unitron

apII 64k

COMPUTADOR

Computador
UNITRON APII 64K

Índice

CAPÍTULO 2

Inicialização à Linguagem

CAPÍTULO 3

Programação Elementar

CAPÍTULO 4

Execução de Gráficos

CAPÍTULO 5

Sequência de Símbolos, Variáveis e Agrupamentos

APÊNDICE A

Resumo dos Comandos

APÊNDICE B

Palavras Reservadas na Linguagem

Dicionário BASIC/BRASICO

APÊNDICE C

Características de Edição

APÊNDICE D

Mensagens de Erro

APÊNDICE E

Recuperação de Linguagem/Programa

APÊNDICE F

Características da Expansão

Introdução

INTRODUÇÃO

Esta introdução visa apresentar o APII 64K e esclarecer as funções das Peças de Seleção ("JUMPERS") incluídas na placa principal do mesmo. Alguns termos, como Memória, RESET, Linguagem, etc., se não compreendidos agora, não deverão constituir fonte de preocupação, pois serão explanados ao longo deste manual.

Apenas deverá ser retirada a tampa do computador (erguendo-a pela parte posterior, até desencaixá-la dos fechos de pressão, e deslizando-a) e constatar a posição dos jumpers (explicados a seguir), conforme indicado no item 3, no final desta introdução, orientando-se pela Figura 1.

1. O APII 64K possui, como parte integrante de sua placa principal, uma expansão de 16K. Para todos os efeitos, corresponde a um APII 48K com uma expansão de 16K no Conector ("slot") 0.

   Uma Peça de Seleção ("jumper"J1) permite a escolha de uma das seguintes opções:

Jumper J1 na posição 64K:

Expansão 16K ativa, como se estivesse presente no conector 0. Nesta opção, o conector 0, apesar de estar livre, não poderá ser usado, em nenhuma hipótese, pois está sob o controle da expansão 16K presente na placa principal.

Jumper J1 na posição 48K:

Memória de 48K, conector 0 livre para uso.

Um segundo jumper (J2), quando colocado na posição RESET, faz com que o RESET da máquina inclua a expansão de 16K da placa. Quando na posição sem indicação, o RESET agirá sobre a memória de 48K.

Obs.- Quando J1 estiver na opção 48K,
J2 estará sem efeito.

2. Na memória ROM foi incluída a linguagem BRASICO (BASIC em português), além do APBASIC normalmente presente.

O jumper J3 permite selecionar a linguagem: BR ou AP.

Qualquer programa em APBASIC rodará em BRASICO e vice-versa. Se estivermos em BASIC, devemos utilizar os comandos em inglês, com a respectiva sintaxe; se em BRASICO, utilizar os comandos em português, também com a respectiva sintaxe. Será necessário verificar se não foram utilizadas palavras reservadas do BASIC OU BRASICO, ao passar de uma linguagem a outra (Veja "Dicionário BASIC/BRASICO", no Apêndice B).

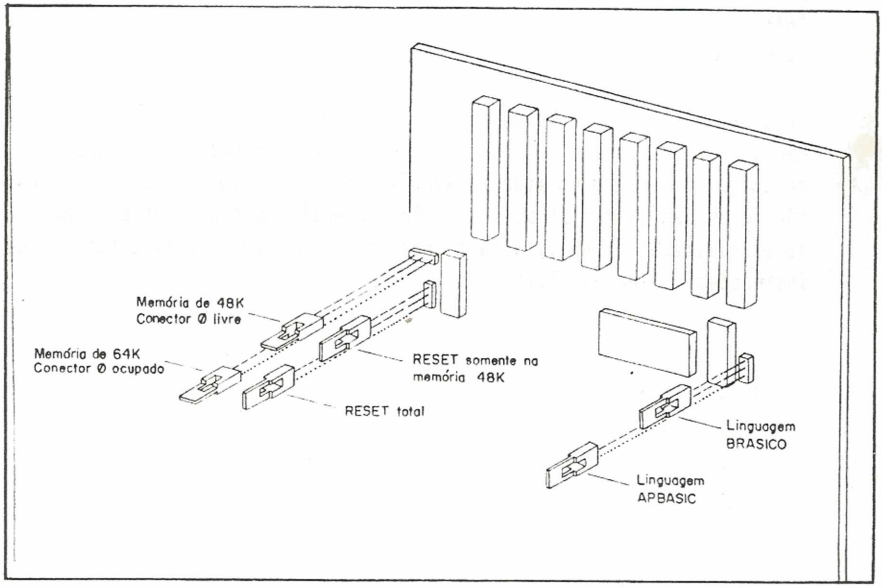
Observações:

1. Os comandos de disco (Comandos DOS) não têm nenhuma relação com o APBASIC ou BRASICO.

2. Em BRASICO, as mensagens de erro que contêm palavras da linguagem também serão exibidas em português:

"PROX." SEM "DE"
"VOLTE" SEM "SUBROT."
SEQUENCIA LONGA.

A figura a seguir ilustra as posições dos jumpers:



Placa Principal
APII 64K

Figura 1

3. O APII 64K é fornecido com os jumpers nas seguintes posições:

J1	em	"64K"
J2	em	"RESET"
J3	em	"AP"

Para acompanhar as explanações deste Manual, os jumpers deverão ser mantidos nessas posições.

Caso se deseje incluir uma Expansão (32K, 64K ou 128K) no Conector 0, o jumper J1 deverá ser colocado na posição "48K".

Caso não seja incluída uma Expansão no conector 0, mas se deseje rodar um programa que exige uma expansão de 16K, o jumper J1 deverá ser mantido na posição "64K".

Caso se deseje carregar uma linguagem na expansão de 16K, colocar J1 em "64K" e J2 na posição oposta à "RESET", de forma a manter a Linguagem carregada quando for dado um RESET ao computador.

4. Se o seu computador é do modelo T.I. (Teclado Inteligente), mantenha a tecla ESC pressionada ao ligar o interruptor de força. Este procedimento coloca o teclado no modo TRAVADO, permitindo acompanhar as explicações contidas no Manual de Instruções do APII. Após dominar por completo a utilização do APII, com o teclado "standard", leia o Manual de Instruções do Teclado T.I.

Capítulo 1

INSTALAÇÃO E USO DOS ACESSÓRIOS

CAPÍTULO 1

Instalação e Uso dos Acessórios

Introdução

Este manual irá lhe mostrar como ligar o seu computador e será um guia assim que você aprender a programá-lo. Se você já tem experiência em programação, encontrará novidades e conveniências na linguagem utilizada que tornarão o ato de programar bem mais interessante. Se você é um iniciante em programação, verá que o aprendizado é fácil e agradável; apesar disso, lembre-se que você só aprenderá se o fizer.

Se você comprou seu computador em um revendedor autorizado, terá oportunidade de aprender a ligá-lo na própria loja. Se não teve essa oportunidade também não encontrará dificuldades, pois é mais fácil que ligar um aparelho de som e nenhum conhecimento tecnológico é necessário.

Não se esqueça de mandar pelo correio o seu "CARTÃO DE REGISTRO E GARANTIA". É através dele que o seu computador será registrado em nossa fábrica e gozará da garantia de funcionamento, permitindo também que você possa receber informações adicionais sobre programas e acessórios.

O QUE VOCÊ VAI PRECISAR:

Na embalagem do computador, você também encontrará um cabo, equipado com "plugs" RCA, para conexão do computador a um monitor de vídeo.

Além desse acessório, você irá necessitar dos seguintes itens:

1 - a) uma unidade de Disco (Disk Drive) com a respectiva interface, ou
b) um gravador cassete com a fita Demonstração de Cores, e os cabos de ligação, que podem ser adquiridos nos Revendedores Unitron.
É útil possuir os dois.

2 - a) um Monitor de Vídeo colorido ou branco-e-preto, ou
    b) um televisor a cores (neste caso, você deve adquirir o Módulo de Cores PAL-M, destinado ao acoplamento computador/antena de TV a cores) ou
    c) um televisor branco-e-preto (adquirir o Modulador de RF, destinado ao acoplamento computador/antena de TV branco-e-preto).

3 - Caso queira operar programas de jogos, ou que contenham instruções gráficas especiais, deverá adquirir também:

    a) um par de controladores simples ("paddles"), ou
    b) um controlador do tipo orbital ("joystick").

CONECTANDO O MONITOR DE VÍDEO

Se você tem um monitor de vídeo branco-e-preto, basta conectar o cabo ao conector com a marca VÍDEO-SAÍDA, no painel posterior do computador.

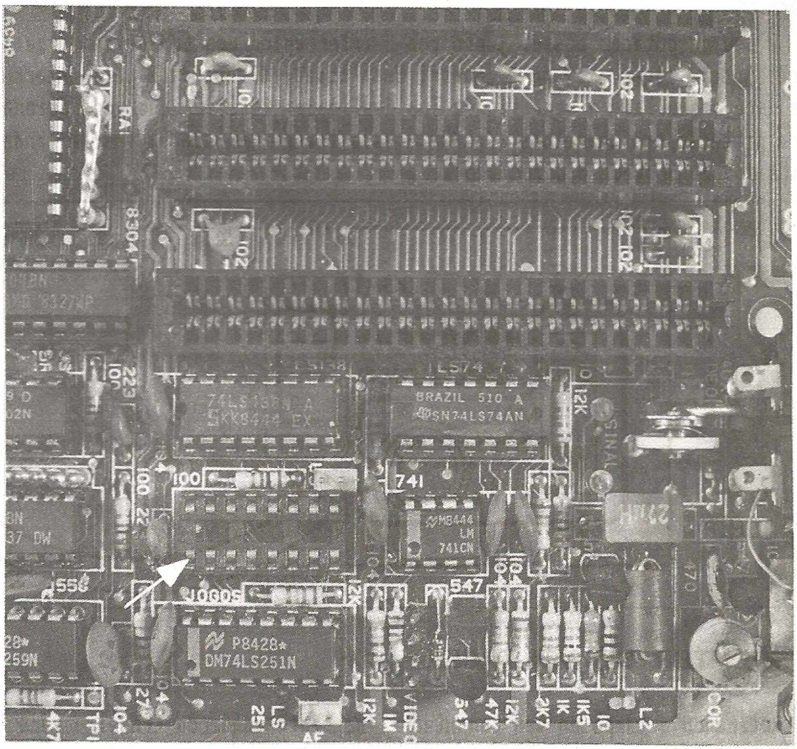
Se o monitor é do tipo colorido (ou TV a cores, ligação direta à antena) deverá ser conectado ao computador através do Módulo PAL-M, de acordo com as instruções que o acompanham.

Para ligação a um aparelho de TV comum, diretamente à antena, deve ser utilizado o Modulador de RF, da forma indicada no folheto explicativo do mesmo.

CONECTANDO OS CONTROLADORES DE JOGOS

Se você adquiriu os controladores de jogos, proceda da seguinte forma para ligá-los ao seu computador:

Com a tampa aberta, ligue os controladores no soquete situado no canto traseiro direito da placa principal do computador. Tenha bastante cuidado e certifique-se que todos os pinos entraram no soquete. A posição correta é aquela na qual o soquete tem sua marcação de "pino 1" coincidente com o pino inferior direito do soquete (marcado com um ponto branco).



A UNIDADE DE DISCO ("Disk-Drive")

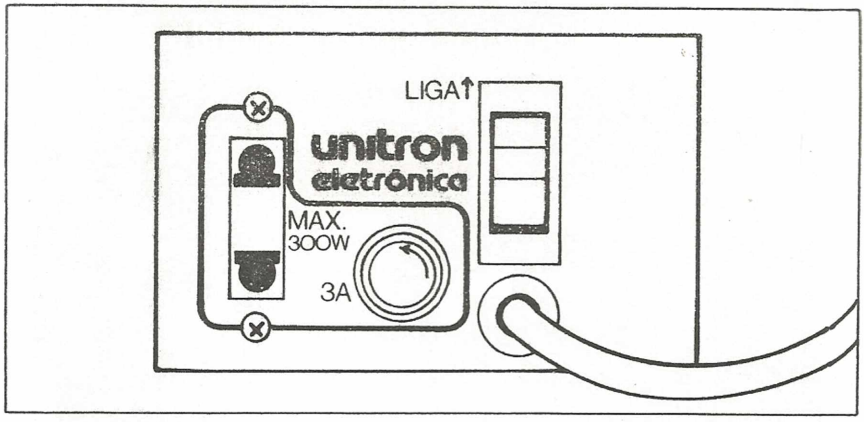
Se você tem uma unidade de disco, as primeiras páginas do manual DOS lhe darão todas as informações de como ligá-la.

O GRAVADOR

Use os cabos destinados a ligar o computador ao seu gravador. Com um dos cabos conecte a entrada MIC (ou microfone) à tomada na parte traseira do computador, onde está escrito GRAVADOR - Saída. Com o outro cabo, conecte a saída do fone de ouvido ou monitor (dependendo do tipo do gravador), e conecte o outro "plug" desse cabo na parte traseira do computador, onde está escrito GRAVADOR - Entrada.

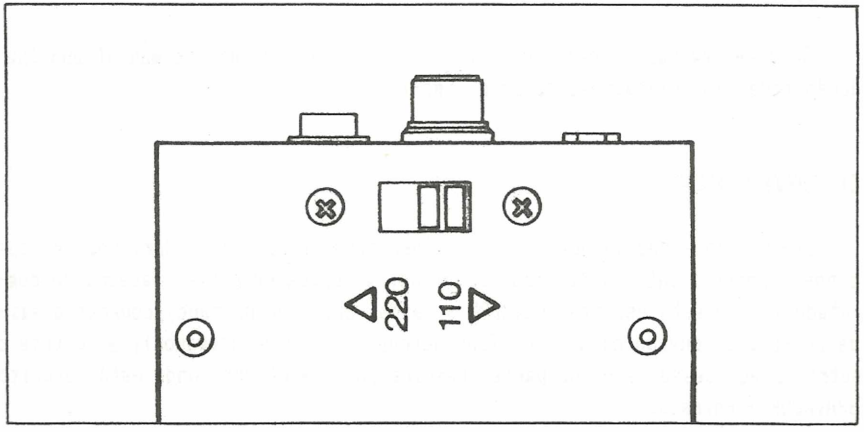
A TOMADA TRASEIRA

O seu computador possui junto com o interruptor uma tomada de força onde você poderá ligar a TV ou gravador. Essa tomada é comandada pelo interruptor que está a seu lado e será ligada (ou desligada) toda vez que você acioná-lo. Para proteção dos aparelhos ligados a essa tomada, existe um fusível independente do que protege o computador.



O SELETOR DE VOLTAGEM

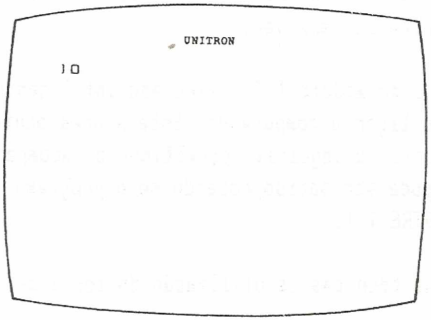
Retirando a tampa do seu computador, no canto esquerdo traseiro você encontrará o seletor de voltagem. Verifique se ele está na posição correspondente à voltagem da sua rede elétrica (110 ou 220V).



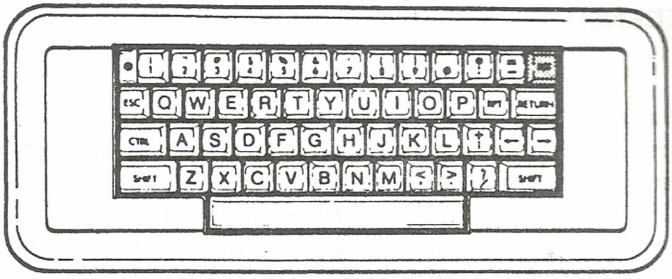
LIGANDO O COMPUTADOR e CONHECENDO O TECLADO

A primeira coisa a fazer, já que todos os acessórios estão devidamente conectados e já foi feita a verificação do seletor de voltagem, é ligar o computador. O interruptor está na parte traseira do computador, próximo ao cabo de força. Note que o "plug" de força do computador é diferente dos utilizados em outros aparelhos elétricos. Ele possui dois pinos chatos e um redondo. O pino redondo corresponde à ligação à "Terra". É importante que você ligue seu computador somente a tomadas com "Terra".

Ligue o interruptor e uma luz irá acender no teclado do seu computador, ao lado da tecla gravada com o número 1. O título UNITRON deverá aparecer na parte superior da tela juntamente com o símbolo] e um quadradinho branco piscando, chamado "CURSOR", no canto esquerdo da tela.



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 RST, situada no canto superior direito do teclado. (Ou CTRL-RST, dependendo da configuração do teclado - veja a página a seguinte). Ao soltá-la, o computador deverá fazer um "bip."



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 acender. A unidade de disco irá girar indefinidamente desde que não haja um disco inserido até que seja acionada a tecla RST. Nessa hora o UNITRON irá desaparecer do vídeo e o "CURSOR" aparecerá no canto inferior da tela.

O teclado permite a obtenção de caracteres maiúsculos e minúsculos, sendo a mudança entre essas duas condições efetuada através da tecla SHIFT, e tendo a tecla CONTROL a função "SHIFT-LOCK"; desta forma, o teclado se comporta como o de uma máquina de escrever.

Se o seu APII é do modelo T.I. (Teclado Inteligente), mantenha a tecla ESC pressionada ao ligar o computador. Este procedimento manterá o teclado no MODO TRAVADO (descrito a seguir), permitindo o acompanhamento deste manual. Este MODO também pode ser obtido rodando-se o programa TECLADO TRAVADO, presente no DISCO MESTRE T.I.

Após dominar as técnicas de utilização do teclado, leia o manual T.I.

As características descritas a seguir referem-se ao teclado Normal:

Nesta versão de teclado há 3 "jumpers" acessíveis ao usuário, e que selecionam as seguintes funções:

-J1 define a condição RESET ou CONTROL-RESET:

Com o jumper na posição "NÃO"_____RESET NORMAL

Com o jumper na posição "SIM"_____CONTROL-RESET

(Escolhendo-se esta última opção, será necessário, para obter-se RESET, pressionar simultaneamente as teclas CTRL e RESET, o que se destina a evitar RESETs acidentais).

-J2 define a condição de repetição automática (AUTO-RPT):

Com o jumper na posição "NÃO"	Não há repetição automática
Com o jumper na posição "SIM"	AUTO-REPEAT

(Escolhendo-se a opção AUTO-RPT, haverá repetição automática da tecla que for mantida pressionada)

-J3 define a condição da função "LOCK":

Com o jumper na posição "NÃO"	Não existe a função "LOCK"
Com o jumper na posição "SIM"	Valerá a função "LOCK", descrita a seguir:

A condição "default" (padrão) do teclado é MODO NORMAL (no qual obtemos todos os caracteres do código ASCII, exceto os que representam o alfabeto minúsculo) e essa condição pode ser alcançada, a qualquer momento, pressionando-se RESET.

Para entrar no MODO MINÚSCULAS (no qual obtemos todos os 128 caracteres do código ASCII), deve-se pressionar simultaneamente SHIFT e RESET, e soltar primeiramente RESET e, depois, SHIFT.

Será então válida a condição da função "LOCK" definida pelo jumper J3 em "SIM":

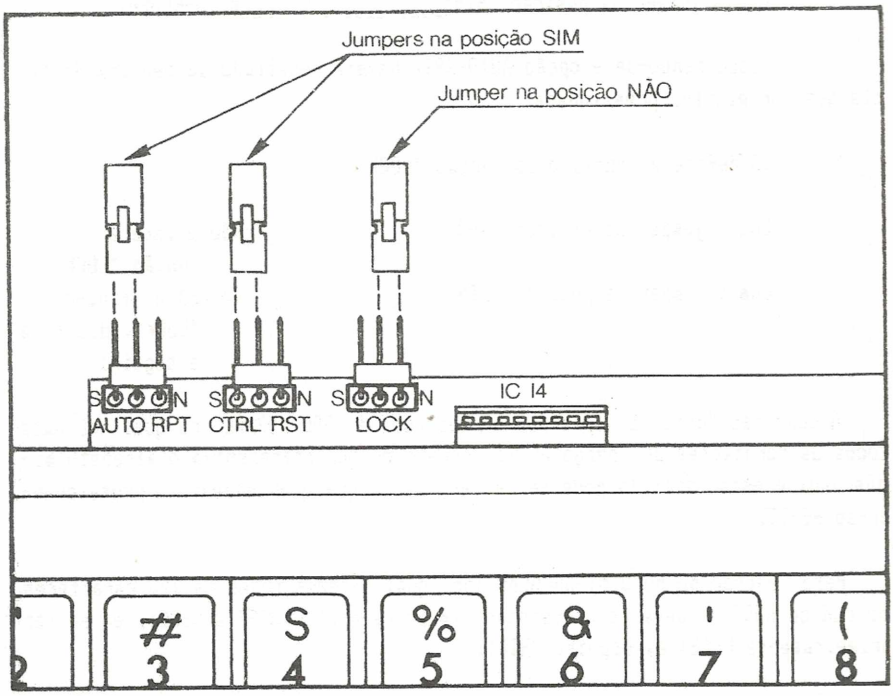
Mantendo-se pressionada a tecla SHIFT, valerá a representação das demais teclas;

Pressionando-se simultaneamente CONTROL e SHIFT, e soltando-as a seguir (primeiramente, solta-se SHIFT e, depois, CONTROL), a condição "SHIFT" se torna contínua;

Pressionando-se novamente SHIFT, retorna-se às minúsculas; ou seja, analogamente às máquinas de escrever:

Tecla CONTROL = Tecla LOCK
Tecla SHIFT = Tecla SHIFT/UNLOCK

Os jumpers localizam-se à esquerda, na interface do teclado, e são acessados simplesmente retirando-se a tampa do APII:



Com o teclado no MODO NORMAL, algumas teclas, se pressionadas juntamente com a SHIFT, farão aparecer na tela um caracter. Experimente.

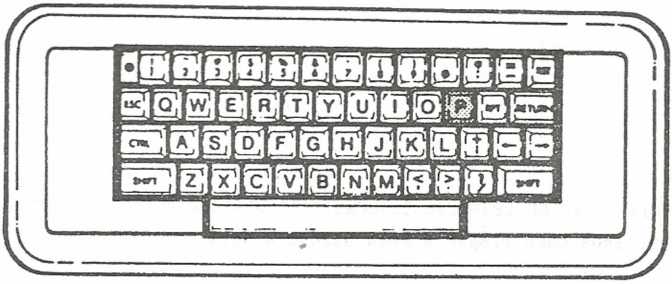
Você deve ter verificado esse fato nas teclas gravadas com dois símbolos e também nas seguintes: H I J K L M N O P U. Essas teclas, quando pressionadas, das farão aparecer na tela, respectivamente, os caracteres:

` } { [\ ] ^ _ @ ~

Uma observação importante é que o computador precisa saber distinguir o "zero" da letra "O". O método usual 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.



NOTAÇÃO DO TECLADO

Como vimos, quando uma tecla é pressionada, como a tecla da letra "A", o símbolo daquela tecla será mostrado: A .

Para indicar que várias teclas têm que ser pressionadas sucessivamente, nós simplesmente as listaremos na ordem em que devem ser pressionadas:
A L O.

Em certas ocasiões, você terá que segurar uma tecla enquanto pressiona outra. Por exemplo, para escrever o símbolo $ você precisa segurar a tecla SHIFT enquanto pressiona a tecla 4. Sempre que esta dupla atuação é necessária, nós mostraremos as duas teclas, uma sobre a outra.

SHIFT

$
4

A tecla superior deve ser segurada enquanto a inferior é pressionada. A seguir mostramos como limpar a tela usando a nova notação:

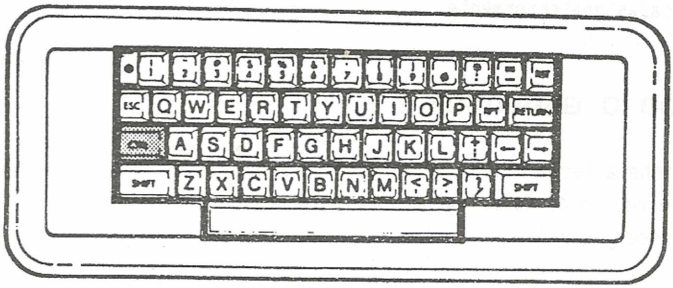
SHIFT

ESC P

Experimente.

CONTROLE E OUTRAS CARACTERÍSTICAS

Quando você pressiona a tecla 5, o numeral aparece na tela. Se você segurar a tecla SHIFT enquanto pressionar a tecla 5, o sinal de porcentagem % aparecerá na tela. A tecla SHIFT permite que algumas teclas tenham duas funções. Várias teclas, no entanto, têm uma terceira função. Essa função é obtida segurando a tecla CTRL enquanto outra tecla é pressionada. "CTRL" significa a palavra "ConTRoLe". Ao invés de gerar novos caracteres na tela, quando usamos a tecla CTRL, o computador responde executando certos comandos. Os caracteres de controle nunca aparecem na tela.



Aperte a tecla CTRL e simultaneamente aperte G.

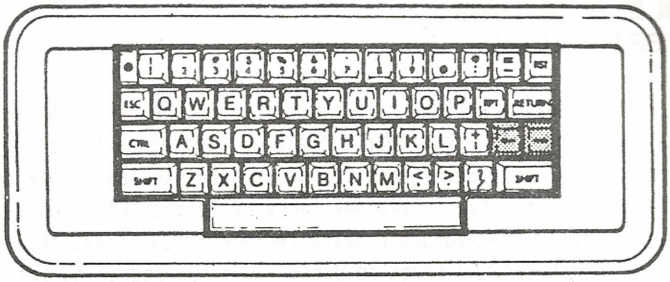
CTRL

G

Você escutará um "bip". Sempre que o computador desejar chamar a atenção ele emitirá um "bip".

Uma outra tecla que não é encontrada em máquinas de escrever é a RPT (RePeTir). Apertando o RPT enquanto teclamos um outro caracter qualquer, este aparecerá repetitivamente na tela. Você precisa teclar o caracter que deseja repetir e, simultaneamente apertar o RPT.

Temos também uma outra tecla, marcada com RETURN. Esta tecla faz o "CURSOR" retornar para o lado esquerdo da tela, além de ser uma mensagem. Se você por acaso teclar RETURN, ouvirá um "bip" e a mensagem ?GRAFIA -ERRO aparecerá na tela. Por enquanto, ignore esta mensagem.



As únicas teclas que ainda não foram mencionadas são <- e ->. Elas serão explicadas posteriormente.

LIGANDO O GRAVADOR

Pressione a tecla RETURN. O caracter] e o "CURSOR" piscando aparecerão no canto esquerdo da tela. Agora você está pronto para ajustar o controle de volume 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 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: ?GRAFIA -ERRO;
B. Nada acontece;
C. A mensagem ERRO aparece (com ou sem "bip");
D. O computador faz um "bip" e nada aparece.

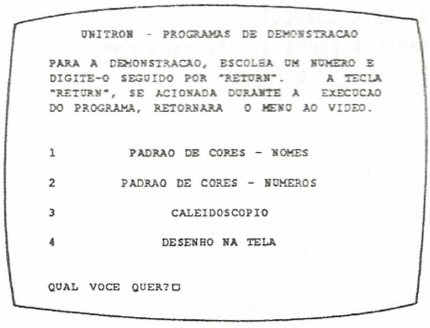
No caso A, não mude a posição do controle de volume, e volte a fita para o início, dando as instruções novamente (PASSO 1).

Nos casos B e C, após os 15 segundos, se não houver nenhum caracter nem o cursor e se o computador não responder ao teclado, aperte RST, aumente um pouco o volume e volte para o PASSO 1. Se o cursor aparecer na tela antes da fita ser rodada, simplesmente desligue e ligue 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 controle de volume 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:



PROCEDIMENTO USUAL PARA CARREGAR FITAS

Uma vez que o controle de volume 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. Nada 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 caracter] reaparecerão.

PASSO 4 - Pare o gravador e volte a fita. A informação foi transferida; por ora, você não mais usará o gravador;

PASSO 5 - Digite RUN , tecle RETURN e o programa começará a ser executado.

Várias palavras são usadas para descrever o processo de inserir e retirar informações do computador. O ato de ler a informação contida numa fita é conhecido em computação por "loading", ou seja, quando você carrega um computador com informações contidas numa fita você executa um "LOAD" da fita para o computador.

* SUGESTÃO ÚTIL

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 esse som, seguem-se ruídos, que são as informações. O som e o bip, desta forma, nos indicam o início de um programa na fita.

USANDO UMA UNIDADE DE DISCO

A Unidade de Disco ("Disk-Drive") é muito mais prática e rápida que um gravador cassete; entretanto, discos flexíveis ("diskettes") 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 Mestre" na unidade, com a etiqueta voltada para cima, e o corte oblongo para a frente, 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 no 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 "UNITRON" irá desaparecer e uma mensagem aparecerá na tela.

Se o "jumper" de seleção de memória da placa principal do APII estiver na posição "64K", (conforme explicado na introdução deste manual) aparecerá também a mensagem "CARREGANDO CARTAO DE LINGUAGEM" indicando que a linguagem Integer Basic, presente no disco, está sendo carregada para os 16K de memória extra do APII 64K, permitindo que sejam rodados programas escritos nessa linguagem.

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 DII (interface controladora de discos) não estiver inserido no 6, escreva:

SHIFT

		#	seguido do número do conector no qual o cartão
P	R	3	estiver inserido. Em seguida, tecle RETURN.

O Disco Mestre 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.

```
]
]CATALOG

APII VOLUME ØØ6

 A  ØØ6 ALO
*A  ØØ9 F
*B  Ø5Ø INTBASIC*
*A  ØØ9 DEMONSTRACAO DE CORES
*B  Ø22 INTBASIC EM DISCO
*A  Ø31 ENGENHARIA ECONOMICA
*A  Ø19 UNITRON
*B  Ø64 SABOTAGEM
*A  ØØ8 COPIA
*B  ØØ3 Copia
*I  Ø11 FORCA
*A  ØØ3 INVASORES
```

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

RUN DEMONSTRACAO DE CORES

Depois tecle RETURN. A tela deverá apresentar-se assim:

```
     UNITRON - PROGRAMAS DE DEMONSTRACAO

PARA A DEMONSTRACAO, ESCOLHA UM NUMERO E
DIGITE-O SEGUIDO POR "RETURN".     A TECLA
"RETURN", SE ACIONADA DURANTE A   EXECUCAO
DO PROGRAMA, RETORNARA   O MENU AO VIDEO.


1          PADRAO DE CORES - NOMES

2         PADRAO DE CORES - NUMEROS

3               CALEIDOSCOPIO

4               DESENHO NA TELA


QUAL VOCE QUER?
```

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:

CTRL

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: PAROU EM 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.

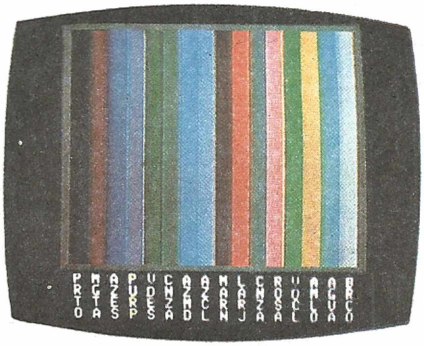
CORES

Para obtenção de imagem a cores, tanto em TV como em Monitor colorido, é necessário instalar um Módulo PAL-M, conforme as instruções que o acompanham. Se você possui um monitor monocromático, o programa Demonstração de cores, descrito a seguir, funcionará, mas as diferenças entre as cores aparecerão somente como diferenças de textura e intensidade.

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

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 delas estarão os nomes das cores básicas, abreviados por quatro letras. As cores são:

0 Preto
1 Vermelho
2 Azul escuro
3 Roxo
4 Verde escuro
5 Cinza
6 Azul
7 Azul claro
8 Marrom
9 Laranja
10 Cinza
11 Rosa
12 Verde
13 Amarelo
14 Agua (azul piscina)
15 Branco



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

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

As instruções para ajuste da TV e do Módulo de Cores PAL-M encontram-se no Manual que acompanha este último.

Capítulo 2

INICIALIZAÇÃO À LINGUAGEM

CAPÍTULO 2

Iniciação à linguagem

O seu computador possui linguagem própria, denominada BASIC AVANÇADO ou BASIC com ponto flutuante (FP BASIC). Antes de rodar um programa, verifique se ele foi escrito nessa linguagem. Se quiser utilizar linguagem diferente, é necessário a introdução da linguagem através de disco ou do uso de cartões de linguagem que contenham, por exemplo, INTEGER BASIC (INT BASIC), PASCAL, COBOL, FORTRAN, etc. Peça ao distribuidor UNITRON, onde você adquiriu seu computador, maiores detalhes sobre esses acessórios .

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 inserção do DOS, tudo estará pronto para o uso da linguagem. Escreva:

```
PRINT "INICIO"
```

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

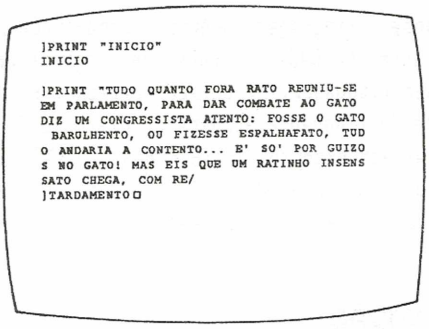
```
INICIO
```

Se não o fizer, verifique se não esqueceu de teclar RETURN. Se você escrever errado a palavra PRINT, a mensagem ?GRAFIA -ERRO 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):

```
Ø
```

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 as mensagens que desejar. No entanto, se você teclar mais de 240 caracteres, o computador começará emitir um "bip", a cada caracter teclado; depois, colocará uma barra no final da linha e deixará você começar tudo outra vez.



Agora experimente o comando:

```
PRINT "150 "
```

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

```
PRINT 150
```

e o computador novamente imprimirá o número, sem nenhuma objeção quanto à falta das aspas. De fato, o seu computador permite imprimir qualquer número sem aspas.

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

```
PRINT 3 + 4
```

e a resposta 7 aparece na próxima linha.

O seu computador pode fazer cinco operações aritméticas básicas:

1 - ADIÇÃO - Indicada pelo sinal mais +

2 - SUBTRAÇÃO - Mostrada pelo sinal menos —

3 - MULTIPLICAÇÃO - Muitas pessoas usam um "x" para representar a multiplicação; mas este símbolo poderia ser confundido com a letra "x". Outras pessoas usam um ponto (.), mas este poderia ser confundido com a indicação de números decimais. No computador, para evitar confusões, usa-se um asterisco (*) para representá-la.
Para saber quanto vale 7 vezes 8 simplesmente tecle:

```
PRINT  7 * 8
```

4 - DIVISÃO - Use uma barra (/) para indicar a divisão.
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 resposta na forma decimal: 1.5 . Note que para a separação da parte inteira das casas decimais, o computador utiliza um ponto (.) no lugar da vírgula usual.

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

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

PRINT 4 ^ 5 , o que simplifica bastante.

O caracter ^ é escrito teclando:

SHIFT

N

Não há nada de especial sobre a potenciação. Ela é simplesmente a abreviação de repetidas multiplicações.

FORMATO DOS NÚMEROS

Digite:

```
PRINT 45.340
```

Seu computador responderá com:

```
45.34
```

Os zeros de um número serão impressos se fizerem parte dos dígitos à esquerda do ponto decimal; ou se estiverem à direita desse ponto, antes do último dígito.

Números muito pequenos, entre

```
0.00000000000000000000000000000000000003
                    e
-0.00000000000000000000000000000000000003
```

serão convertidos para 0. Um método mais simples de escrever estes números é

```
3 * 10 ^ -39        e      -3 * 10 ^ -39
```

Agora escreva:

```
PRINT 985988.6898
```

Note que o número aparece arredondado na tela.

Experimente teclar:

```
PRINT 988.6898
```

O seu computador não arredondou o número. Pode parecer muito esquisito, mas o fato é que só os números com mais de nove dígitos são arredondados. O computador só dispõe de nove dígitos para operar.

Se você comandar a impressão de um número muito grande, como:

```
1234567890
```

O computador responderá com:

```
1.23456789 E + 09
```

O número aparece na tela em "notação científica", ou seja, um número fracionário multiplicado por uma potência de 10. No exemplo temos o número 1.2345678 multiplicado por 10 elevado à nona potência.

Lembre-se que qualquer número será impresso exatamente como você digitar, se este estiver entre aspas. No entanto, o computador não pode usar números entre aspas em operações aritméticas.

EXPLICAÇÕES SOBRE O RETURN

Até agora você tem pressionado RETURN após cada linha. A constante necessidade dessa tecla se deve ao fato dela ser uma instrução muito especial para o computador. Sem o RETURN, o computador não sabe que a sua instrução está completa. Por exemplo, se você começasse a escrever:

```
PRINT  4 + 5
```

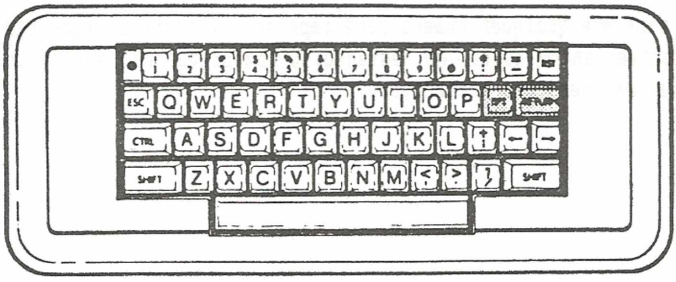
e o computador imediatamente imprimisse 9, você talvez ficasse chateado, pois tinha planejado escrever:

```
PRINT  4 + 5 + 346
```

o que daria uma resposta bem diferente. Desde que o computador não pode saber quando a sua instrução terminou, você terá que informá-lo. Isto será feito através da tecla RETURN. Dado que você tem que fazer isto após cada instrução, nós pararemos de mencionar que há necessidade de teclar RETURN. Teclar RETURN após cada instrução já deve ser um hábito, se você esteve praticando todos os exemplos.

CARACTERÍSTICAS SIMPLES DE EDIÇÃO, OU O QUE FAZER ANTES DE TECLAR RETURN

O computador possui inúmeras características que simplificam a correção de erros, evitando assim a necessidade de reescrever a linha inteira sempre que cometermos um pequeno erro. Este 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. Escreva (exatamente igual) o comando:

```
PRINT  UNITRON"
```

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

```
Ø
?GRAFIA -ERRO
```

porque esta faltando uma aspa. Se tivessemos digitado:

```
PRINT  "UNITRON"
```

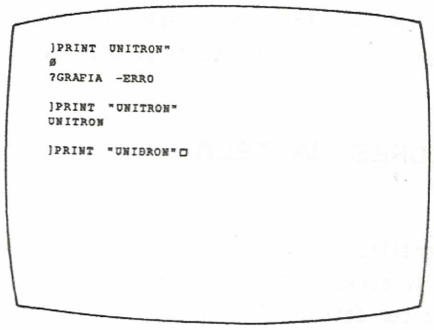
o computador responderia escrevendo UNITRON . Não acredite neste manual. Experimente.

Agora, sem teclar RETURN, escreva a instrução errada:

```
PRINT  "UNIDRON"
```

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 UNIDRON

Para UNITRON,

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

Retroceda o cursor até o D . Digite T. Como você vê, o T se coloca no lugar do D.

Agora tecle RETURN. Você obteve UNIT no lugar de UNITRON. Isto ocorreu porque você retrocedeu sobre RON. Qualquer caracter da linha em que você está escrevendo, sobre o qual o cursor retrocedeu, não é enviado ao computador quando você tecla RETURN. Uma solução seria corrigir o D através do retrocesso e então teclar:

SHIFT

*

R O N 2 RETURN Experimente.

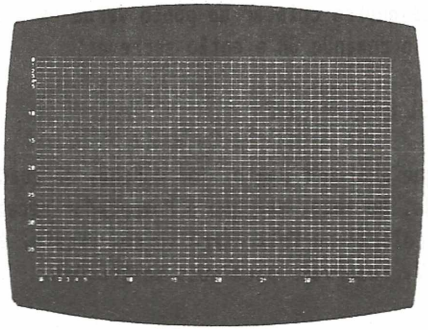
Funciona. No entanto, há um método mais simples. Quando você pressiona a tecla ->, o "CURSOR" se move para a direita. Assim que o "CURSOR" se mover para a direita, passando por cima do caracter, produzirá o mesmo efeito que reescrever o caracter. A tecla -> é conhecida por tecla de "reescrever". Novamente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

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 a queremos. Para especificar onde a 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, descobrirá que este sistema é mais cômodo 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 de 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:

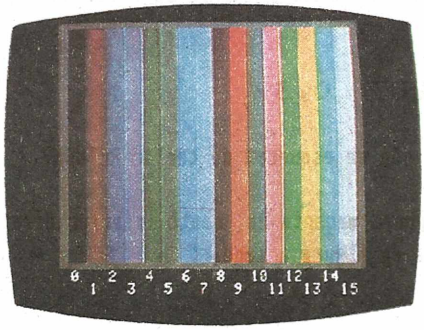
```
GR
```

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 estado anterior (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.

Para poder colocar um ponto colorido na tela, você precisa, antes, dizer ao computador qual cor deseja. Há disponibilidade de 16 cores. Você já as viu antes: elas são numeradas de 0 a 15, como indicado na "DEMONSTRACAO 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
```

Isso significa que qualquer ponto (ou mancha, ou quadrado) colorido que você fizer será verde. De fato, até que o computador receba uma outra instrução, tudo o que ele colocar na tela será verde, exceto, é claro, a 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 for colocar algo na tela, 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 se 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

?GRAFIA -ERRO

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:

?QUANT ILEGAL-ERRO

Esta mensagem indica que você tentou marcar um ponto fora da área permitida, 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 a seguir:

```
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:

```
?QUANT. ILEGAL-ERRO
```

DESENHANDO LINHAS

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

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

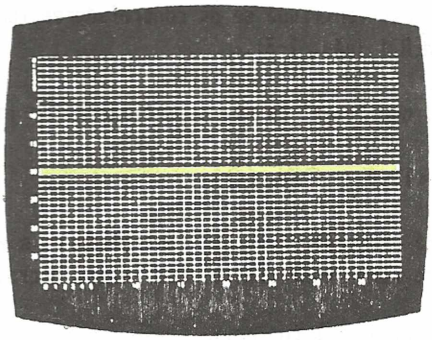
Note que as ligações entre quadrados adjacentes não aparecem; eles formam uma linha contínua. Entretanto, há um método bem mais simples de se desenhar linhas horizontais. Suponhamos que você queira desenhar uma linha horizontal verde escuro no meio da tela. Usando o caminho mais longo teríamos que escrever 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, e você terá a linha horizontal da coluna 0 à 39, ao nível do quadrado 20. Compare com a figura abaixo:



Agora tente colocar uma linha magenta desde a coluna 19 até a 28, ao nível do quadrado 18. Tente outras.

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 roxa em torno da tela, apenas com cinco instruções.

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

OS CONTROLES

Antes de iniciarmos, verifique se os controladores foram ligados conforme instruções do CAPÍTULO 1 (pág. 1-3).

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 menor mudança de posição (número) 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 é fornecido ao programa.

ARQUIVOS E OUTRAS POSSIBILIDADES DE CÁLCULO

Em calculadoras com memória, você pode guardar um número para mais tarde consultá-lo 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 77 escreva:

```
M = 77
```

O valor 77 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 = 324
```

e imprima o valor de M. E 324, certo? O que aconteceu com o 77? 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 estas duas instruções:

```
RATO TEM QUATRO PATAS
```

"RATO" TEM QUATRO LETRAS

No primeiro caso estamos nos referindo ao animal RATO. E no segundo estamos nos referindo à palavra RATO. Assim é 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. O fato do o nome de uma variável provocar a mensagem ?GRAFIA -ERRO 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 B no fim deste manual. Caso se utilize a linguagem BRASICO, veja as palavras reservadas no Dicionário BASIC/BRASICO, também no Apêndice B.

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.

Agora, experimente escrever

```
AVE = 11
```
e então
```
PRINT AVE
```

Você obteve o que esperava? Então 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 à mesma variável que:

```
AVEIA
  e
AVO
```

e assim por diante.

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

```
PRECO = 28

PRINT PRECO

PRINT 28 + 5

PRECO = 33
```

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

```
PRECO = 28

PRECO = PRECO + 5
```

Experimente agora escrever os seguintes comandos, em ordem:

```
PRECO = 2

PRINT PRECO
```

```
PRECO = PRECO + 3

PRINT PRECO

PRECO = PRECO * 6

PRINT PRECO

PRECO = PRECO / 10

PRINT PRECO
```

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

Experimente a sequência:

```
PERAS = 55

BANANAS = 11

QUOCIENTE = PERAS / 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 garçon 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 8 dividido por 2, com 4, 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 número negativo, por exemplo:

```
-3 + 2
```

o computador primeiro levará em conta o sinal de menos da variável. Então -3 + 2 resulta em -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 primeiro 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 primeiro 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 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 ten-

te 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 + 2

4 + 6 - 2 + 1

8 * 4

4 ^ 2 + 1

6 / 4 + 1

5 - 4 / 2

4 / 2 - 2

6 * -2 + 6 / 3 + 8

4 + -2

2 ^ 2 ^ 3 + 1

2 * 2 * 3 + 1

2 * 2 + 1 * 3

8 / 2  / 2 / 1

8 * 2 / 2 + 3 * 2 ^ 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 expressão 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, e não apenas para o seu computador.

```
44/(2+2)
(44/2)+2
3+(-2*2)
(3+-2)*2
100/(200/(1*(9-5)))
32/(1+(7/3)+(5/4))
```

Capítulo 3

PROGRAMAÇÃO ELEMENTAR

CAPÍTULO 3

Programação elementar

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 ?GRAFIA -ERRO), 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:

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 .

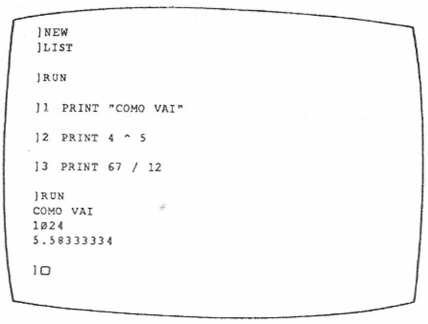
É possível guardar várias instruções, dando a cada uma delas um número diferente. Experimente escrever isto:

```
1 PRINT "COMO VAI?"
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 "N"
0 PRINT "U"
3 PRINT "T"
2 PRINT "I"
4 PRINT "R"
5 PRINT "N"
```

Note que 0 é um número de linha permitido. O maior número de linha que você pode usar é 63999. Agora, rode as instruções com o comando RUN.

O resultado deverá ser este:

```
U
N
I
T
R
N
```

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 executadas quando você tecla RUN é chamado de programa.

O programa deveria imprimir:

```
U
N
I
T
R
O
N
```

Mas parece que um comando PRINT foi esquecido. Como você pode inserí-lo no programa? A única maneira é reescrever as instruções da linha 5 e colocar uma nova linha 6. Para fazer as correções tecle isto:

```
5 PRINT "O"
6 PRINT "N"
```

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. Por isso, é usual, em programação, não escrever programas com linhas consecutivas, e deixar 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 você pode agora 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

Páginas atrás 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 algo 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 modificar 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 POSICAO"
```

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. Aproveite e marque 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 .

Modificar um programa desta maneira é um exemplo de EDIÇÃO. 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 programação.

A melhor maneira de constatar como se comporta a função PDL e de compreender os "loops", é usar um comando, muito simples, que ainda não discutimos. Escreva as linhas que 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 valor atual 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 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 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 ao 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".

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:

```
PAROU EM 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 1 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.

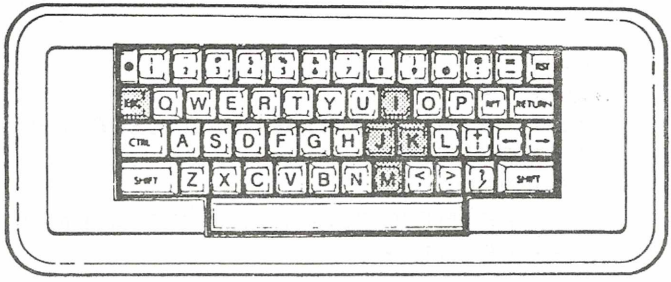
ELIMINANDO OU COPIANDO O QUE FOI ESCRITO

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

O movimento simples 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 sobre o qual passa, enquanto que as teclas retrocesso e reescrever apagam ou reescrevem os caracteres sobre os quais passam.

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

```
130 PRUNT "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:

```
?GRAFIA -ERRO EM 130
```

Para fazer a correção necessária, pode-se 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 precedem a letra "U" no comando "PRUNT". Tecle um "I" sobre o "U" e continue com a tecla de reescrever 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, 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 teclar RETURN, a linha de programa que você acabou de escrever (não os caracteres sobre os quais o retrocesso passou) serão afetados pelos movimentos da tecla retrocesso. A tecla reescrever, no entanto, irá reescrever só o caracter sobre 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, neste 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 queria eliminar a linha 1100 do seu programa, mas enganou-se e escreveu:

```
110  RETURN
```

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

```
CTRL

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 e quais são 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, tanto quanto números. Por exemplo, se introduzirmos no computador a instrução:

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

nós imprimiremos o valor 1, pois 270 não é igual a 51 (lembre-se que 1 indica que a afirmação é verdadeira). Você já viu o 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 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 à 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". Experimente 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 é falsa. Geralmente, quando você combina duas sentenças, ou afirmações com a letra E , você obterá:

a. Uma nova sentença verdadeira, se as duas originais o forem, ou,
b. 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:

a. Uma nova sentença verdadeira, se pelo menos uma das originais o for, ou
b. 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 sobre qual deverá ser a resposta:

Geralmente o comandoIF 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 restante 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 vem usando, em discos flexíveis ("diskettes"). Simplesmente faça seu programa (não esquecendo, na hora da digitação, de começar por NEW) e depois de terminado, digite:

S A V E

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 LINHAS, você escreveria:

SAVE LINHAS

Uma vez que o programa está guardado, tecle:

C A T A L O G

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 LINHAS, sempre que desejar, escrevendo:

```
RUN  LINHAS
```

Experimente guardar (SAVE) algum programa que escolher. Se você acidentalmente errar o comando, provavelmente obterá a mensagem ?GRAFIA -ERRO. Simplesmente reescreva o comando corretamente e tecle RETURN.

Obs: Para operar este comando, é necessário ter o DOS carregado (ver página 1-14).

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 LINHAS. Escreva:

```
LOAD  LINHAS
```

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 se esqueça e digite

```
RUN  LINHAS
```

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 no outro. Pratique, usando os comandos LOAD e SAVE.

GUARDANDO PROGRAMAS EM FITA "CASSETE"

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

Para guardar (SAVE) um programa que você deseja usar mais tarde, numa fita cassete, primeiro coloque um cassete virgem no seu gravador e volte-o até o começo, onde o seu programa gravado será facilmente encontrado. Acione a gravação (geralmente as teclas PLAY e RECORD pressionadas simultaneamente). De volta ao teclado, escreva:

```
S A V E
```

Quando você teclar RETURN, o cursor piscante irá desaparecer. Após 10 ou 15 segundos, o computador emitirá um "bip", indicando que a gravação terminou, e o cursor reaparecerá. Aperte a tecla STOP do 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, quando o valor de coluna atingir 40 (na tela, a máxima coluna é a 39), o programa chegou ao fim.

```
260 GOTO  210
```

e 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 os comandos REM, que 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 O
    MAXIMO VALOR DE X SER 36
270 X=X/7
280 REM LER CONTROLE UM
290 Y=PDL(1)
300 REM LIMITAR A VARIACAO 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 desde (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 serem marcadas (PLOT). Este método não utiliza a tela toda. Para utilizar toda a largura da tela, no lugar de:

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

você poderia 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 arredondado de 39.83333 para 39. Este uso do comando IF para limitar o alcance 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 de linha na memória) será executado.

A linha 200 contém o novo comando FOR. Ele começa colocando em NUMERO o valor 0. Isto é exatamente o que a linha 100 faz. Então a linha 210 é executada. O fim do "loop" está na linha 220. A variável NUMERO é acrescida de 1 e então comparada com o limite superior especificado no comando FOR (12) Se NUMERO 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 não se precisa pensar

muito para escrever um "loop", quando se usa o FOR/NEXT. Se você quisesse desenhar uma série de linhas horizontais na tela usando cada uma das 15 cores, você poderia escrever:

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

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

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

```
DEL 3000,3040
```

e então listar (LIST) o programa para conferir o resultado.

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

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

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

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

O resto do programa será como as linhas 210 a 220 das páginas anteriores, exceto pelo NUMERO que deverá ser mudado para COISA, sempre que aparecer. 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 COISA=39 TO 15 STEP -3
```

Escreva esta linha e experimente, digitando:

```
RUN 200
```

Pratique um pouco com o comando FOR, se você deseja aprender a usá-lo. Muitos 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 internos, mas não devem se cruzar. Eis aqui 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 COLUNA=0 TO 39
380 VLIN 0,39 AT COLUNA
390 NEXT COLUNA
400 NEXT C
```

Este programa é um exemplo de dois níveis internos. Pense sobre isto e rode (RUN) este programa antes de ir para o próximo. Lembre-se, quando estiver escrevendo programas usando comandos FOR, de que cada FOR deve ter um NEXT correspondente.

UM 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 TRACO=0 TO 30 STEP 10
340 C=C+1
350 IF C>15 THEN C=0
360 COLOR=C
370 FOR LINHA=TRACO TO TRACO+9
380 HLIN COLUNA,COLUNA+4 AT LINHA
390 NEXT LINHA
400 NEXT TRACO
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 em 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 NO BRANCO"
```

e rode-o. Não é bem diferente do habitual? 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 experiência, escreva este programa e veja o que ele faz quando você o rodar:

```
NEW
100 PRINT"CHARME"
110 GOTO 100
```

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

```
100 PRINT "CHARME",
```

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

```
100 PRINT "CHARME";
```

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 CHARME após CHARME, 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 usado 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. Portanto, 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 a 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 usado 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 disto, HTAB pode fazer a impressão começar à esquerda ou à direita da posição em que estava sendo feita. O caracter mais à esquerda em uma linha está na posição 1, enquanto que o mais à direita está na 40. A seguir, apresentamos 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 rodá-lo, 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:

```
?QUANT.ILEGAL-ERRO
```

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 HTAB K
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

CAPÍTULO 4
Execução de Gráficos

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 exemplo. 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 um erro 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 as primeiras. 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 listá-lo e rapidamente digitar:

```
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 interressante idéia que o computador diga o que ele está esperando, na hora em que aparecer o ponto de interrogação. Você pode adicionar estas 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
    INTERROGACAO."
380 PRINT "ENTAO PRESSIONE A
    TECLA 'RETURN'."
```

Podemos, também, 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, a interrogação não 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 teclar RETURN, receberá uma mensagem de erro. Se o caracter inserido não for um número,

```
? REENTRE
```

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

```
?QUANT. ILEGAL ERRO EM 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, desta 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 recomendamos para deixar bastante espaço entre uma linha e a outra?

É uma boa prática de programação fazer um programa tão "à 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 ?GRAFIA -ERRO, 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 separa-

dos 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
310 HOME
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
430 HOME
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 BOLA 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 sons são facilmente gerados. Você pode fazer sons no seu computador gerados por programas.

Para conseguir produzir qualquer som, 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 alto-falante do computador e foi definido no projeto do mesmo.

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 perto do computador, com atenção.

Agora adicione esta linha e experimente seu novo programa:

```
160 GOTO 150
```

Para fazer o programa emitir "bips" por um período limitado 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 som diferente", 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 e 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 ENDERECO
```

```
    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 se deseja ter mais de um comando na mesma linha.

No entanto, só o primeiro comando da linha tem um número de comando. Desta forma, um acesso a essa linha vai sempre se referir à 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 velocidade. (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 executam uma função, como a PAUSA na linha 40 acima.
5 - Requer menos memória. Isto será uma vantagem somente se você está ultrapassando o espaço de memória e o computador lhe dá a mensagem:

```
?FALTA MEMORIA -ERRO
```

ou a mensagem

```
?PROGRAMA LONGO
```

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 comando múltiplo para depois receber uma mensagem de erro como ?GRAFIA -ERRO quando o programa for rodado, sendo necessário então reescrever novamente toda a linha.

NOTAS ALEATÓRIAS ("RANDOM")

Experimente este pequeno programa:

```
NEW
100 PRINT RND(1)
110 GOTO 100
RUN
```

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

```
CTRL
 C
```

Os números gerados por este programa são frações decimais aleatórias 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 são pouco usuais. Os inteiros (números como 3, 6 e 10) são usados mais frequentemente. 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 (1)
110 REM MULTIPLICAR X POR 10
120 X = X * 10
130 REM ELIMINAR A 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. O parêntese 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, em 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 ELIMINAR A PARTE
    FRACIONARIA
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",X
170 PRINT : PRINT :  PRINT
180 FOR PAUSA=1 TO 2000 :
    NEXT PAUSA
190 GOTO 100
```

Rode este programa e veja o que ele faz.

E eis aqui um exemplo de eficiência: pode-se condensar este programa em apenas uma linha:

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

Procure explicar como essa linha 100 funciona.

SIMULANDO UM PAR DE DADOS

Você pode usar o que aprendeu sobre números aleatórios para escrever um programa que simula 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 esses "dados"? Experimente.

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

Aqui temos um meio colorido de usar inteiros aleatórios:

```
NEW
200 GR
 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 escrever um programa que desenhe linhas através da tela, em cores aleatórias?

SUB-ROTINAS

Imagine que inventamos um jogo que necessita de uma peça parecida com um cavalo azul, com patas cor de laranja e 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 lugar 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-lo novamente.

A chave para fazer isto começa com a observação de que se pode mover um ponto que está nas coordenadas (A,B) para a direita, adicionando algo ao valor da primeira coordenada (A, neste caso).

Por exemplo, o ponto (4,17) se move 10 colunas para a direita se somarmos 10 à primeira coordenada, criando o ponto (14,17).

Analogamente, um ponto se move para a esquerda se subtrairmos algo da primeira coordenada (ou adicionarmos um número negativo). Uma simples experiência mostrará que somando a, ou subtraindo de, a segunda coordenada se moverá para cima e para baixo, respectivamente.

Com esses fatos em mente, você pode reescrever o seu programa para "centrar" o cavalo em quase todos os pontos X,Y da tela. Por que "quase" todos os

pontos? Porque se você escolher um ponto central em alguma das laterais, o cavalo irá sair da tela e isto talvez lhe dê a mensagem

```
?QUANT. ILEGAL ERRO EM 1030 (ou outro número de linha)
```

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 está parte do programa para colocar vários cavalos na tela. Um GR aqui iria limpar a tela antes que cada novo cavalo fosse desenhado.

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
60
```

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:

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

```
60 GOSUB 1000
100 GOSUB 1000
1090 RETURN
```

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

```
?"RETURN" SEM "GOSUB" ERRO EM 1090
```

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

```
GOSUB 1000
```

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

VERIFICAÇÕES

A parte do programa da linha 1000 é chamada sub-rotina ou subprograma. A parte do programa da linha 20 à linha 100 é chamada programa principal.

Para ver o programa rodar, ou o curso da execução, pode-se utilizar de um de um comando especial, chamado TRACE. Esse comando especial pode lhe mostrar porque o computador forneceu uma mensagem de erro quando o programa de desenhar 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, esse programa será verificado toda vez que for rodado. Para parar o TRACE, você precisa introduzir o comando NOTRACE numa linha do programa ou em execução imediata.

UMA SUB-ROTINA PARA DESENHAR UM CAVALO MELHOR

As sub-rotinas devem ser escritas de forma a evitar que ocorram erros quando o programa é rodado. Um problema com a nossa sub-rotina 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; nossa escolha, entretanto, elimina esse incoviniente.

Caso se deseje mudar os valores em uma sub-rotina 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 sub-rotina novamente, com cores diferentes. No entanto, vamos experimentar usar variáveis no lugar de números. Na linha 1010, em lugar de COLOR=7, digite:

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

Analogamente, poderíamos escrever:

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

O programa principal pode 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 haverá um número menor de comandos, em relação ao que seria necessário para reescrever o programa todas as vezes. Para facilitar ainda mais a programação, pode-se usar o seguinte truque: criamos sub-rotinas que designam as cores para o cavalo de cada jogador, e cada uma dessas sub-rotinas chama aquela que desenha o cavalo.

```
NEW

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 LARANJA 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, somente o que precisamos para obter uma 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 duas sub-rotinas de desenhar um cavalo, que começam nas linhas 2000 e 2500, chamam uma outra sub-rotina, que começa na linha 1000. As coisas se tornam bastante eficientes a esta altura. Uma vez que você escreveu uma boa sub-rotina que verifica erros e usa variáveis cujos valores são atribuídos no programa que for chamado (talvez seja o programa principal ou uma outra sub-rotina), então você pode sobrepor ainda outras sub-rotinas a ele. Isso torna os programas principais muito mais fáceis de serem escritos. Usando as três sub-rotinas, é muito fácil colocar na tela uma atrativa exibição de cavalos.

Mas, primeiro, teste uma outra rotina útil:

```
3000 REM ESCOLHER X,Y ALEATORIOS
3010 X = INT(RND(1) * 37) + 1
3020 Y = INT(RND(1) * 38) + 1
3030 RETURN
```

E agora para o programa principal:

```
10 REM ACIONAR MODO GRAFICO
20 GR
30 REM ESCOLHER UM PONTO
    ALEATORIO
40 GOSUB 3000
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 apresentar um programa principal. Todo bom programador inclui vários REMs e GOSUBs. O trabalho deve ser feito em sub-rotinas rerelativamente curtas, cada uma delas fácil de ser escrita, e completa. Sinta-se à vontade para usar TRACE a fim de verificar como este programa trabalha. Caso tenha havido algum engano em incluir ou apagar linhas, segue a listagem completa do programa:

```
10 REM ACIONAR MODO GRAFICO
20 GR
30 REM ESCOLHER UM PONTO ALEATORIO
40 GOSUB 3000
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
1000 REM COLOCAR UM CAVALO EM
     QUALQUER LUGAR DA TELA
1010 COLOR= C: REM  C=CORPO
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
1020 PLOT X,Y - 1
1030 HLIN X,X + 2 AT Y
1040 COLOR= PE
1050 PLOT X,Y + 1
1060 PLOT X + 2,Y + 1
```

```
1070 COLOR= CARA
1080 PLOT X - 1,Y - 1
1090 RETURN
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 LARANJA 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
3000 REM ESCOLHER X,Y ALEATORIOS
3010 X = INT (RND (1) * 37) + 1
3020 Y = INT (RND (1) * 38) + 1
3030 RETURN
```

Se você preferir que a tela não seja preenchida pelos cavalos, mude a linha 120 para:

```
120 FOR N= 1 TO 500: NEXTN: GR: GOTO 30
```

Estude como funciona essa linha.

GRÁFICO DE ALTA RESOLUÇÃO

Nós chamamos o tipo de gráfico que você tem usado até então de gráfico 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. Basicamente, os comandos dos gráficos de alta resolução são os mesmos que os comandos correspondentes dos de baixa resolução, exceto pela adição da letra H (alta resolução = "high resolution"). Um bom conhecimento dos gráficos de baixa resolução será útil nesta seção.

Escreva:

```
HGR
```

Este comando limpa a tela, deixando quatro linhas, embaixo, 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	ROXO	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 as coordenadas horizontais, par e ímpar, forem marcadas.

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

```
HCOLOR=3
HPLOT 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 do que em gráficos de baixa 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 tem a fazer é escrever:

```
HPLOT TO 279,159
```

e a linha aparece no canto direito da tela. Quando se 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).

Podemos encadear estes comandos e desenhar várias linhas em uma só instrução, se assim desejarmos.

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 verticais e horizontais; desenhar linhas inclinadas em gráficos de alta resolução é também bastante simples. Para desenhar uma linha do 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, inclinações e cores.

Aqui temos um programa que torna o seu computador uma tela para desenhos em 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 sub-rotina 1000. A sub-rotina 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 sub-rotina, 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 através da posição do controle e 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 havia 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 : REM MOVER O CURSOR
    PARA ULTIMA LINHA
120 HGR : REM ACIONAR GRAFICO
    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.

Certifique-se de colocar as variáveis do comando NEXT na ordem certa, para evitar loops cruzados.

Para voltar à 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 cassete, experimente fazer o valor de HCOLOR variar aleatoriamente. Experimente desenhar primeiro linhas laranja e depois azuis, ou somente linhas azuis.

Capítulo 5

SEQUÊNCIA DE SÍMBOLOS, VARIÁVEIS E AGRUPAMENTOS

CAPÍTULO 5

Sequências de símbolos, variáveis e agrupamentos

DANDO SEQUÊNCIA

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

fará imprimir o conteúdo da 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$)
```

e 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:

```
?GRAFIA -ERRO
ou
?STRING LONGO-ERRO
```

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 do meio da sequência, ao invés do começo ou do fim, a função MID$ é a 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 em que o computador deve começar a impressão. O segundo número (9) indica quantos caracteres vem ser impressos. A instrução 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$ (NOME$,N,6)
```

Antes de seguir a 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 PRINT "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 ";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
    ";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 minunciosamente, 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 de 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$ (X$,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$. Talvez você queira ver como também o nome PEDRO ao contrário. A próxima vez que o programa solicitar um nome, escreva PEDRO em N$. A linha 140 irá então colocar em R$ o valor antigo de R$ mais N$ ao contrário: 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 computador deverá escrever 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 o seguinte:

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

O computador responderá com:

```
BOM DIA, RENATO
```

A concatenação é especialmente útil quando se deseja 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 SEQUENCIAIS

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:

```
? DIFERE O TIPO-ERRO
```

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 do conteúdo de uma sequência, e não o conteúdo real. 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:

```
? DIFERE O TIPO-ERRO
```

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 obtém. O conteúdo de Q era o que você esperava? Podemos também usar VAL para adicionar o valor numérico de 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 como 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$(P)
PRINT P$
```

Podemos ver 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 se fazer uma rotina de entrada verdeiramente "à prova de erros".

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

INTRODUÇÃO AOS AGRUPAMENTOS

Nesta seção usaremos os AGRUPAMENTOS em exemplos da matemática recreativa que não necessitarão de nada mais que aritmética elementar.

Os AGRUPAMENTOS lhe darão condições para lidar com 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 variável simples 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 possamos julgá-las desajeitadas para 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 e imprime um quadro dos conteúdos de cada elemento do AGRUPAMENTO:

```
100 REM DIMENSIONAR AGRUPAMENTO
    CHAMADO DIAS, PARA GUARDAR 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 primeiro preenche um AGRUPAMENTO com números e então mistura-os.

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

As linhas 230 a 250 preenchem o AGRUPAMENTO e designam, a cada 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 320 garante 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 a 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:

?REDIMENSIONAR -ERRO

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 adicionada ao programa.

?SUBSCRITO-ERRO

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.

?QUANT. ILEGAL-ERRO

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

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

CONCLUSAO: Este manual apresentou a base 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 muitos recursos, e, se você assimilou todos as aqui apresentados, há ainda vários para você explorar e descobrir.

Apêndice A

RESUMO DOS COMANDOS

APÊNDICE "A"

RESUMO DOS COMANDOS

Apresentamos a seguir um resumo dos comandos que podem ser usados em programação na linguagem de seu computador (APBASIC).

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 caracter 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 caracter sobre o qual ela passa é eliminado da linha do programa que está sendo escrito.

CALL-151

Provoca o aparecimento do "prompt" asterisco (*) 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 arquivos são:

I - O arquivo é um programa na linguagem BASIC INTEGRAL ("Integer BASIC").
A - O arquivo é um programa na linguagem de seu computador (APBASIC).
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 Roxo	11 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) modificado, 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 caracter inserido. O INPUT não é interrompido enquanto 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 do 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 na direção que 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 espaco para
ESC I	cima
ESC J	esquerda
ESC K	direita
ESC M	baixo

FLASH

Este comando faz com que a saída para o vídeo 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 a 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 Roxo	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 de texto da tela 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 = 10 até 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 50,60
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 da tela, até a coluna especificada (1 até 40). No exemplo dado, o cursor será posicionado na coluna 23.

```
IF  IDADE < 18  THEN A = 0 : B = 1 : C = 2
ANS$ = "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 $ ("UNITRON" , 5)
```

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

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

Fornece o número de caracteres em um string, entre 0 e 255. No primeiro exemplo, a resposta é 17.

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

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

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

O primeiro exemplo provoca o aparecimento de todo o programa na tela; 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 temporariamente, em algum ponto, a listagem, use CTRL S. Use CTRL S novamente para continuar a listagem.

LOAD

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

Quando LOAD é satisfatoriamente completado, um segundo bip será ouvido e o caracter 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.

NOTRACE

Anula o comando TRACE. Veja TRACE.

PDL (1)

Fornece o valor corrente, um número de 0 a 255, do controle respectivo. Pode-se utilizar valores de 0 a 3.

```
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 cria uma linha em branco e fornece à tela (pula uma linha). Itens de uma lista a ser impressa devem ser separados por vírgulas (,) se cada um deve ser mostrado em um campo tabulado. Os itens devem ser se-

parados por ponto e vírgulas (;) se devem 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.

```
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 subsequentes aleatórios (RND) com argumentos positivos sempre irão seguir uma sequência 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 sequência de números aleatórios iniciados 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, 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. Somente NOTRACE anula o TRACE.

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

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 10, 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= 10 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.

Apêndice B

PALAVRAS RESERVADAS DA LINGUAGEM DICIONÁRIO BASIC/BRASICO

APÊNDICE "B"

PALAVRAS RESERVADAS NA LINGUAGEM

A lista abaixo contém todas as palavras reservadas na linguagem. Embora muitas destas palavras não apareçam neste manual, a lista é útil como um guia para nomear variáveis.

```
&
ABS
AND
ASC
AT
ATN

CALL
CHR$
CLEAR
COLOR=
CONT
COS

DATA
DEF
DEL
DIM
DRAW

END
EXP

FLASH
FN
FOR
FRE

GET
GOSUB
GOTO
GR

HCOLOR=
HGR
HGR2
HIMEM:
HLIN
HOME
HPLOT
HTAB

IF
IN#
INPUT
INT
INVERSE

LEFT$
LEN
LET
LIST
LOAD
LOG
LOMEM:

MID$

NEW
NEXT
NORMAL
NOT
NOTRACE

ON
ONERR
OR

PDL
PEEK
PLOT
POKE
POP
POS
PRINT
PR#

READ
RECALL
REM
RESTORE
RESUME
RETURN
RIGHT$
RND
ROT=
RUN

SAVE
SCALE=
SCRN(
SGN
SHLOAD
SIN
SPC(
SPEED=
SQR
STEP
STOP
STORE
STR$

TAB(
TAN
TEXT
THEN
TO
TRACE

USR

VAL
VLIN
VTAB

WAIT

XPLOT
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 caracter não espaço é o sinal de substituição, = . Isto é de pouca utilidade no caso de COLOR e HCOLOR, pois como elas contém 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 caracter 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 (:) para 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
```

É 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
```

DICIONARIO BASIC/BRASICO

	A
ABS	ABS
AND	EE
ASC	ASC
AT	EM
ATN	ARCTG
	C
CALL	CHAME
CHR$	CR$
CLEAR	ZERO
COLOR=	COR=
CONT	SIGA
COS	COS
	D
DATA	DADO
DEF	DEF
DEL	APAGUE
DIM	DIM
DRAW	DES
	E
END	FIM
EXP	EXP
	F
FLASH	PISQUE
FN	FN
FOR	DE
FRE	MD
	G
GET	ACEITE
GOSUB	SUBROTINA
GOTO	LINHA
GR	GR
	H
HCOLOR=	ACOR=
HGR	AGR
HGR2	AGR2
HIMEM:	MSUP:
HLIN	LHOR
HOME	LT
HPLOT	APLOT
HTAB	TABH
	I
IF	SE
IN	IN
INPUT	ENTRE
INVERSE	INVERTA
INT	INT
	L
LEN	NC
LET	DEIXE
LIST	LISTE
LOAD	CARREGUE
LOMEM:	MINF:
	M
MID$	MEIO$
	N
NEW	NOVO
NEXT	PROXIMO
NORMAL	NORMAL
NOT	NAO
NOTRACE	SEM
	O
ON	DEP
ONERR	ERRANDO
OR	OU
	P
PDL	PDL
PEEK	VEJA
PLOT	PLOT
POKE	PONHA
POP	POP
POS	POS
PR	PR
PRINT	ESCREVA
	R
READ	LEIA
RECALL	ACHE
REM	LEM
RESTORE	RES
RESUME	RSM
RETURN	VOLTE
RIGHT$	DIR$
RND	RND
ROT=	ROT=
RUN	RODE
	S
SAVE	GRAVE
SCALE=	ESC=
SCRN(	TELA(
SGN	SN
SHLOAD	TRAGA
SIN	SEN
SPC(	ESP(
SPEED=	VEL=
SQR	RQ
STEP	PASSO
STOP	PARE
STORE	GUARDE
STR$	SEQ$
	T
TAB(	TAB(
TAN	TG
TEXT	TEXTO
THEN	ENTAO
TO	ATE
TRACE	COM
	U
USR	US
	V
VAL	VALOR
VLIN	LVER
VTAB	TABV
	W
WAIT	ESPERE
	X
XDRAW	XDES

Sinais (& + - * / ^ < = >) permanecem os mesmos.

Apêndice C

CARACTERÍSTICAS DE EDIÇÃO

APÊNDICE "C"

CARACTERÍSTICAS DE EDIÇÃO

As (teclas) setas esquerda e direita

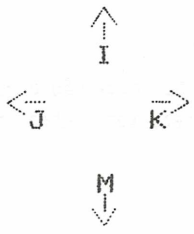
A seta esquerda (-, também chamada tecla de retrocesso, move o cursor um espaço para trás (esquerda), apagando o caracter sobre o qual passa. Se você não teclou RETURN no final da linha que escreveu, a tecla retrocesso somente afeta os caracteres naquela linha.

Pressionando a seta direita ->, também conhecida como tecla de reescrever, o cursor se move para a frente (direita) reescrevendo o caracter sobre o qual ele passa. Se você reescrever uma linha com a tecla de reescrever <- e 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 SIMPLES 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 letras acima mencionadas o cursor irá se mover de um caracter 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 também tecle RPT. O cursor irá circular através da tela enquanto ambas as teclas estiverem pressionadas. Se o cursor atingir a parte inferior da tela, ele passará, e as linhas começarão a "deslizar para cima". Se 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 programa é simplesmente escrever o número da linha que deseja apagar e depois teclar RETURN. Se você tem mais de uma linha para apagar, deve utilizar o comando DEL. Para apagar, por exemplo, as linhas de 100 a 200, escreva

```
DEL  100, 200
```

Todas as linhas de programa de 100 a 200, inclusive, serão apagadas (eliminadas).

Digitando CTRL X

a linha que você está escrevendo será eliminada. Este comando e 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 "]". O caracter "]" 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 voltar ao 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 caracter 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 e E
Limpar do cursor até o final da tela	Tecle ESC e F
Limpar a tela toda	Tecle ESC e SHIFT e P
Parar a listagem	Tecle CTRL e S
Continuar a listagem	Tecle CTRL e S

Apêndice D

MENSAGENS DE ERRO

APÊNDICE "D"

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 caracter "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 contadores de "loop" de GOSUB e FOR são 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 -ERRO
Comando de execução no programa ?XX -ERRO EM 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. Erros no programa não são detectados até que o programa seja executado.

São os seguintes os possíveis erros e seus significados:

?S/ CONTINUACAO -ERRO

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.

?DIVISAO POR ZERO -ERRO

Dividir por zero é um erro!

?FORMULA COMPLEXA -ERRO

Mais de dois comandos do tipo IF "XX" THEN foram executados.

?DIRETO ILEGAL -ERRO

Você não pode usar INPUT, DEF FN, GET ou DATA como comando de execução imediata.

?QUANT. ILEGAL -ERRO

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" SEM "FOR" -ERRO

A variável num comando NEXT não corresponde à variável num comando FOR que ainda está válido, ou um NEXT sem nome correspondeu a qualquer FOR que ainda estava válido.

?FALTA DADOS -ERRO

Um comando READ foi executado, mas todos os comandos DATA no programa foram lidos.

?FALTA MEMORIA -ERRO

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 parentêses; tentativa de comandar LOMEM: menor que o valor presente; tentativa de comandar HIMEM: muito baixo.

?S/ESPACO -ERRO

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 zero), o zero é dado como resultado e a execução continua, sem que seja apresentada mensagem de erro.

?REDIMENSIONAR -ERRO

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 "default" 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 desejar 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" SEM "GOSUB" -ERRO

Um comando RETURN foi encontrado sem que haja o correspondente GOSUB.

?"STRING" LONGO -ERRO

Foi feita uma tentativa, através de um comando de concatenação, para criar um "string" com mais de 255 caracteres.

?SUBSCRITO -ERRO

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).

?GRAFIA -ERRO

Falta de parênteses numa expressão, caracter ilegal numa linha, pontuação incorreta, etc.

?DIFERE O TIPO -ERRO

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.

?DECLARACAO INDEF. -ERRO

Tentativa de GOTO, GOSUB ou THEN a um número de linha que não existe.

?FUNCAO INDEF. -ERRO

Refere-se a uma função criada pelo usuário, que ainda não foi definida.

Apêndice E

RECUPERAÇÃO DE LINGUAGEM/PROGRAMA

APÊNDICE "E"

RECUPERAÇÃO DE LINGUAGEM/PROGRAMA

Para voltar de linguagem de máquina (#) para Basic (]):

1) Tecle RST
2) Tecle e mantenha CTRL, e a tecla B
3) Tecle RETURN

Na notação simbólica de teclas:

CTRL

RST B RETURN

RECOBRAR O PROGRAMA APÓS UM RST ACIDENTAL:

1 - Sem o disco:

Se você está usando a linguagem "BASIC", pode voltar ao seu programa após um RST acidental, teclando:

CTRL

C RETURN

Isto retornará o BASIC que você estava usando quando teclou RST, sem perder o programa.

Se você está usando Basic do cassete, após teclar RST deverá teclar

Ø G RETURN

2 - Com o disco

Se você está usando DOS (sistema de operação de disco), depois de teclar RESET você deverá teclar

3 D Ø G RETURN

Isto retornará o DOS e o BASIC que você estava usando quando teclou RESET, sem perder o programa.

CTRL

RST C RETURN

tentará recolocar a linguagem Basic para programação. Isto pode apagar as linguagens anteriores e ou programas da memória.

Apêndice F

CARACTERÍSTICAS DA EXPANSÃO

APÊNDICE "F"
CARACTERÍSTICAS DA EXPANSÃO 16K

I - INTRODUÇÃO

A expansão incluída na placa principal permite obter os 16K adicionais de memória RAM de seu computador Unitron APII 64K.

A expansão é compatível com o Módulo Unitron Z80, que permite operação com CP/M. A combinação dessas duas expansões transforma seu APII em um computador flexível (dois microprocessadores) e "full memory" (56 K).

Com as duas expansões, você tem 56K de RAM disponível para rodar quaisquer das linguagens existentes para o Z80, incluindo BASIC-80, COBOL-80, FORTRAN-80, BASIC Compiler e ALDS (Assembly Language Development System).

A expansão contém 16K de memória; entretanto, como somente são disponíveis 12K de endereçamento adicional no APII, 4K de espaço de endereçamento têm ser divididos por dois bancos de 4K. Isto significa que somente 12K da expansão são disponíveis num dado momento.

II- ENDEREÇANDO A EXPANSÃO

A informação contida neste Apêndice é de ordem técnica, sendo colocada como material de referência, somente.

Este tópico descreve o endereçamento da expansão de 16K, através de mapas de memória da mesma e das ROMs do APII.

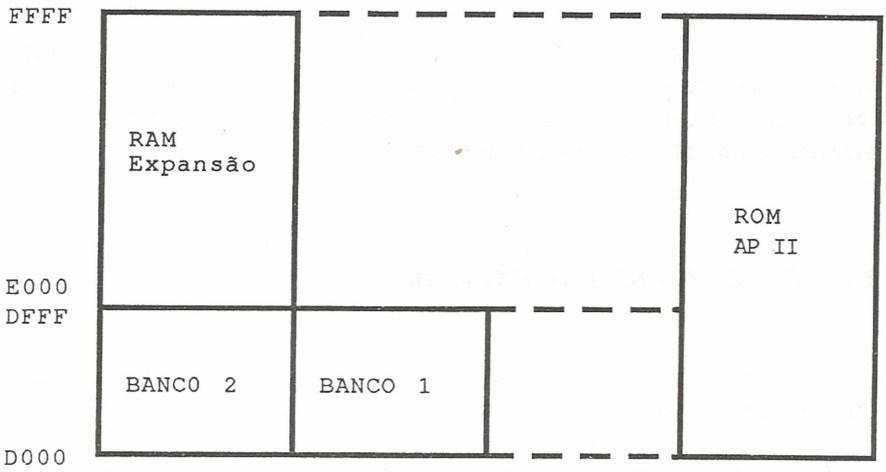
A expansão contém 16 Kbytes de memória; entretanto, como os endereços de memória de $C000 até $CFFF são usados para entrada-e-saída (I/O) do APII, somente 12K do espaço entre $D000 e $FFFF são disponíveis para endereçar os 16K expansão. Para endereçar todos os 16K, os 4K inferiores do espaço de endereçamento são usados duas vezes. O programador pode escolher qualquer um dos bancos de 4K para ocupar o espaço $D000 a $DFFF, num dado tempo. Isto

permite acessar 8K de memória num espaço de endereçamento de 4K.

A expansão 16K também divide esse espaço de memória com as ROMs do APII. O programador pode, entretanto, escolher se os endereços entre $D000 e $FFFF serão usados para ler ROM ou ler RAM da expansão, e se esses mesmos endereços serão "write-enable" ou "write-protected" para as RAMs da expansão. A seleção destas funções, mais a seleção de qual banco de 4K deve ser usado, é realizada pelo acesso aos endereços de controle.

MAPAS DE MEMÓRIA

O diagrama a seguir ilustra a extensão dos endereços usados para as funções de memória da 16K e das ROMs do APII:



ENDEREÇOS DE CONTROLE

Os endereços de controle são endereços especiais de memória que, quando acessados, controlam simultaneamente três funções:

1. Selecionam "write-protect" ou "write-enable" para as RAMs da expansão 16K
2. Selecionam RAMs da 16K ou ROMs do APII.

3. Selecionam qual dos 2 bancos de 4K deve ser mapeado no espaço de endereçamento $D000 a $DFFF. Como os endereços de $C000 a $DFFF são usados pelo APII, somente 12K ($D000 a $FFFF) do espaço de endereçamento são disponíveis para a expansão 16K. O uso os endereços de controle para alternar entre os dois bancos de 4K permite endereçar 8K em um espaço de somente 4K.

   O restante da RAM da expansão 16K ($E000 a $FFFF) é diretamente endereçáveis.

Os endereços de controle são fornecidos tanto em hexadecimal, para programas em linguagem de máquina, como em decimal, para programas em BASIC. Endereços em hexa começam pelo cifrão ($).

Todos os endereços de controle em hexadecimal têm a forma $C08x, onde x é um dos dígitos hexadecimais 0-3, 8-9, A-B. O valor de x determina o banco, e que funções são selecionadas. Quando o valor de x é convertido para binário, as funções associadas com as posições dos bits, podem ser vistas com maior clareza:

Os dígitos hexadecimais 0-3, 8-9, A-B convertem-se em binários como segue

HEXA	BINÁRIO
0	0000
1	0001
2	0010
3	0011
8	1000
9	1001
A	1010
B	1011

Os bits 0 e 1 (as duas colunas à direita em cada número binário) são lidos juntos para selecionar as funções.

Como você pode ver, só há quatro seleções de função: 00, 01, 10, 11:

00- Seleciona RAMs da expansão para leitura e protege-as contra escrita

(RAM read and RAM write-protect).

01- Seleciona leitura das ROMs do APII. Duas ou mais leituras sucessivas ao endereço habilitam as RAMs da expansão para escrita.

10- Seleciona leitura das ROMs do APII e protege as RAMs contra escrita.

11- Seleciona leitura das RAMs da expansão. Duas ou mais leituras ao endereço habilitam as RAMs para escrita.

Quando se selecionam as ROMs, estas são selecionadas para o espaço de endereçamento $D000 a $FFFF. Consulte o mapa de memória para ilustrar esta relação.

Note que é possível escrever às RAMs da expansão e ler as ROMs do APII ao mesmo tempo (condição 01 acima).

O bit 2 (terceira coluna) é ignorado.

O bit 3 (coluna à esquerda) seleciona qual dos dois bancos de 4K será acessado. Se o bit 3 é zero (x entre $0-$3), então o BANCO 2 será mapeado em $D000-$DFFF. Se o bit 3 é um (x entre $8-$B), então o BANCO 1 é mapeado em $D000-$DFFF. Quando um banco é selecionado, não há acesso ao outro.

A ligação do computador (power-on/RESET) seleciona as ROMs (desabilita a leitura das RAMs da expansão), habilita escrita das RAMs da expansão, e seleciona mapeamento do BANCO 2 no espaço $D000-$DFFF. Isto é idêntico a dois acessos consecutivos a $C081.

A tabela a seguir mostra as seleções possíveis, com seus endereços de controle em hexadecimal e decimal:

Endereço de controle	Ultimo dígito (binário)	Funções/Banco Selecionado
$C080 -16256	0000	Seleciona leitura das RAMs da expansão, Banco 2, e RAM "write-protect" na expansão.
$C081	0001	Seleciona leitura das ROMs do APII, BANCO

-16255		2, e habilita leitura das RAMs da expansão, se acessado 2 vezes consecutivas.
$C082 -16254	0010	Seleciona leitura das ROMs do APII, BANCO 2, e RAM "write-protect" na expansão.
$C083 -16253	0011	Seleciona leitura das RAMs da expansão, BANCO 2 e habilita escrita nas RAMs da expansão se acessado 2 vezes consecutivas
$C088 -16248	1000	Seleciona leitura das RAMs da expansão, BANCO 1, e RAM "write-protect" na expansão.
$C089 -16247	1001	Seleciona leitura das ROMs do APII, BANCO 1, e habilita para escrita as RAMs da expansão se acessado 2 vezes consecutivas.
$C08A -16246	1010	Seleciona das leitura ROMs do APII, BANCO 1 e RAM "write-protect" na Expansão.
$C08B -16245	1011	Seleciona leitura das RAMs da expansão, BANCO 1, e habilita para escrita as RAMs da expansão, se acessado 2 vezes consecutivas.

NOTA: Se as RAMs da expansão já estão no estado de habilitação para escrita, somente um acesso aos controles ($C081, $C083, $C089, $C08B) é necessário para permanecer no estado de escrita habilitada, até que um controle de proteção de escrita ("write-protect") seja acessado. Este estado é a condição logo após ligar o APII.

AP2.L01.44

unitron

Caixa Postal 14127 - São Paulo SP - Telex (011) 32 003 UEIC BR