS usados neste livro são todos de uma dimensão. Você também pode usar AGRUPAMENTOS que tenham duas ou mais dimensões. Experimente!

CONCLUSÃO

Este manual apresentou o básico da linguagem. Se você tornar a lê-lo, escrevendo os seus próprios programas com os comandos que foram apresentados, você irá solidificar os seus conhecimentos consideravelmente. O seu computador tem muitas outras habilidades e uma vez que você assimilou todas as aqui apresentadas, há ainda várias para você explorar e descobrir.

APÉNDICES

APÊNDICE "A"

RESUMO DOS COMANDOS

O que se segue é um resumo dos comandos que podem ser usados em programação na linguagem de seu computador.

TECLAS COM SETAS

As teclas indicadas por uma seta apontando para a direita e outra para a esquerda, são usadas para corrigir programas. A tecla + move o cursor para a direita; todo caractere sobre o qual ela passa é interpretado como se tivesse sido escrito. A tecla - move o cursor para a esquerda; à medida que o cursor se move, o caractere sobre o qual ela passa é eliminado da linha do programa que está sendo escrito.

CALL-151

Provoca o aparecimento do "prompt" asterísco (*) que indica que o computador está agora na sua linguagem nativa, chamada linguagem de máquina. Se você não é um programador avançado, provavelmente não necessitará deste comando.

CATALOG

Coloca na tela a lista de todos os arquivos contidos no disco flexível que você inseriu na unidade de disco ("disk-drive") especificada. O número de setores ocupados e o tipo do arquivo são indicados à esquerda do nome do programa (ou arquivo). Os tipos de arquivo são:

- I - O arquivo é um programa na linguagem BASIC INTEGRAL ("Integer BASIC").
- A - O arquivo é um programa na linguagem de seu computador (APPLESOFT).
- T - O arquivo consiste num texto (ele foi criado por um comando WRITE).
- B - O arquivo é uma gravação de "bit-por-bit" de uma parte da memória do seu computador.

O comando CATALOG é um comando do DOS (Sistema de Operação do Disco).

CLEAR

Coloca todas as variáveis em zero e torna nulas todas as sequências.

COLOR= 12

Determina a cor que usaremos em gráficos de baixa-resolução. No exemplo, a cor será verde. O comando GR coloca a cor em zero. O nome das cores e seus respectivos números são:

0 Preto	8 Marrom
1 Vermelho	9 Laranja
2 Azul Escuro	10 Cinza
3 Lilás	11 Cor de rosa
4 Verde Escuro	12 Verde
5 Cinza	13 Amarelo
6 Azul	14 Azul Piscina
7 Azul Claro	15 Branco

CONT

Se a execução do programa foi interrompida por STOP, END ou CTRL C, o comando CONT faz com que a execução prossiga a partir da próxima instrução (como GOSUB) e não da próxima linha.

CONT não pode ser usado se você tiver:

- a) modificando, adicionado ou apagado uma linha de programa, ou,
- b) obtido alguma mensagem de erro e a execução do programa foi interrompida.

CTRL C

Pode ser usado para interromper um programa ou uma listagem. Também pode ser usado para interromper um INPUT, se este for

o primeiro caractere inserido. O INPUT não é interrompido quando a tecla RETURN não for pressionada.

CTRL X

Diz ao computador para ignorar a linha que está sendo escrita, sem contudo apagar qualquer outra linha de mesmo número. Uma barra (/) é colocada no final da linha que deve ser ignorada.

DEL 23,56

Remove do programa as linhas compreendidas entre os dois números. No exemplo dado, as linhas de 23 a 56 serão eliminadas do programa. Para apagar apenas uma linha como por exemplo a 350, use a forma DEL 350,350 ou simplesmente escreva o número da linha e tecle RETURN.

DIM NOME\$(50)

Quando um comando DIM é executado, ele cria um espaço para o AGRUPAMENTO especificado com seus subscritos variando desde 0 até o subscrito dado. No exemplo dado, NOME\$(50) reservará espaço para 51 sequências de qualquer tamanho. Se um elemento de um AGRUPAMENTO é usado num programa antes de ser dimensionado, um subscrito máximo de 10 é reservado para cada dimensão no subscrito do elemento. Elementos de AGRUPAMENTO são colocados em zero quando RUN ou CLEAR são executados.

END

Provoca o cancelamento da execução do programa e retorna o controle ao usuário. Nenhuma mensagem é escrita.

ESC I ou ESC J ou ESC K ou ESC M

A tecla ESC deve ser usada juntamente com as letras I, J, K ou M para mover o cursor sem afetar os caracteres sobre os quais ele passou. Para mover o cursor, primeiro pressione e solte a tecla ESC para entrar em modo de edição. Depois pressione a tecla apropriada para mover o cursor para a direção

que você desejar. A tecla RPT pode ser usada para acelerar os movimentos: para isso, pressione a letra desejada e simultaneamente tecle RPT.

Comando Move o cursor um espaço para

ESC I	cima
ESC J	esquerda
ESC K	direita
ESC M	baixo

FLASH

Este comando faz com que a saída do computador fique piscando, ou seja, alternando entre caracteres brancos em fundo preto e caracteres pretos em fundo branco. Use NORMAL para a tela parar de piscar voltando assim ao normal.

```
FOR W = 1 TO 20 ..... NEXT W
FOR Q = 2 TO -3 STEP -2 ..... NEXT Q
FOR Z = 5 TO 4 STEP 3 ..... NEXT Z
```

Permite que você escreva um "loop" para realizar determinadas vezes qualquer instrução entre o comando FOR (topo do loop) e o comando NEXT (fim do loop).

No primeiro exemplo, a variável W conta quantas vezes a instrução deve ser executada. As instruções contidas no loop serão executadas para W igual a 1, 2, 3.... 20 e então o loop termina (com W = 21) e a instrução após NEXT W é executada. O segundo exemplo ilustra como fazer para o passo (STEP) desejado ser diferente de 1. A verificação ocorre no final do loop; então no terceiro exemplo a instrução contida no loop é executada uma vez.

GOSUB 250

Faz o programa pular para linha indicada (250, no exemplo dado). Quando o comando RETURN é executado, o programa retorna para o comando imediatamente após o último GOSUB.

GOTO 250

Faz o programa pular para a linha indicada (250, no exemplo

dado).

GR

Aciona o modo gráfico de baixa-resolução (40 por 40), deixando as últimas quatro linhas para o texto. A tela é limpa, o cursor é movido para a área de texto, e 0 (preto) é colocado em COLOR.

HCOLOR = 4

Determina a cor dos gráficos de alta-resolução. Os nomes e os números das cores são:

0 Preto 1	4 Preto 2
1 Verde	5 Laranja
2 Violeta	6 Azul
3 Branco 1	7 Branco 2

HGR

Aciona o modo gráfico de alta-resolução (280 por 160), deixando as quatro últimas linhas para texto. A tela é limpa, e a página 1 da memória é acionada. Tanto HCOLOR como a memória de texto da tela não são afetados quando HGR é executado. O cursor não é movido para a área de texto.

HGR 2

Aciona o modo gráfico de alta-resolução sem deixar a janela de texto (280 por 192). A tela é limpa e a página 2 da memória é acionada. A memória da tela de texto não é afetada.

HLIN 10,20 AT 30

Usado para desenhar linhas horizontais em gráficos de baixa resolução, usando a mais recente cor especificada pela função COLOR. A origem (X = 0 e Y = 0) para o sistema é um ponto situado no canto superior esquerdo da tela. No exemplo acima, a linha é desenhada desde X = 20 em Y = 30. Uma outra

maneira de dizer isto é: a linha é desenhada desde o ponto (10,30) até o ponto (20,30).

HOME

Move o cursor para o topo da janela de texto, limpando todo o texto da janela.

```
HPLOT 10,20
HPLOT 30,40 TO 50,80
HPLOT TO 70,80
```

Marcam pontos e linhas em modo gráfico de alta resolução usando o valor mais recente especificado por HCOLOR. A origem é o ponto no canto superior esquerdo (X = 0, Y = 0). O primeiro exemplo marca um ponto de alta-resolução em X = 10, Y = 20. O segundo exemplo desenha uma linha de alta-resolução do ponto X = 30, Y = 40 até o ponto X = 50, Y = 60. O terceiro exemplo desenha uma linha desde o último ponto que foi marcado até o ponto X = 70, Y = 80, usando a cor do último ponto marcado, não necessariamente o mais recente HCOLOR.

HTAB 23

Move o cursor tanto para a esquerda como para a direita até a coluna da tela especificada (1 até 40). No exemplo dado, o cursor será posicionado na coluna 23.

```
IF IDADE < 18 THEN A = 0 : B = 1 : C = 2
IF RSP = "SIM" THEN GOTO 100
IF N < MAX THEN GOTO 25
```

Se a expressão que sucede o comando IF for verdadeira (não 0), as instruções que sucedem THEN na mesma linha, serão executadas. Caso contrário, qualquer instrução que venha após THEN será ignorada, e a execução do programa continua na linha seguinte. Expressões com string são avaliadas de acordo com a ordem alfabética.

```
INPUT A
INPUT "ESCREVA IDADE DEPOIS UMA VIRGULA DEPOIS
      NOME. " ; B , C$
```

No primeiro exemplo, INPUT imprime um ponto de interrogação e espera o usuário escrever um número que será designado à variável numérica A. No segundo exemplo, INPUT imprime exatamente o que está entre aspas e espera o usuário escrever um número (que será designado à variável B) e depois uma vírgula, e depois o STRING (que será designado à variável String C\$). Entradas múltiplas num comando INPUT devem ser separadas por vírgulas ou RETURNS.

INT (NUM)

Fornece o maior inteiro, menor ou igual ao argumento dado. No exemplo, se NUM é 2.389, então 2 será fornecido. Se NUM é -45.123345 então -46 será fornecido.

INVERSE

Coloca o vídeo no modo em que a saída do computador é impressa com letras pretas em fundo branco. Use o comando NORMAL para retornar à impressão com letras brancas em fundo preto.

```
LEFT$ ("microcomputador",5)
```

Fornece o número especificado de caracteres à esquerda do string. No exemplo dado, MICRO (os 5 caracteres mais à esquerda) serão fornecidos.

TECLA " + "

Veja "Teclas com Setas".

```
LEN ("UMA FRUTA POR DIA")
LEN (B$)
```

Fornece o número de caracteres em um string, entre 0 e 255. No primeiro exemplo, 17 serão fornecidos.

```
A = 23.567
A$ = "DELICIOSO"
```

O nome da variável à esquerda de = é designada com o valor do string ou da expressão situada à direita de =.

```
LIST
LIST 200 , 3000
LIST 200 - 3000
```

O primeiro exemplo provoca o aparecimento de todo o programa na tela da TV; o segundo e terceiro exemplos fazem com que sejam listadas as linhas de 200 até 3000. Para listar do começo do programa até a linha 200, use LIST -200; para listar da linha 200 até o fim do programa, use LIST 200-. A listagem é "abortada" por CTRL C, e o comando CONT não pode ser usado. Para interromper o programa temporariamente em algum ponto da listagem, use CTRL S. Use CTRL S novamente para continuar a listagem.

LOAD

Lê um programa da fita cassete para inserir na memória do computador. Nenhum "prompt" é dado; o usuário deve voltar a fita e apertar "play" no gravador antes de LOAD. Um bip é ouvido quando a informação é encontrada na fita que está sendo carregada (LOAD).

Quando LOAD é satisfatoriamente completado, um segundo bip será ouvido e o caractere prompt da linguagem (>) reaparecerá. Somente o RESET pode interromper um LOAD.

LOAD LISTAS

Procura encontrar no diskette do drive especificado (ou omitido) um "arquivo de programa" cujo nome é LISTAS. Se o programa for encontrado, ele será inserido (LOAD) na memória do computador. LOAD apaga qualquer programa antes de colocar o novo programa na memória. Este comando, quando seguido de um nome de arquivo, é um comando DOS.

```
MID$ ("UMA PERA POR DIA" , 4)
MID$ (DIA$ , 4, 9)
```

Retorna o substring especificado. No primeiro exemplo, do quarto até o último caracter do string serão impressos: PERA POR DIA. No segundo exemplo, nove caracteres, a partir do quarto serão impressos: PERA POR.

NEW

Apaga o programa da memória e todas as variáveis.

NEXT

Veja a explanação sobre FOR... TO... STEP.

NORMAL

Coloca o vídeo no modo usual, ou seja, letras brancas em fundo preto, tanto para entrada como para saída.

NOTRACE

Anula o comando TRACE. Veja TRACE.

PDL (1)

Retorna o valor usual, um número de 0 a 255, indicado no controle. Pode-se utilizar de 0 a 3 controles.

PLOT 10,20

Em gráficos de baixa-resolução, marca um ponto no lugar especificado. No exemplo, o ponto estará em X=10, Y=20. A cor do ponto é determinada pelo valor mais recente de COLOR, que será 0 (preto) se não for previamente especificado.

PRINT

PRINT A\$; "X = " ; X

O primeiro exemplo faz o computador "pular uma linha", isto é, imprime uma linha em branco na tela. Itens de uma lista a ser impressa devem ser separados por vírgulas (,) se cada um deva ser mostrado em um campo tabulado. Os itens devem ser separados por ponto e vírgulas (;) se devam ser impressos um após o outro sem deixar espaços em branco. Se A\$ contém "CORE" e X é 3, o segundo exemplo imprimirá:

COREX = 3

REM ISTO E' UM LEMBRETE

Permite que seja feita uma inserção de texto num programa, como lembrete.

REPETIÇÃO

Se você pressionar a tecla de repetição, denominada RPT, enquanto estiver pressionando qualquer outro caracter, este será repetido.

RETURN

O programa retorna para o comando imediatamente após o último GOSUB.

RIGHT\$ (" COMPUTADOR ", 5)
RIGHT\$ (S \$, 2)

Fornece o número especificado dos caracteres mais à direita de um STRING. No primeiro exemplo, TADOR (os 5 caracteres mais à direita) serão fornecidos.

RIGHT ARROW

Veja "Teclas Setas".

RND (5)

Fornece um número aleatório real maior ou igual a 0 e menor do que 1. RND (0) retorna o mais recente número aleatório gerado. Cada argumento negativo gera um número aleatório particular que é o mesmo todas as vezes que RND é usado com aquele argumento, e subseqüentes aleatórios (RND) com argumentos positivos sempre irão seguir uma seqüência repetitiva particular. Toda vez que RND é usado com qualquer argumento positivo, um novo número aleatório de 0 a 1 é gerado, a menos que faça parte de uma seqüência de números aleatórios, indicados por um argumento negativo.

RUN 500

Apaga todas as variáveis, indicadores, "stacks" e começa a execução do programa na linha indicada (500 no exemplo). Se o número da linha não for especificado, a execução começa no menor número de linha do programa.

RUN ESTOQUE

Carrega (LOAD) o arquivo chamado ESTOQUE do drive especificado ou omitido e então roda (RUN) o programa carregado (LOAD). Quando seguido de um nome de arquivo, RUN é um comando do DOS.

SAVE

Guarda o programa da memória, na fita cassete. Nenhum "prompt" ou sinal é dado. O usuário deve pressionar "record" e "play" no gravador antes de executar o SAVE. O comando SAVE não verifica se os botões corretos do gravador estão apertados; "bips" sinalizam o começo e o fim da gravação.

SAVE CAIXA

Guarda o arquivo da memória. Se nenhum arquivo chamado CAIXA for encontrado no diskette especificado (ou omitido), um arquivo é criado naquele diskette e o programa que estava na memória é guardado (SAVE) com esse nome. Se o diskette já contiver um arquivo com esse mesmo nome e na mesma linguagem, o conteúdo do arquivo original é perdido e o programa novo é guardado no seu lugar. Nenhum aviso é dado. SAVE, quando seguido de um nome de arquivo é um comando do DOS.

STR\$ (12.45)

Fornece o string que representa o valor do argumento. No exemplo, o string "12.45" é fornecido.

TAB (23)

Deve ser usado num comando PRINT; o argumento deve estar entre parênteses e variar entre 0 e 255. Para argumentos de 1 até 255, se o argumento é maior do que o valor da posição atual do cursor, o TAB move o cursor para a posição especificada, contando a partir do extremo esquerdo da atual linha

do cursor. Se o argumento é menor do que o valor da posição atual do cursor, então o cursor não é movido. TAB (0) coloca o cursor na posição 256.

TEXT

Coloca a tela no modo usual de texto (não gráfico), com 40 caracteres por linha e 24 linhas. Também a janela de texto é recolocada na tela toda.

TRACE

Faz com que o número da linha de cada comando apareça na tela assim que é executado. TRACE não é "desligado" por RUN, CLEAR, NEW, DEL ou RESET. NOTRACE anula o TRACE.

VAL (" -3.7 E 4 A 5 PLE ")

Tenta interpretar um string, até o primeiro caracter não numérico, como um real ou um inteiro, e fornece o valor daquele número. Se nenhum número aparece antes do primeiro caracter não numérico, um 0 é fornecido. No exemplo, -37000 é fornecido.

VLIN 1,20 AT 30

Em modo gráfico, de baixa-resolução, desenha uma linha vertical na cor indicada pelo mais recente comando COLOR. A linha é desenhada na coluna indicada pelo terceiro argumento. No exemplo, a linha é desenhada desde Y = 0 até Y = 20 em X = 30.

VTAB (15)

Move o cursor para a linha especificada pelo argumento. A linha superior é a linha 1, a linha inferior é a linha 24. VTAB moverá o cursor para cima ou para baixo, mas não para a esquerda ou a direita.

APENDICE "B"

PALAVRAS RESERVADAS NA LINGUAGEM

A lista abaixo contém todas as palavras reservadas na linguagem. No entanto, a maioria destas palavras não aparece neste manual. A lista é útil como um guia para nomear variáveis.

&	GET	NEW	SAVE
ABS	GOSUB	NEXT	SCALE=
AND	GOTO	NORMAL	SCRN(
ASC	GR	NOT	SGN
AT	HCOLOR=	NOTRACE	SHLOAD
ATN	HGR	ON	SIN
CALL	HGR2	ONERR	SPC(
CHR\$	HIMEM:	OR	SPEED=
CLEAR	HLIN	PDL	SQR
COLOR=	HOME	PEEK	STEP
CONT	HPLOT	PLOT	STOP
COS	HTAB	POKE	STORE
DATA	IF	POP	STR\$
DEF	IN/	POS	TAB(
DEL	INPUT	PRINT	TAN
DIM	INT	PR/	TEXT
DRAW	INVERSE	READ	THEN
END	LEFT\$	RECALL	TO
EXP	LEN	REM	TRACE
FLASH	LET	RESTORE	USR
FN	LIST	RESUME	VAL
FOR	LOAD	RETURN	VLIN
FRE	LOG	RIGHT\$	VTAB
	LOMEM:	RND	WAIT
	MID\$	ROT=	XPLOT
		RUN	XDRAW

O símbolo & é projetado somente para o uso interno do computador; não é um comando particular da linguagem. Este símbolo quando executado como uma instrução, causa um salto incondicional para a posição \$ 3F5.

XPLOT é uma palavra reservada que não corresponde a um comando real.

Algumas palavras reservadas são reconhecidas pela linguagem

somente em certos contextos.

COLOR, HCOLOR, SCALE, SPEED E ROT

Analisar como palavras reservadas somente se o próximo caractere não espaço é o sinal de substituição, =. Isto é de pouca utilidade no caso de COLOR e HCOLOR, pois como elas contêm a palavra reservada OR, não podem ser usadas como nomes de variáveis de qualquer forma.

SCRN, SPC E TAB

Analisar como palavras reservadas somente se o próximo caractere não espaço é um parêntese esquerdo, (.

HIMEM: Deve ser seguido dos dois pontos (:) para ser analisado como uma palavra reservada.

LOMEM: Também requer os dois pontos (:) se quiser ser analisada como palavra reservada.

ATN é analisado como uma palavra reservada somente se não houver espaço entre o T e o N. Se houver um espaço entre "T" e "N", a palavra reservada AT é analisada no lugar de ATN.

TO é analisado como uma palavra reservada, a menos que seja precedido por um A e houver um espaço entre "T" e "O". Se ocorrer um espaço entre "T" e "O", a palavra AT é analisada no lugar de TO.

Algumas vezes podem ser usados parênteses para contornar as palavras reservadas:

100 FOR A = LOFT OR CAT TO 15

E listado assim:

100 FOR A = LOF TO RC AT TO 15

Mas

100 FOR A = (LOFT) OR (CAT) TO 15

É listado

100 FOR A = (LOFT) OR (CAT) TO 15

APÊNDICE "C"

CARACTERÍSTICAS DE EDIÇÃO

As (teclas) setas esquerda e direita.

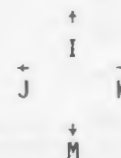
A seta esquerda + , também chamada tecla de retrocesso, move o cursor um espaço para trás (esquerda), apagando o caracte sob o qual passa. Se você não teclou RETURN no final da linha que escreveu, a tecla retrocesso somente afeta os caracteres naquela linha.

Pressionando a seta direita + , também conhecida como tecla de reescrever, o cursor se move para frente (direita) reescrevendo o caractere sob o qual ele passa. Se você reescrever uma linha com a tecla de reescrever + , então teclar RETURN, o computador irá se comportar como se você tivesse escrito a linha novamente.

Você pode acelerar o movimento do cursor pressionando a tecla RPT enquanto tecla alguma das setas.

MOVIMENTO PURO DO CURSOR

As teclas ESC I, J, K e M são usadas para mover o cursor sem afetar nenhum dos caracteres na tela. Imagine setas desenhadas nas teclas das letras como mostra a ilustração:



Pressionando ESC você entra no modo de edição. Uma vez que você está no modo de edição, ao pressionar qualquer uma das teclas acima mencionadas o cursor irá se mover de um caractere na direção da seta correspondente. Você pode usar estas teclas para mover o cursor para qualquer lugar da tela.

Para movimentos mais rápidos, pressione uma destas teclas e depois tecle RPT. O cursor irá circular através da tela en-

quanto ambas as teclas estiverem pressionadas. Se o cursor atingir o topo da tela, ele parará. Se atingir a parte inferior, ele pára e as linhas começam "deslizar para cima". Se ele atingir o canto direito, o cursor irá desaparecer e reaparecer no canto esquerdo, mas na próxima linha. No canto esquerdo, ele reaparecerá no direito, uma linha para cima. Para retornar ao modo normal, pressione a barra de espaço uma vez.

APAGANDO LINHAS DE PROGRAMAS

Um método simples de apagar linhas de programas é simplesmente escrever o número da linha que deseja apagar e depois teclar RETURN. Se você tem mais de uma linha para apagar, você deve utilizar o comando DEL. Para apagar por exemplo as linhas de 100 a 200, você escreveria:

```
DEL 100, 200
```

Todas as linhas de programa de 100 a 200, inclusive, serão apagadas (eliminadas).

Escrevendo CTRL X

a linha que você está escrevendo será eliminada. Este comando é útil quando você percebe que cometeu um erro antes de teclar RETURN.

LIMPANDO A TELA

O comando a seguir afeta somente o que é visível na tela, não o que está arquivado na memória. Pressionando a tecla ESC uma vez, você entra no modo de edição.

Para limpar a tela, tecle:

SHIFT

ESC P

O cursor aparecerá no canto superior esquerdo da tela sem o "j". O caracter "j" aparecerá quando você teclar RETURN.

Se você já se encontra em modo de edição pode limpar a tela simplesmente teclando "@" (SHIFT P). O computador irá colocar você novamente em modo normal.

O comando HOME também é usado para limpar a tela. Simplesmente escreva:

HOME, tecle RETURN

e o cursor aparecerá no canto superior esquerdo da tela.

É também possível limpar somente pedaços da tela. Para limpar de um ponto da tela até o fim, entre em modo de edição pressionando ESC. Então use as características de movimento puro do cursor para movê-lo para o primeiro caracter que deseja limpar. Pressione a tecla F, e todos os caracteres daquele ponto em diante serão eliminados. Para "limpar" caracteres até o fim de uma linha, você deve primeiro estar em modo de edição. Então mova o cursor até o primeiro caractere a ser eliminado e pressione E. Em ambos os casos o computador voltará ao modo normal após o comando ser executado.

RESUMO DAS CARACTERÍSTICAS DE EDIÇÃO

Entrar em modo de edição	Tecle ESC
Sair do modo de edição	Tecle barra espaço
Mover o cursor	Tecle I J K ou M
Eliminar um caracter	Tecle +
Reescrever um caracter	Tecle +
Limpar do cursor até o final da linha	Tecle ESC então E
Limpar do cursor até o final da tela	Tecle ESC então F
Limpar a tela toda	Tecle ESC então SHIFT e P
Parar a listagem	Tecle CTRL e S
Continuar a listagem	Tecle CTRL e S

APÊNDICE "D"

132

MENSAGENS DE ERRO

Todas as mensagens de erro que podem ser geradas na linguagem "BASIC" estão listadas a seguir, com seus respectivos significados.

Quando ocorre um erro, o basic volta ao nível de comando, indicado pelo caractere "prompt" (>) e o cursor intermitente.

Os valores das variáveis e o texto do programa permanecem intactos, mas o programa não pode CONTINUAR e todos os comandos de GOSUB e "loops" FOR serão zerados!

Quando ocorre um erro num comando de execução imediata, não aparece número de linha.

Formato das mensagens de erro:

Comando de execução imediata ?XX ERROR
Comando de execução no programa ?XX ERROR IN YY

Em ambos os casos exemplificados, "XX" é o nome do erro, e "YY" é o número da linha de programa onde o erro ocorreu. Erro no programa não são detectados até que o programa seja executado.

São os seguintes os possíveis erros e seus significados:

?CAN'T CONTINUE ERROR

Tentativa de continuar um programa quando não existe nenhuma continuação, ou depois que ocorreu um erro, ou depois que uma linha foi apagada ou adicionada ao programa.

?DIVISION BY ZERO ERROR

Dividir por zero é um erro!

?FORMULA TOO COMPLEX ERROR

133

Mais de dois comandos do tipo IF "XX" THEN foram executados.

?ILLEGAL DIRECT ERROR

Você não pode usar INPUT, DEF FN, GET ou DATA como comando de execução imediata.

?ILLEGAL QUANTITY ERROR

O parâmetro, passando por uma função matemática ou string, cai fora da faixa de alcance.

Pode ocorrer devido a:

- a) SUBSCRIPT negativo (Ex: A(-1)=0)
- b) Usar LOG com argumento 0 ou negativo
- c) Usar SQR com argumento negativo
- d) A^B com A negativo e B não-inteiro
- e) Uso de MID\$, LEFT\$, RIGHT\$, WAIT, PEEK, POKE, TAB, SPC, ON...GOTO, ou qualquer função gráfica com argumento não apropriado.

?NEXT WITHOUT FOR ERROR

A variável num comando NEXT não corresponde à variável num comando FOR que ainda está válido, ou um NEXT sem nome correspondente a qualquer FOR que ainda estava válido.

?OUT OF DATA ERROR

Um comando READ foi executado, mas todos os comandos DATA no programa já foram lidos. O programa tentou ler muitos dados ou foram fornecidos poucos dados ao programa.

?OUT OF MEMORY ERROR

Pode ser causado por: programa muito extenso; muitas variáveis; mais de 10 "loops" de FOR; mais de 24 comandos GOSUB; expressão muito complexa; mais de 36 níveis de parênteses; tentativa de comandar LOMEM: menor que o valor presente; tentativa de comandar HIMEM: muito baixo.

?OVERFLOW ERROR

O resultado de um cálculo foi muito grande para ser apresentado no formato de números do BASIC. Se o erro ocorre por ser o número muito pequeno (muito próximo a 0), o zero é dado como resultado e a execução continua sem que seja apresentada mensagem de erro.

?REDIM'D ARRAY ERROR

Depois que um array foi dimensionado, outro comando de dimensão, para o mesmo array, foi encontrado. Este erro geralmente ocorre se é dada a um array a dimensão errada 10 devido a que um comando como A(I)=3 é seguido mais tarde num programa por DIM A(100). Esta mensagem de erro mostrar-se-á útil se você deseja descobrir em que linha do programa um certo array foi dimensionado: coloque a dimensão do array em questão na primeira linha, rode o programa, e o BASIC lhe dirá onde está o comando original de dimensão.

?RETURN WITHOUT GOSUB ERROR

Um comando RETURN foi encontrado sem que haja o correspondente GOSUB.

?STRING TOO LONG ERROR

Foi feita uma tentativa, através de um comando de concatenação, para criar um "string" com mais de 255 caracteres.

?BAD SUBSCRIPT ERROR

Foi feita uma tentativa de consultar um elemento de um array que está fora das dimensões do array. Este erro pode ocorrer se o número errado da dimensão é usado numa referência a um array; por exemplo, LET A(11)=Z quando A foi dimensionado usando DIM A(2).

?SINTAX ERROR

Falta de parênteses numa expressão, caractere ilegal numa linha, pontuação incorreta, etc.

?TYPE MISMATCH ERROR

O lado esquerdo de um comando foi uma variável numérica e o

lado direito foi um "string", ou vice-versa; ou a uma função que esperava um argumento de "string" foi dado um número, ou vice-versa.

?UNDEF'D STATEMENT ERROR

Tentativa de GOTO, GOSUB ou THEN a um número de linha que não existe.

?UNDEF'D FUNCTION ERROR

Refere-se a uma função criada pelo usuário, que ainda não foi definida.

GUIA DE REFERENCIA

VARIÁVEIS SIMPLES

TIPO	MODE	CAPACIDADE
Real	AB +/- 9.999999999 e +3/	
Básico	AB1 +/- 32767	
String	AB1 0 à 255 caracteres	

Onde A é uma letra, B é uma letra ou dígito. O nome pode ter mais de 2 caracteres, mas só os 2 primeiros são significativos: AB1 e AB3Q82 são a mesma variável.

VARIÁVEIS "ARRAY"

TIPO	ELEMENTO TÍPICO
Real	AB(3,12,7)
Básico	AB1(3,12,7)
String	AB1(3,12,7)

A dimensão do array é limitada pela memória disponível.

OPERAÇÕES ALGÉBRICAS

- Atribuição de valores
- Negação
- Exponenciação
- Multiplicação
- Divisão
- Adição
- Subtração

OPERAÇÕES LÓGICAS E DE RELAÇÃO

- Igual
- Diferente
- Menor que
- Maior que
- Menor ou igual
- Maior ou igual

NOT "Não lógico"
AND "e" lógico
OR "Ou" lógico

Expressões lógicas ou de relação sempre valor 1 se são verdadeiras e 0 se são falsas. Estes operadores também podem ser utilizados para "comparar" strings.

COMANDOS DE UTILIZAÇÃO

LOAD Carrega um programa da fita.
SAVE Grava um programa da memória.
REN Apaga o programa da memória.
RUN Executa o programa, começando no menor número de linhas.
RUN 477 Executa o programa, começando na linha 477.

STOP Detém a execução e informa em que linha foi detido.
END Simplesmente detém a execução sem nenhuma mensagem.

ctrl C É usado para deter o modo imediato um programa ou listagem.

reset (tela EST). Retorno imediato ao Monitor. Use ctrl C ou 9C para retornar ao básico.

CONT Continua a execução de um programa que foi parado por STOP, END ou ctrl C.

TRACE Lista cada número de linha e executa a linha.

NOTRACE Anula o TRACE.

PEEK(X) Fornece o conteúdo da memória na localização X.

POKE X, Y Muda o conteúdo da memória na localização X, para o valor Y.

WAIT X,Y,Z Espera até que o conteúdo da localização X quando comparado (OR) com Y e comparado (AND) com Z, de resultado diferente de 0 (zero).

CALL X Passa para a subrotina de linguagem de máquina, começando na localização X da memória.

USR(X) Passa o valor X para a subrotina de linguagem de máquina.

HIGH: Coloca à disposição o mais alto número de endereço da memória para uso no programa.

LOW: Coloca à disposição o mais baixo número de endereço da memória para uso no programa.

COMANDOS DE EDIÇÃO

LIST Lista o programa todo.

LIST X-Y Lista da linha X à linha Y do programa.

DEL X,Y Apaga da linha X à linha Y do programa.

REN ALD Usa-se para lembretes no programa, não afeta o mesmo.

VTAB Y Move o cursor para a linha Y (1 a 24).

HTAB X Move o cursor para a posição X (1 a 40).

TAB(X) Somente nos comandos de PRINT; move o cursor para a posição X.

POS(O) Retorna o cursor a posição horizontal normal (1 a 39).

PRINT(X) Comando de PRINT; coloca X espaços entre (isto item impresso e próximo).

HOME Limpa a tela e coloca cursor no canto superior.

CLEAR Coloca as variáveis em zero.

PRE(O) Fornece a quantidade de memória ainda disponível.

FLASH Coloca os caracteres em modo intermitente.

INVERSE Coloca os caracteres em modo invertido (preto em branco).

NORMAL Traz à condição normal após FLASH ou INVERSE.

SPEED=X Ajusta a velocidade de saída dos caracteres (0 a 255).

esc A Move o cursor um espaço para a direita.

esc B Move o cursor um espaço para a esquerda.

esc C Move o cursor um espaço para baixo.

esc D Move o cursor um espaço para cima.

Move o cursor um espaço para a direita e, coloca na memória o caracter sobre o qual passou.

Move o cursor um espaço para a esquerda, e apaga da memória o caracter sobre o qual passou.

ctrl X Apaga a linha sobre a qual está o cursor.

ANKA: (funções) e TRING: (sequências)

DIM A(X,Y,Z) Dimensiona o máximo espaço para A; reservam do espaço para X+1 até Z+1 elementos reais, começando em A(0,0,0).

LEN A(X,Y) Dimensiona o máximo espaço para A; pode conter X+1 elementos de string, cada um com até 255 caracteres.

LEN(A) Fornece o número de caracteres de A.

STR(X) Fornece o valor numérico de X, convertido num string.

VAL(A) Fornece o valor numérico de A, até o primeiro caracter não numérico.

CHR(X) Fornece o código ASCII do caracter.

ASC(A) Fornece o código ASCII do primeiro caracter de A.

LEFT(A,X) Fornece os X caracteres mais à esquerda de A.

RIGHT(A,X) Fornece os X caracteres mais à direita de A.

Sinal de operação para concatenar strings.

STORE A Grava o array numérico A na fita. Não pode ser usado para gravar string arrays.

RECALL B Carrega o array de (fita); o array B deve ter sido dimensionado corretamente.

GUIA DE REFERENCIA

VARIÁVEIS SIMPLES

TIPO	NOME	CAPACIDADE
Real	AB	+/- 9.99999999 e +3/
Básico	ABZ	+/- 32767
String	AB\$	0 à 255 caracteres

Onde A é uma letra, B é uma letra ou dígito. O nome pode ter mais de 2 caracteres, mas só os 2 primeiros são significativos: ABZ e ABQST são a mesma variável.

VARIÁVEIS "ARRAY"
(arranjos)

TIPO	ELEMENTO TÍPICO
Real	AB(3,12,7)
Básico	ABZ(3,12,7)
String	AB\$(3,12,7)

A dimensão do array é limitada pela memória disponível.

OPERAÇÕES ALGÉBRICAS

- Atribuição de valores
- Negação
- Exponenciação
- Multiplicação
- / Divisão
- + Adição
- Subtração

OPERAÇÕES LÓGICAS E DE RELAÇÃO

- = Igual
- <> Diferente
- < Menor que
- > Maior que
- = Menor ou igual
- = Maior ou igual

NOT "Não" lógico
AND "E" lógico
OR "Ou" lógico

Expressões lógicas ou de relação: somam valor 1 se são verdadeiras e 0 se são falsas. Estes operadores também podem ser utilizados para comparar "strings".

COMANDOS DE UTILIZAÇÃO

LOAD	Carrega um programa da fita.
SAVE	Grava um programa da memória.
NEW	Apaga o programa da memória.
RUN	Executa o programa, começando no menor número de linhas.
RUN 477	Executa o programa, começando na linha 477.
STOP	Detém a execução e informa em que linha foi detida.
END	Simplemente detém a execução sem nenhuma mensagem.
ctrl C	É usado para deter de modo imediato um programa ou listagem.
reset	(tecla RST). Retorno incondicional ao Monitor. Use ctrl C ou 96 para retornar ao básico.
CONT	Continua a execução de um programa que foi parado por STOP, END ou ctrl C.
TRACE	Lista cada número de linha e executa a linha.
NOTRACE	Anula o TRACE.
PEEK(X)	Fornece o conteúdo da memória na localização X.

POKE X,Y

Muda o conteúdo da memória na localização X, para o valor Y.

WAIT X,Y,Z

Espera até que o conteúdo da localização X quando comparado (OR) com Z e comparado (AND) com Y, dê resultado diferente de 0 (zero).

CALL X

Passa para a subrotina de linguagem de máquina, começando na localização X da memória.

USR(X)

Passa o valor X para uma subrotina de linguagem de máquina.

NIMEM:

Coloca à disposição o mais alto número de endereços da memória para uso no programa.

LOMEM:

Coloca à disposição o mais baixo número de endereços da memória para uso no programa.

COMANDOS DE EDIÇÃO

FORMATO

LIST

Lista o programa todo.

LIST X-Y

Lista da linha X à linha Y do programa.

DEL X,Y

Apaga da linha X à linha Y do programa.

REM ALO

Usa-se para lembretes no programa, não afeta o mesmo.

VTAB Y

Movimenta o cursor para a linha Y (1 a 24).

HTAB X

Movimenta o cursor para a posição X (1 a 40).

TAB(X)

Somente nos comandos de PRINT; move o cursor para a posição X.

POS(O)

Retorna o cursor a posição horizontal normal (1 a 39).

SFC(X)

Nos comandos de PRINT, coloca X espaços entre o próximo item impresso e o próximo.

HOMU

Limpa a tela e coloca o cursor no canto superior.

CLEAR

Coloca as variáveis em zero.

FRE(O)

Fornece a quantidade de memória ainda disponível.

FLASH

Coloca os caracteres em modo intermitente.

INVERSE

Coloca os caracteres em modo invertido (preto em branco).

NORMAL

Traz à condição normal após FLASH ou INVERSE.

SPEED=X

Ajusta a velocidade de saída dos caracteres (0 a 255).

esc A

Movimenta o cursor um espaço para a direita.

esc B

Movimenta o cursor um espaço para a esquerda.

esc C

Movimenta o cursor um espaço para baixo.

esc D

Movimenta o cursor um espaço para cima.

+

Movimenta o cursor um espaço para a direita e, coloca na memória o caractere sobre o qual passou.

ctrl X

Movimenta o cursor um espaço para a esquerda, e apaga da memória o caractere sobre o qual passou.

ctrl X

Apaga a linha sobre a qual está o cursor.

ARRAIZ

(arranjos)

TRIM

(sequências)

DIM A(X,Y,Z)

Dimensiona o máximo espaço para A; reservando espaço para X+1*Y+1*Z elementos reais, começando em A(0,0,0).

DIM AS(X,Y)

Dimensiona o máximo espaço para AS, que pode conter X+1*Y+1 elementos de string, cada um com até 255 caracteres.

LEN(AS)

Fornece o número de caracteres de AS.

STR\$(X)

Fornece o valor numérico de X, convertido num string.

VAL(AS)

Fornece o valor numérico de AS, até o primeiro caractere não numérico.

CHR\$(X)

Fornece o código ASCII do caractere.

ASC(AS)

Fornece o código ASCII do primeiro caractere de AS.

LEFT\$(AS,X)

Fornece os X caracteres mais à esquerda de AS.

RIGHT\$(AS,X)

Fornece os X caracteres mais à direita de AS.

+

Sinal de operação para concatenar strings.

STORE A

Grava o array numérico A na fita. Não pode ser usado para gravar string arrays.

RECALL B

Carrega o array da fita; o array B deve ter sido dimensionado corretamente.

GUIA DE REFERENCIA

VARIÁVEIS SIMPLES

TIPO	NOME	CAPACIDADE
Real	AB +/-	9.99999999 e +3/
Básico	ABI +/-	32767
String	ABS 0 à 255	caracteres

Onde A é uma letra, B é uma letra ou dígito. O nome pode ter mais de 2 caracteres, mas só os 2 primeiros são significativos: ABI e AB3Q52 são a mesma variável.

VARIÁVEIS "ARRAY"
(arranjos)

TIPO	ELEMENTO TÍPICO
Real	AB(3,12,7)
Básico	ABI(3,12,7)
String	ABS(3,12,7)

A dimensão do array é limitada pela memória disponível.

OPERAÇÕES ALGÉBRICAS

- Atribuição de valores
- Negação
- Exponenciação
- Multiplicação
- / Divisão
- Adição
- Subtração

OPERAÇÕES LÓGICAS E DE RELAÇÃO

- Igual
- <> Diferente
- > Menor que
- > Maior que
- Menor ou igual
- Maior ou igual

NOT "Não"
AND "e"
OR "ou"

Expressões lógicas ou de relação assumem valor 1 se são verdadeiras e 0 se são falsas. Estes operadores também podem ser utilizados para comparar "strings".

COMANDOS DE UTILIZAÇÃO

LOAD Carrega um programa da fita.
SAVE Grava um programa da memória.
NEW Apaga o programa da memória.
RUN Executa o programa, começando no menor número de linhas.

RUN 477 Executa o programa, começando na linha 477.

STOP Detém a execução e informa em que linha foi detida.

END Simplesmente detém a execução sem nenhuma mensagem.

ctrl C f usado para deter de modo imediato um programa ou listagem.

reset (tecla RST). Retorno incondicional ao Monitor. Use ctrl C ou SC para retornar ao BASIC.

CONT Continua a execução de um programa que foi parado por STOP, END ou ctrl C.

TRACE Lista cada número de linha e executa a linha.

NOTRACE Anula o TRACE.

PEEK(X) Fornece o conteúdo da memória na localização X.

POKE X, Y Muda o conteúdo da memória na localização X, para o valor Y.

WAIT X,Y,Z Espera até que o conteúdo da localização X quando comparado (OR) com Z e comparado (AND) com Y, dê resultado diferente de 0 (zero).

CALL X Passa para a subrotina de linguagem de máquina, começando na localização X da memória.

USR(X) Passa o valor X para uma subrotina de linguagem de máquina.

NIMEM: Coloca à disposição o mais alto número de endereço da memória para uso no programa.

LOWEM: Coloca à disposição o mais baixo número de endereço da memória para uso no programa.

COMANDOS DE EDIÇÃO

FORMATO

LIST Lista o programa todo.

LIST X-Y Lista da linha X à linha Y do programa.

DEL X,Y Apaga da linha X à linha Y do programa.

REN AL0 Usa-se para lembretes no programa, não afeta o mesmo.

VTAB Y Move o cursor para a linha Y (1 a 24).

RTAB X Move o cursor para a posição X (1 a 40).

TAB(X) Somente nos comandos de PRINT; move o cursor para a posição X.

POS(0) Retorna o cursor a posição horizontal normal (1 a 39).

SPEC(X) Nos comandos de PRINT, coloca X espaços entre o item impresso e o próximo.

HOME Limpa a tela e coloca o cursor no canto superior.

CLEAR Coloca as variáveis em zero.

PRE(0) Fornece a quantidade de memória ainda disponível.

FLASH Coloca os caracteres em modo intermitente.

INVERSE Coloca os caracteres em modo invertido (preto em branco).

NORMAL Traz à condição normal após FLASH ou INVERSE.

SPEED=X Ajusta a velocidade de saída dos caracteres (0 a 255).

esc A Move o cursor um espaço para a direita.

esc B Move o cursor um espaço para a esquerda.

esc C Move o cursor um espaço para baixo.

esc D Move o cursor um espaço para cima.

Move o cursor um espaço para a direita e, coloca na memória o carácter sobre o qual passou.

Move o cursor um espaço para a esquerda, e apaga da memória o carácter sobre o qual passou.

ctrl X Apaga a linha sobre a qual está o cursor.

ARRA: (arranjos)
TRING: (sequências)

DIM A(X,Y,Z) Dimensiona o máximo espaço para A; reserva do espaço para X; e Y; e Z; elementos reais, começando em A(0,0,0).

LEN(AS) Fornece o número de caracteres de AS.

STR\$(X) Fornece o valor numérico de X, convertido num string.

VAL(AS) Fornece o valor numérico de AS, até o primeiro carácter não numérico.

CHR\$(X) Fornece o código ASCII do carácter.

ASC(AS) Fornece o código ASCII do primeiro carácter de AS.

LEFT\$(AS,X) Fornece os X caracteres mais à esquerda de AS.

RIGHT\$(AS,X) Fornece os X caracteres mais à direita de AS.

+ Sinal de operação para concatenar strings.

STORE A Grava o array numérico A na fita. Não pode ser usado para gravar string arrays.

RECALL B Carrega o array da fita; o array B deve ter sido Dimensionado corretamente.

FIGURA 1.1

!	"	#	\$	%	&	'	(	)	*	=		
COPY	DIR	ERA	FORMAT	GBASIC	LOAD	MBASIC	PIP	SAVE	STAT	SUBMIT	TYPE	RESET
1	2	3	4	5	6	7	8	9	0	:	-	
ESC	CATALOG	STOP	STEP	RUN	TEXT	COLOR	THEN	INPUT	PEEK	PRINT	USER	RETURN
	Q	W	E	R	T	Y	U	I	O	P		
CTRL	Ä	DEF FN	STORE	DATA	FOR	GOSUB	HOME	CHR \$	NEW	LIST	+ POKE	
	A	S	D	F	G	H	J	K	L		←	→
SHIFT	RESTORE	RETURN	CALL	CONT	GOTO	NEXT	CLEAR	READ	HTAB	REM	?	SHIFT
	Z	X	C	V	B	N	M	,	.	/		
LOCK	SPACE BAR											

7	8	9	÷
4	5	6	*
1	2	3	-
0	.	'	+

Características:

- 1- 42 funções já programadas
- 2- 9 funções à serem programadas
- 3- Auto - Repeat

O Repeat do teclado é automático "AUTO-REPEAT", ou seja, pressionando qualquer tecla após 1 segundo o repeat gera 20 caracteres por segundo. Menos as teclas programadas pelo usuário.

										MILMAR - IND. COM. LTDA									
										TOLERÂNCIAS					TÍTULO				
										DIM					TECLADO INTELIGENTE MILMAR				
										ESCALA					DIMENSÕES				
										DATA					TRAT. TÉRMICO				
										DES. 5-11-85, J. CARLOS					ACABAMENTO				
										PROJ.					MATERIAL				
										APROV.					CÓDIGO DO DESENHO				
															MSE - 002				

INTRODUÇÃO

1 - LOCK

Quando esta tecla é acionada o teclado gera todos os caracteres alfabéticos em minúsculo. E pressionando a tecla **SHIFT** junto com outra tecla alfabética, irá gerar caracteres maiúsculos, menos as teclas: A, O, C que irá gerar os caracteres: ã, õ, ç.

Quando esta tecla é novamente acionada, o teclado gera todos os caracteres alfabéticos em maiúsculos. E pressionando a tecla **SHIFT** junto com outra tecla, irá gerar o caracterer superior, por exemplo:

SHIFT + **\$ 4** = \$

2 - FUNÇÕES PRÉ-PROGRAMADAS

Todas as teclas que aparecem na figura 1.1 com palavras (CATALOG, HOME, COPY, etc.) são teclas de funções pré-programadas.

Para gerar a função, pressione juntas as teclas **SHIFT**, **CTRL** e a função escolhida, por exemplo:

SHIFT + **CTRL** + **4** = FORMAT

3 - TECLAS PROGRAMÁVEIS

No seu teclado inteligente existem 09 teclas que podem ser programadas pelo usuário, com isso ele permite que voce defina seus próprios comandos de até no máximo 10 caracteres, pode fazer parte de seu comando.

Para programar seus comandos pressione juntas as teclas **SHIFT** + **USER** + 1 ou 2,3,4,5,6,7,8,9 do teclado numérico. E assim o teclado estará preparado para programar seu comando.

Exemplo:

Pressione as teclas juntas **SHIFT** + **USER** e **1**, agora é só programar: CATALOG; em seguida pressione as teclas juntas **SHIFT** + **USER** e **1**, aparecerá no vídeo, CATALOG.

4 - HABILITA E DESABILITA FUNÇÕES

Em alguns software, é necessário usar um comando específico, para isto, é necessário desabilitar o teclado inteligente.

Para isto aperte as teclas **SHIFT** + **CTRL** + **ESC**

E para habilitar novamente o teclado, o processo é o mesmo **SHIFT** + **CTRL** + **ESC**.

COD. 602001