TK //e

MICRODIGITAL

TK //e

Projeto e execução editorial:

Departamento de Publicações Técnicas da Microdigital Eletrônica Ltda.

MICRODIGITAL ELETRÔNICA LTDA.

Caixa Postal 54.121 - SÃO PAULO (SP)

Impresso no Brasil - Printed in Brazil

SUMÁRIO

INTRODUÇÃO 1

I - O TECLADO I-1

1. ESTRUTURA I-1
2. TECLAS QUE GERAM CARACTERES I-2
   - 2.1. Teclas não equivalentes I-2
   - 2.2. Caracteres maiúsculos e minúsculos I-2
   - 2.3. A Barra de espaço I-3
   - 2.4. A tecla delete I-4
3. A TECLA DE CONTROLE (control) I-4
4. A TECLA DE ENVIO (return) I-4
5. AS TECLAS DE MOVIMENTAÇÃO DO CURSOR I-4
   - 5.1. return I-4
   - 5.2. ← (seta à esquerda) I-4
   - 5.3. → (seta à direita) I-5
   - 5.4. ↓ (seta para baixo) I-5
   - 5.5. ↑ (seta para cima) I-5
   - 5.6. Tab I-5
6. AS TECLAS COM FUNÇÕES ESPECIAIS I-5
   - 6.1. Programando o teclado (prog) I-5
   - 6.2. A tecla mode e a acentuação I-7
   - 6.3. A tecla reset I-8
   - 6.4. A tecla círculo vazio I-8
   - 6.5. A tecla círculo cheio I-9
   - 6.6. A tecla esc e o modo ESCAPE I-9
7. AS TECLAS PRÉ-PROGRAMADAS (control - mode) I-9
   - 7.1. Funções acionadas por control+shift+tecla I-9
   - 7.2. Funções acionadas por shift+tecla I-10
   - 7.3. Delete como apagador retroativo I-10

II - PRIMEIRAS OPERAÇÕES II-1

1. O TK //e COMO CALCULADORA II-1
   - 1.1. O modo imediato II-1
2. OS CÁLCULOS II-1
   - 2.1. Soma II-2
   - 2.2. Subtração II-2
   - 2.3. Multiplicação II-3
   - 2.4. Divisão II-4
   - 2.5. Potenciação II-4
   - 2.6. Radiciação II-5
   - 2.7. Expressões algébricas II-6
3. OS NÚMEROS NO TK //e II-6
   - 3.1. Forma dos números no TK //e II-6
   - 3.2. Noções de notação científica II-8
   - 3.3. Limites numéricos no TK //e II-9

III - PROGRAMAÇÃO BASIC III-1

1. AS PRIMEIRAS INSTRUÇÕES III-1
   1.1. NEW III-1
   1.2. PRINT III-1
   1.3. HOME III-3
   1.4. END III-3
   1.5. LIST III-3
   1.6. RUN III-4
2. O MODO PROGRAMADO III-4
   2.1. A enumeração das linhas III-4
   2.2. O uso de ponto e vírgula após o comando PRINT III-5
   2.3. O primeiro programa III-6
3. O USO DE DOIS PONTOS NUMA LINHA DE PROGRAMA III-8

IV - OUTROS COMANDOS IV-1

1. ALGUNS CONCEITOS IMPORTANTES IV-1
   1.1. Os Dados IV-1
   1.2. Variáveis IV-1
   1.3. Nome que uma variável pode ter no BASIC do TK //e IV-1
2. LET IV-2
3. GOTO IV-4
4. INPUT IV-5
   4.1. Incrementando a instrução INPUT IV-6
   4.2. Uso de INPUT com variáveis numéricas IV-7
5. REM IV-8
6. INSTRUÇÃO FOR...NEXT IV-8
   6.1. FOR...NEXT IV-8
   6.2. Uso de FOR...NEXT como artifício matemático IV-9
   6.3. STEP IV-10
   6.4. Uso de FOR...NEXT como tempo de espera IV-12

V - COMANDOS DE EDIÇÃO V-1

1. CORRIGINDO E ALTERANDO A DIGITAÇÃO DOS PROGRAMAS V-1
2. A SUPRESSÃO DE LINHAS DO PROGRAMA V-1
   2.1. Supressão de uma linha V-1
   2.2. Supressão de um conjunto de linhas (DEL) V-1
3. A PERMUTA E A SUPRESSÃO DE CARACTERES V-2
   3.1. Permuta de caracteres V-2
   3.2. Supressão de caracteres V-2
4. UTILIZANDO O MODO ESCAPE (esc) V-2
   4.1. Reeditando uma linha de programa V-3
   4.2. Regras para utilização do modo ESCAPE V-4
   4.3. Outros comandos do modo ESCAPE V-4
   4.4. Os limites do modo ESCAPE V-5
   4.5. Inserindo caracteres numa linha V-5
5. RESUMO DOS COMANDOS DE EDIÇÃO V-7
   5.1. Modo ESCAPE V-7
   5.2. Modo texto V-7
6. INSTRUÇÕES RELATIVAS À FORMATAÇÃO V-7
   6.1. TAB V-7
   6.2. VTAB e HTAB V-8
   6.3. SPC V-8
   6.4. POS V-9

VI - GRUPOS DE COMANDOS E INSTRUÇÕES VI-1

1. A OPERAÇÃO COM CADEIAS VI-1
   1.1. LEN VI-1
   1.2. LEFT$ VI-2
   1.3. RIGHT$ VI-3
   1.4. MID$ VI-4
2. A OPERAÇÃO COM DADOS NUMÉRICOS VI-5
   2.1. RND VI-5
   2.2. INT VI-6
3. OS COMANDOS RELACIONADOS À EXECUÇÃO DE PROGRAMAS VI-7
   3.1. STOP, CONT e END VI-7
   3.2. Control-C VI-9
   3.3. TRACE e NOTRACE VI-9
   3.4. CLEAR VI-10
   3.5. FRE (Ø) VI-10
   3.6. INVERSE e NORMAL VI-11
   3.7. FLASH VI-11
   3.8. SPEED VI-11
   3.9. Control-X VI-12
4. ACESSO À MEMÓRIA RAM VI-12
   4.1. POKE VI-12
   4.2. PEEK VI-12

VII - OPERAÇÕES NUMÉRICAS, COMPARATIVAS E LÓGICAS VII-1

1. AS OPERAÇÕES NUMÉRICAS VII-1
2. AS OPERAÇÕES COMPARATIVAS VII-1
3. AS OPERAÇÕES LÓGICAS VII-2
4. OUTRAS OPERAÇÕES COM CADEIAS VII-4
   4.1. STR$ e + ... VII-4
   4.2. VAL(A$) VII-6
   4.3. O código ASCII VII-6
   4.4. ASC(A$) VII-7
   4.5. CHR$(X) VII-8
5. AS MATRIZES VII-9
   5.1. DIM VII-11
   5.2. Matrizes unidimensionais VII-11
   5.3. Matrizes bidimensionais VII-13
   5.4. Matrizes tridimensionais VII-13

VIII - INSTRUÇÕES DE ENTRADA E SAÍDA VIII-1

1. DATA...READ VIII-1
2. RESTORE VIII-4
3. GET VIII-6
4. DEF FN VIII-6

IX - LIDANDO COM DESVIOS IX-1

1. OS DESVIOS CONDICIONAIS IX-1
   1.1. IF...GOTO IX-1
   1.2. IF...THEN IX-1
   1.3. ON...GOTO IX-2
   1.4. IF...THEN...GOTO IX-2
2. AS SUB-ROTINAS IX-3
   2.1. GOSUB e RETURN IX-3
   2.2. Sub-rotina dentro de uma sub-rotina IX-5

2.3. ON...GOSUB IX-5
2.4. ONERR GOTO IX-6
2.5. POP IX-8

X - AS FUNÇÕES MATEMÁTICAS X-1

1. FUNÇÕES TRIGONOMÉTRICAS X-1
1.1. Função seno - SIN(X) X-1
1.2. Função cosseno - COS(X) X-2
1.3. Função tangente - TAN(X) X-2
2. OS TRAÇADOS GRÁFICOS DE FUNÇÕES X-3
2.1. Diagrama simples X-3
2.2. Gráfico da função seno X-4
2.3. Gráfico da função cosseno X-5
2.4. As outras funções (RND-ABS-ATN-SGN EXP-LOG-SQR) X-5

XI - ARMAZENAMENTO EM DISQUETE XI-1

1. SALVANDO PROGRAMAS EM DISQUETE XI-1
1.1. Inicializando o disquete (INIT) XI-1
1.2. Examinando o conteúdo do diretório (CATALOG) XI-2
1.3. Introduzindo um programa de abertura XI-2
1.4. Usando o comando SAVE XI-3
2. CARREGANDO UM PROGRAMA GRAVADO EM DISQUETE (LOAD) XI-4

XII - ARMAZENAMENTO EM FITA CASSETE XII-1

1. GRAVANDO PROGRAMAS EM FITA MAGNÉTICA (SAVE) XII-1
2. CARREGANDO PROGRAMAS DE UMA FITA (LOAD) XII-1
3. GRAVANDO E RECUPERANDO DADOS NUMÉRICOS XII-2
3.1. STORE XII-2
3.2. RECALL XII-3

XIII - GRÁFICOS E FIGURAS XIII-1

1. TRAÇADOS DE BAIXA-RESOLUÇÃO XIII-1
1.1. COLOR XIII-1
1.2. GR - PLOT - TEXT XIII-1
1.3. HLIN - VLIN XIII-2
1.4. SCRN XIII-3
2. TRAÇADOS DE ALTA-RESOLUÇÃO XIII-3
2.1. HCOLOR XIII-3
2.2. HPLOT - HGR - TEXT XIII-4
2.3. HPLOT X1,Y1 TO X2,Y2 ..... XIII-6
2.4. HGR2 ..... XIII-7

XIV - LINGUAGEM DE MÁQUINA XIV-1

1. USO DE ROTINAS EM LINGUAGEM DE MÁQUINA XIV-1
1.1. CALL XIV-1
1.2. WAIT XIV-1
1.3. USR(X) XIV-2
1.4. HIMEM: XIV-2
1.5. LOMEM: XIV-2
2. SISTEMA MONITOR XIV-2
2.1. Entrando no Monitor XIV-3
2.2. Examinando a memória XIV-3
2.3. Alterando o conteúdo da memória XIV-5

2.4. Transferindo um bloco de memória XIV-6
2.5. Comparando dois blocos de memória XIV-7
2.6. Soma e subtração de valores hexadecimais XIV-7
2.7. Salvando e carregando blocos de memória em fita cassete XIV-8
2.8. Salvando e carregando blocos de memória em disquete..... XIV-8
2.9. Criando e executando programas em linguagem de máquina .. XIV-10
2.10. Outros comandos..... XIV-10
2.11. Examinando registradores XIV-11
3. O MINI ASSEMBLER XIV-11
3.1. Acessando o Mini Assembler..... XIV-12
3.2. Usando o Mini Assembler XIV-12
3.3. Saindo do Mini Assembler..... XIV-12
3.4. Digitando um programa em Mini Assembler..... XIV-13

XV - MANIPULAÇÃO DE FIGURAS EM ALTA-RESOLUÇÃO..... XV-1

1. INTRODUÇÃO XV-1
1.1. Modo Monitor XV-6
1.2. Carga da tabela de figuras XV-6
1.3. Indicando ao BASIC o endereço da tabela de figuras XV-7
1.4. Proteção da tabela de figuras XV-7
1.5. Armazenando uma tabela de figuras e SHLOAD..... XV-7
2. USANDO A TABELA DE FIGURAS XV-8
2.1. DRAW..... XV-9
2.2. XDRAW..... XV-9
2.3. ROT e SCALE..... XV-9
2.4. Definição de três figuras..... XV-10
2.5. SHLOAD XV-12

XVI - EFEITOS SONOROS XVI-1

1. O SOM NO BASIC XVI-1
2. USANDO O PEEK (-16336)..... XVI-1
3. SOM EM LINGUAGEM DE MÁQUINA XVI-2

APÊNDICES

A - FUNÇÕES TRIGONOMÉTRICAS..... A-1
B - MENSAGENS DE ERRO B-1
C - ECONOMIA DE ESPAÇO E DE TEMPO C-1
D - CÓDIGOS DOS COMANDOS D-1
E - PALAVRAS RESERVADAS..... E-1
F - CÓDIGO ASCII E CARACTERES ACENTUADOS F-1
G - MAPA DE MEMÓRIA..... G-1
H - O CONJUNTO DE INSTRUÇÕES DO 65C02..... H-1

GLOSSÁRIO i-1

ÍNDICE ALFABÉTICO - REMISSIVO ii-1

INTRODUÇÃO

Este manual tem por objetivo ensinar os fundamentos do BASIC, permitindo ao usuário de microcomputadores da linha TK //e familiarizar-se com esta linguagem.

Os microcomputadores TK //e podem operar com outras linguagens, porém o BASIC (Begginer's All Pourpose Symbolic Code) é o código mais difundido entre os computadores pessoais, devido à sua vasta aplicação e à facilidade de assimilação por parte de usuários leigos.

Seguindo a teoria e a prática proposta neste livro, você conhecerá todos os comandos e instruções essenciais para que possa "conversar" com o seu computador e até mesmo com outros "que falem a mesma língua".

I - O TECLADO

1. Estrutura

O teclado do TK //e é o meio pelo qual o usuário comunica-se com a máquina. Como dispositivo de entrada de dados, serve ao envio de caracteres à tela e à memória do computador.

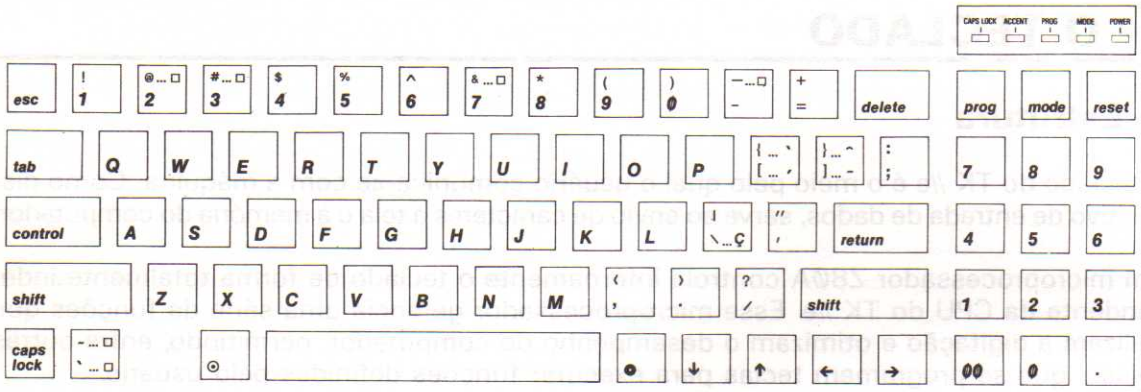
Um microprocessador Z8ØA controla internamente o teclado de forma totalmente independente da CPU do TK //e. Esse microprocessador gerencia uma série de funções que agilizam a digitação e otimizam o desempenho do computador, permitindo, entre outras coisas, que se programem teclas para executar funções definidas pelo usuário.

O teclado possui uma memória interna que também funciona como um buffer, isto é, armazena temporariamente o que é digitado, enquanto a CPU trabalha em outra função. Assim, os dados digitados são considerados pelo computador no momento em que a CPU está livre, ainda que, quando da digitação, o TK //e esteja realizando alguma tarefa, por exemplo, compondo uma figura na tela.

Para a CPU, não existe diferença entre o que é digitado no momento e o que foi armazenado no buffer. O microprocessador da CPU, o 65CØ2, "ignora" que existe um Z8ØA com memória independente. Tudo o que o Z8ØA faz é interpretado como se tivesse sido digitado letra por letra no teclado, por um digitador extremamente veloz.

Obs.: Existem alguns softwares que possuem controlador de teclado. Quando são utilizados, podem prejudicar o correto funcionamento do buffer do teclado.

No TK //e existem 77 teclas, sendo que a maioria delas desempenha mais de uma função. As teclas estão separadas em dois blocos distintos: um bloco maior, destinado a gerar caracteres alfanuméricos, simbólicos, gráficos e de controle; e outro menor, destinado a produzir caracteres numéricos. Na fileira superior deste bloco, situam-se três teclas não numéricas, com funções especiais.



2. Teclas que Geram Caracteres

Os caracteres gerados por estas teclas podem ser classificados em: alfabéticos, numéricos, simbólicos, gráficos e de funções. O TK //e é capaz de gerar 96 caracteres visíveis e 32 caracteres de controle.

A disposição das teclas de caracteres do TK //e é semelhante à das máquinas de escrever. O bloco numérico, por sua vez, assemelha-se ao teclado de uma calculadora. Quando uma tecla é mantida pressionada, o caractere digitado auto-repete-se. Estas características fazem com que um datilógrafo logo se habitue com o computador e sinta-se descontraído ao digitar.

Além dos símbolos normalmente usados na máquina de escrever (sinais de pontuação, $, &, etc.), o TK //e ainda permite a emissão dos sinais característicos de computação: <, >, #, @, ^, etc.

2.1. Teclas não equivalentes

Diferentemente do que ocorre em datilografia, o zero (Ø) e o caractere "O" não podem ser tratados como caracteres equivalentes. Para distingui-los, o computador apresenta na tela os zeros cortados por um traço. De maneira análoga, o número 1 e a letra "l" minúscula não devem ser confundidos. Também as aspas não devem ser substituídas por dois apóstrofos. Os demais caracteres são idênticos aos convencionais.

2.2. Caracteres maiúsculos e minúsculos

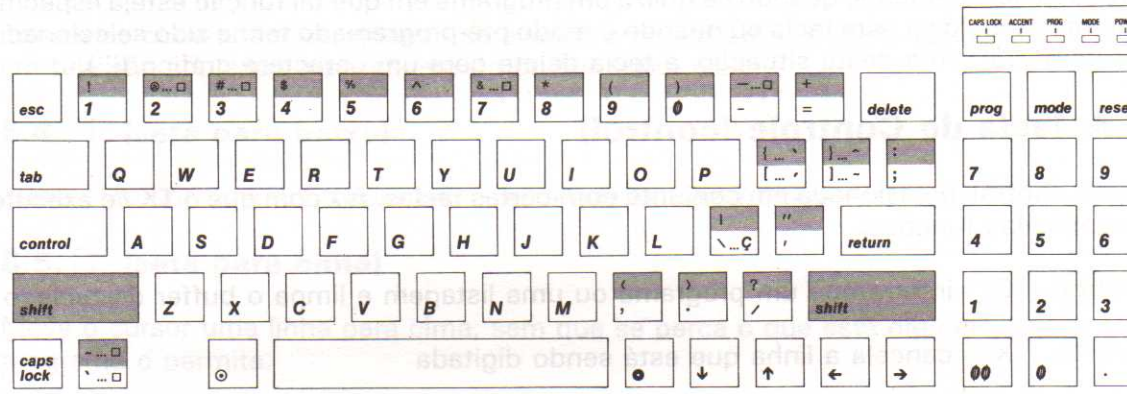
O TK //e pode produzir todos os caracteres alfabéticos, tanto em maiúscula como em minúscula. O modo padrão é o de letras minúsculas. Quando o computador é ligado, este modo entra em ação. Para obterem-se as letras maiúsculas, podem-se usar dois métodos:

a) pressionar a tecla [shift] e a tecla do caractere desejado; ou

b) ativar o modo caps lock.

2.2.1. As teclas shift

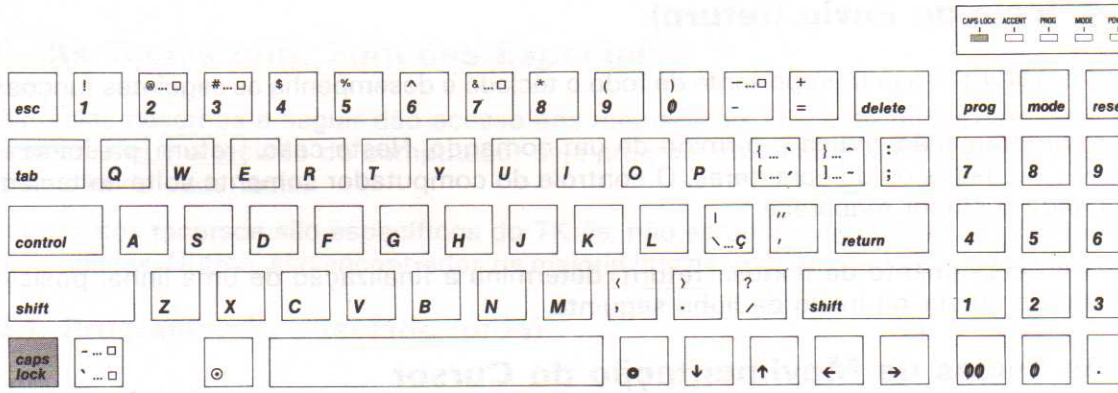
O teclado do TK //e possui duas teclas [shift]. Ambas têm a mesma função: produzir caracteres maiúsculos ou dar acesso aos caracteres representados na parte superior de algumas teclas.



2.2.2. O modo CAPS LOCK

Neste modo o teclado produz apenas caracteres maiúsculos. Para acioná-lo, deve-se pressionar uma vez a tecla [caps lock], situada na parte inferior esquerda do teclado.

Quando caps lock está ativado, o led correspondente fica aceso.



Para desativar o modo caps lock, basta que se pressione novamente a tecla [caps lock].

Nota: O modo CAPS LOCK não dá acesso aos caracteres representados na parte superior de determinadas teclas, sendo necessário, neste caso, o uso da tecla [shift].

2.3. A barra de espaço

Esta tecla destina-se à produção de espaços em branco. O TK //e considera cada espaço gerado por esta tecla como um caractere.

Geralmente o computador encarrega-se de eliminar espaços excedentes, quando estes não possuem uma função específica.

2.4. A tecla delete

Esta tecla é muito especial e bastante característica do TK //e. Funciona como apagador retroativo de caracteres, quando se utiliza um programa em que tal função esteja especificamente atribuída a esta tecla ou quando o modo pré-programado tenha sido selecionado (ver item 7.3). Fora desta situação, a tecla delete gera um caractere gráfico (▯).

3. A Tecla de Controle (control)

A tecla control, pressionada em conjunto com certas teclas, faz com que o TK //e execute determinadas funções:

[control] [C] - interrompe um programa ou uma listagem e limpa o buffer do teclado

[control] [X] - cancela a linha que está sendo digitada

[control] [G] - gera um "beep" no alto-falante

[control] [S] - suspende a impressão de caracteres.

[control] [B] - retorna à linguagem BASIC quando se está operando em linguagem de máquina

As combinações da tecla [control] com outras teclas podem ser utilizadas em programas, para finalidades diversas.

4. A Tecla de Envio (return)

A tecla [return] é a mais importante de todo o teclado e desempenha as seguintes funções:

a) Em programação, indica o término de um comando. Neste caso, [return] presta-se ao envio de dados ou de caracteres. O controle do computador somente volta ao teclado quando o cursor reaparece.

b) Em processamento de textos, [return] determina a finalização de uma linha, posicionando o cursor no início da linha seguinte.

5. As Teclas de Movimentação do Cursor

A posição do cursor ao longo da tela pode ser determinada por controles diversos e pela utilização das seguintes teclas:

5.1. [return]

Move o cursor para o início da próxima linha.

5.2. [←] (seta à esquerda)

Move o cursor uma posição à esquerda.

Quando se retrocede o cursor através desta tecla e pressiona-se [return], os dados à direita do cursor são ignorados pelo computador.

5.3. [→] (seta à direita)

Move o cursor uma posição à direita.

Em certas circunstâncias, quando se avança o cursor através desta tecla, os caracteres em tela são copiados à passagem do cursor sobre eles.

5.4. [↓] (seta para baixo)

Move o cursor uma linha para baixo, sem que se perca o que está digitado.

5.5. [↑] (seta para cima)

Move o cursor uma linha para cima, sem que se perca o que está digitado, desde que o programa o permita.

No modo BASIC normal, esta seta não move o cursor, apesar de enviar ao computador o código ASCII correspondente.

5.6. [tab]

A tecla [tab] no BASIC, por si só, não exerce nenhuma função específica. Todavia, se o computador está operando determinados processadores de texto ou programas similares, esta tecla recebe a incumbência de mover o cursor rapidamente até uma posição predeterminada. O local da tabulação pode ser determinado pelo próprio programa ou pelo usuário, se assim for permitido.

6. As Teclas com Funções Especiais

As teclas descritas a seguir dão acesso aos recursos do TK //e de maior destaque. Elas permitem que o teclado do computador seja programado, multiplicando, dessa maneira, a função de cada tecla.

Alguns dos recursos são específicos do TK //e, não estando disponíveis em computadores similares. Outros, são encontrados na maioria dos microcomputadores da família Apple.

6.1. Programando o teclado (prog)

O TK //e permite ao usuário programar o teclado da maneira que bem entender.

Programar o teclado significa atribuir às teclas macrofunções, isto é, funções que elas não possuem originalmente. Por exemplo, programar a tecla [R] de maneira que toda vez que ela seja tocada, em vez do caractere R, gere uma seqüência de caracteres, cujo significado para o computador seja uma ordem. Se é especificado que tal tecla deve gerar a seqüência RUN (comando BASIC que determina a execução de um programa em memória), quando se pressiona [R], este comando é digitado automaticamente, bastando bater [return], para que a ordem RUN seja executada. Pode-se ir além, incluindo o acionamento da tecla [return] na mesma programação. Desse modo, quando se tecla [R], além de dar-se a digitação automática do comando, processa-se também sua execução.

São os seguintes os passos necessários para programar-se uma tecla:

a) Pressionar uma vez a tecla prog (localizada acima do bloco numérico do teclado). Isto faz piscar o led PROG, indicando que o computador está à espera do caractere a ser programado.

b) Pressionar uma vez a tecla a ser programada. O led PROG pára de piscar, permanecendo aceso.

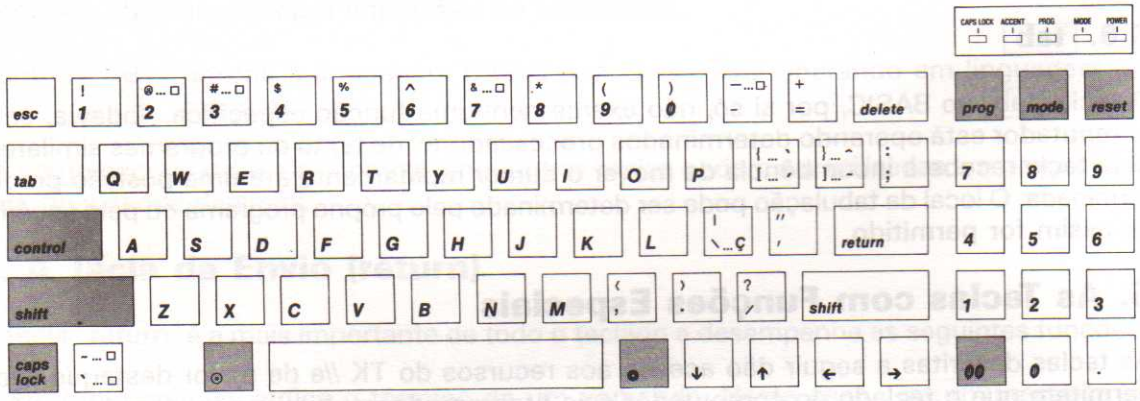
Nota: Os caracteres digitados durante a programação do teclado não aparecem na tela.

c) Digitar a seqüência de caracteres desejada. Teclar return, caso se deseje incluir também este comando na programação da tecla.

d) Finalizar o processo, desativando o modo PROG: bate-se uma vez a tecla prog. O led correspondente apaga-se. A tecla está programada.

Todas as teclas do TK //e podem ser programadas, com exceção das seguintes:

caps lock | prog | mode | ØØ | shift | control | ◉ | ● | reset



Atenção: Não podem ser usadas macrofunções na programação de uma nova tecla (nem na geração de uma nova macrofunção).

Quando o sistema de auto-repetição de caracteres é utilizado numa tecla programada, apenas o último caractere da seqüência é repetido.

6.1.1. Anulando a programação do teclado

Para cancelar a programação do teclado, pode-se agir de duas maneiras distintas:

a) Pressionar as teclas control e prog conjuntamente. Este processo limpa o buffer do teclado, cancelando a programação de todas as teclas de uma vez e desativando CAPS LOCK e MODE.

b) Usar o processo descrito para programar, mas em lugar da digitação de comandos e/ou seqüências, bater uma vez a tecla que se deseja desprogramar (se acidentalmente a tecla em questão é batida duas vezes, tem sua função duplicada, ocupando espaço na memória, desnecessariamente). No exemplo da tecla [R], faça [prog] - [R] - [prog].

6.1.2. Salvando e carregando a programação do teclado

Para salvar ou carregar a programação do teclado, consulte as instruções que acompanham o Disquete de Apresentação.

6.2. A tecla mode e a acentuação

Além de o teclado do TK //e ser passível de programação, é inteiramente adaptado à língua portuguesa, visto que integra os sinais gráficos próprios de nosso idioma: os acentos, o til e a cedilha.

Nota: Somente os caracteres minúsculos em modo normal podem ser acentuados. Existem artifícios para obterem-se na impressora maiúsculas acentuadas.

A tecla [mode], acionada, faz acender o led do mesmo nome e dá acesso ao modo acentuado disponível no TK //e.

Para indicar que está operando no modo acentuado, o TK //e substitui o sinal de prompt (]) por um cê-cedilha (ç).

Para obterem-se caracteres acentuados, deve-se proceder da seguinte maneira:

a) Acionar o modo acentuado, batendo uma vez a tecla mode. O led [MODE] acende-se e o sinal "ç" aparece, confirmando o acesso ao modo acentuado.

b) Digitar o acento antes do caractere a ser acentuado. Quando se toca uma tecla contendo um acento gráfico, o led ACCENT ilumina-se.

c) Digitar o caractere a ser acentuado. O led ACCENT apaga-se, e na tela surge o caractere acentuado.

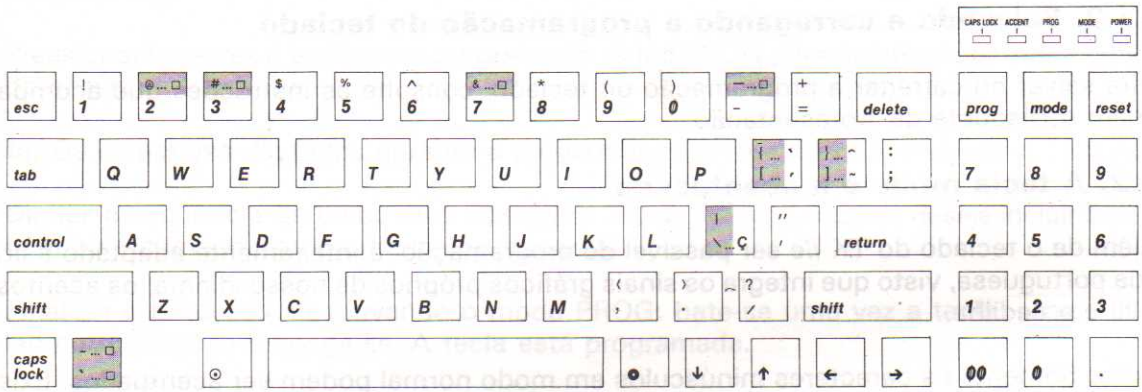
Nota: Para obterem-se caracteres maiúsculos acentuados, segue-se o mesmo procedimento descrito acima. Contudo, tais caracteres são representados no vídeo por seus correspondentes minúsculos, precedidos de um sinal > (por exemplo, um A com acento agudo, é representado por >á).

As teclas de acentuação do modo acentuado não produzem efeito algum na tela, se não precedem caracteres acentuáveis, ou seja, se são acionadas antes de caracteres que não representam vogais.

Nota: O cê-cedilha é obtido diretamente numa única tecla.

Os caracteres gerados pelo modo acentuado podem ser impressos via impressora, desde que o computador esteja utilizando a interface Microdigital, que traduz os sinais enviados pelo TK //e ao periférico de impressão.

Nota: Os caracteres @ # & _ ~ ` são alterados no modo acentuado, mas não perdem sua função, podendo ser obtidos na impressora, desde que sejam digitados precedidos do sinal <. Consulte o manual da Super Parallel Card para maiores detalhes.



6.3. A tecla reset

A tecla reset está localizada no extremo superior direito do teclado, acima do bloco numérico. Quando é acionada isoladamente, não exerce função alguma. Se é pressionada em conjunto com a tecla control, pode interromper a execução de um programa.

Nota: Via de regra, havendo um programa aplicativo na memória do computador, control-reset não destrói o conteúdo da memória, fazendo simplesmente com que o programa volte à tela inicial ou ao menu principal.

6.4. A tecla círculo vazio

Situada imediatamente à esquerda da barra de espaço, esta tecla somente funciona isoladamente quando um vídeo-jogo está na memória do computador. Neste caso, ela atua como o botão #0 do controlador manual.

6.4.1."Partida fria"

Mantendo-se presa a tecla ◉ (círculo vazio), enquanto se acionam as teclas control e reset, obtém-se o mesmo efeito da comutação da chave liga-desliga, ou seja:

a) Apaga-se todo o conteúdo da memória RAM.

b) Aciona-se a unidade de disco, operando-se um "boot".

c) Carrega-se na memória o conteúdo do disquete lido, caso este contenha um sistema operacional compatível com o TK //e, ou gera-se uma mensagem de erro, ou o sistema entra em linguagem de máquina, exibindo o prompt asterisco, seguido de caracteres da linguagem Assembly.

Este sistema é o mais indicado quando se deseja reiniciar um processamento qualquer. Substitui o "liga-desliga" com muita vantagem, não desgastando as partes eletroeletrônicas do computador.

Nota: Esta função não afeta a programação do teclado, apenas limpa o buffer.

6.5. A tecla círculo cheio

Como acontece com a tecla círculo vazio, o toque na tecla [●] (circulo cheio) não surte nenhum efeito. Quando se carrega um vídeo-jogo, ela equivale ao botão #1 do controlador manual.

6.6. A tecla esc e o modo ESCAPE

Devido à sua ampla e variada utilização, a tecla [esc] é tratada particularmente no item 4 do capítulo V deste manual.

7. As Teclas Pré-Programadas (control-mode)

Quase todas as teclas do TK //e são previamente programadas.

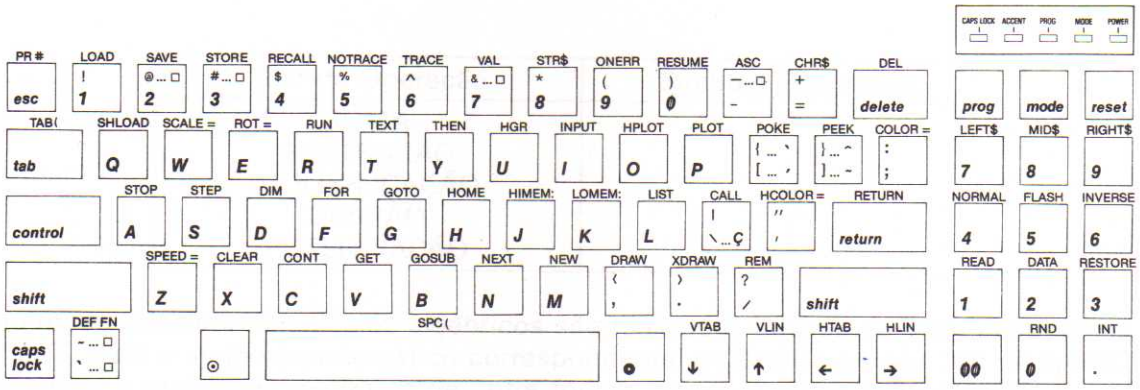
Nota: A pré-programação das teclas pode ser alterada pelo usuário, bastando que este as reprograme (veja o item 6.1).

Para que uma programação já definida seja utilizada, é necessário entrar no modo pré-programado. Para tanto, deve-se reter a tecla [control] e pressionar a tecla [mode]. A partir deste passo, é preciso acompanhar as instruções que seguem, para dar acesso à função desejada.

7.1. Funções acionadas por control + shift + tecla:

Estas funções são relativas ao BASIC.

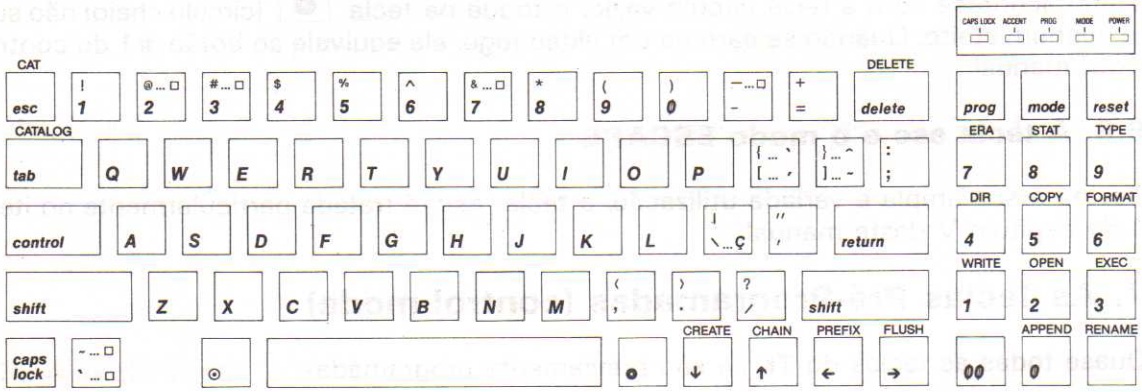
Mantendo-se pressionadas as teclas [control] e [shift], deve-se pressionar a tecla correspondente à função desejada, baseando-se na figura apresentada a seguir:



7.2. Funções acionadas por shift + tecla:

Estas funções estão relacionadas com sistemas operacionais para disquete.

As funções arroladas a seguir são obtidas mantendo-se apenas a tecla shift pressionada, enquanto se toca a tecla correspondente ao comando desejado:



7.3. Delete como apagador retroativo

O acionamento da tecla delete isoladamente, em modo pré-programado, faz o cursor mover-se uma posição à esquerda, eliminando-se o caractere que ali se encontre.

Para sair do modo pré-programado, devem-se usar as teclas control e prog conjuntamente, conforme descrevemos no subitem 6.1.1.

II - PRIMEIRAS OPERAÇÕES

1. O TK //e como Calculadora

Você pode utilizar seu computador como calculadora. Para tanto, não há nenhuma complicação: basta que se empregue o chamado modo imediato da linguagem BASIC.

Neste capítulo mostra-se como fazer para que o TK //e efetue operações algébricas através do modo imediato. Nos capítulos referentes à programação BASIC, encontram-se explicações sobre outras operações matemáticas mais complexas.

1.1. O modo imediato

Modo imediato é o modo do processamento que o computador realiza quando se emprega a linguagem BASIC diretamente, ou seja, sem escrever um programa. Neste modo, também denominado modo direto, o TK //e processa imediatamente as instruções e, de pronto, a tela mostra o resultado da tarefa executada pelo computador.

Faça um pequeno exercício. Digite a instrução PRINT e o número 583. Em seguida, pressione a tecla return.

Aparece no vídeo:

```
PRINT 583
583
]
```

2. Os Cálculos

A linguagem BASIC usa uma simbologia própria para expressar as operações algébricas.

Observe a tabela abaixo:

Tipo de Operação	Símbolo Empregado
SOMA	+
SUBTRAÇÃO	-
MULTIPLICAÇÃO	*
DIVISÃO	/
POTENCIAÇÃO	^

No modo imediato os cálculos algébricos são realizados através do mesmo comando, o PRINT, mais o símbolo matemático correspondente, no BASIC, a cada tipo de operação. Somente a radiciação realiza-se de outra forma, como veremos mais adiante.

2.1. Soma

Para somar, você deve digitar:

] **PRINT (1.ª parcela) + (2.ª parcela) + ... + (n parcela)**

Ao final, tecle return.

Exemplos:

a) Somar 30 e 50:

] **PRINT 30 + 50** **return**

A tela mostra:

```
] PRINT 30 + 50
80
]
```

b) Efetuar 100 + 25 + 4 + 1520

] **PRINT 100 + 25 + 4 + 1520** **return**

A tela apresenta:

```
] PRINT 100 + 25 + 4 + 1520
1649
]
```

2.2. Subtração

Para subtrair, você deve digitar:

] **PRINT (minuendo)-(1.º subtraendo)-...-(n subtraendo)**

Ao final, tecle return.

Exemplos:

a) Subtrair 30 de 200:

```
] PRINT 200-30 return
```

A tela exibe:

```
] PRINT 200-30
170
]
```

b) Efetuar 10-20-86:

```
] PRINT 10-20-86 return
```

A tela mostra:

```
] PRINT 10-20-86
-96
]
```

2.3. Multiplicação

Para multiplicar você deve digitar:

```
] PRINT (multiplicando)*(1.º multiplicador) *...* (n multiplicador)
```

Ao final, tecle return.

Exemplos:

a) Multiplicar 5 por 6:

```
] PRINT 5*6 return
```

A tela apresenta:

```
] PRINT 5*6
30
]
```

b) Efetuar 10 x 4 x 200:

```
] PRINT 10*4*200        return
```

A tela exibe:

```
] PRINT 10*4*200
8000
]
```

2.4 Divisão

Para dividir, você deve digitar:

```
] PRINT (dividendo)/(1.º divisor)/.../(n divisor)
```

Tecle return ao final.

Exemplos:

a) Dividir 505 por 5

```
] PRINT 505/5        return
```

A tela mostra:

```
] PRINT 505/5
101
]
```

b) Efetuar 240 ÷ 4 ÷ 2

```
] PRINT 240/4/2        return
```

A tela expõe:

```
] PRINT 240/4/2
30
]
```

2.5. Potenciação

Para elevar um número a uma potência, você deve digitar:

```
] PRINT (base)^(expoente)
```

Tecle return ao final.

Exemplo:

Elevar 5 à 5.ª potência:

```
] PRINT 5^5        return
```

A tela mostra:

```
] PRINT 5^5
3125
]
```

2.6. Radiciação

Para extrair a raiz de um número, você deve digitar:

$$\text{] PRINT (radicando) } \wedge \left(\frac{1}{\text{expoente}} \right)$$

Pressione return ao final.

Exemplo:

Calcular a raiz quarta de 8:

```
] PRINT 8^(1/4)        return
```

Note que o expoente (1/4) está entre parênteses. Os parênteses são necessários para informar ao computador que realize em primeiro lugar a divisão.

A tela mostra:

```
] PRINT 8^(1/4)
1.68179283
```

2.6.1. Raiz quadrada

Para extrair a raiz quadrada de um número, você pode digitar o comando PRINT, a função SQR (radicando) e return.

Exemplo:

Extrair a raiz quadrada de 64:

```
] PRINT SQR(64)        return
```

A tela exibe:

```
PRINT SQR(64)
8
]
```

2.7. Expressões algébricas

O computador realiza as operações algébricas segundo uma ordem de prioridade estabelecida:

1.º) potenciação
2.º) multiplicação/divisão
3.º) soma/subtração

Esta ordenação pode ser alterada com o emprego de parênteses nas operações a serem realizadas em primeiro lugar.

Exemplos:

a) Calcular 55 + 20 - 10 X 2 - 45 ÷ 3

```
] PRINT 55+20-10*2-45/3
  40
```

b) Efetuar 55 + (20-10) X 2 - 45 ÷ 3

```
] PRINT 55+(20-10)*2- 45 / 3
  60
```

c) Resolver [55 + (20 - 10)] X 2 - 45 ÷ 3

```
] PRINT (55+(20-10))*2-45/3
  115
```

d) Efetuar 55 + {20 - [10 X 2 - (45÷3)]}

```
] PRINT 55+(20-(10*2-(45/3)))
  70
```

O TK //e também é capaz de somar e subtrair números hexadecimais. Este assunto é tratado capítulo XIV.

3. Os Números no TK //e

3.1. Forma dos números no TK //e

A forma dos números no TK //e obedece a determinadas normas, como veremos a seguir:

3.1.1. O computador emprega "ponto", em vez de vírgula, nos números fracionários. Assim sendo, sempre se deve usar ponto para que o micro entenda tratar-se de fração decimal.

Ex.: Faremos 1,5 + Ø,8

```
] PRINT 1.5+Ø.8
  2.3
```

3.1.2. Todo zero à esquerda do número é eliminado. O mesmo acontece aos zeros da extremidade direita pospostos ao ponto decimal.

Ex.:

```
] PRINT ØØ57Ø.8ØØ
  57Ø.8
```

Se o número não tiver parte inteira, não e necessário digitar o zero à esquerda do ponto decimal.

Ex.: 2,3 + Ø,9

```
] PRINT 2.3+.9
  3.2
```

3.1.3. O TK //e emprega a notação científica para imprimir números que, sendo tão-somente fracionários (sem parte inteira), possuam mais de nove algarismos. Neste caso, os nove algarismos de precisão começam a ser contados a partir do primeiro algarismo à direita do ponto decimal, diferente de zero.

Ex.:

```
] PRINT .ØØ1234567894
  1.23456789E-Ø3
```

Exceção: Na faixa de Ø,Ø1ØØØØØØØ1 a Ø,Ø999999998 os números são impressos com dez dígitos, incluindo-se o zero à direita do ponto decimal.

Também emprega a notação científica para imprimir inteiros com mais de nove algarismos.

Ex.:

```
] PRINT 1ØØØØØØØØØ
  1E+Ø9
```

3.1.4. Se, entre as partes inteira e decimal de um número, incluíem-se mais de nove algarismos, o número é arredondado: para mais ou para menos, conforme o último algarismo seja, respectivamente, maior (ou igual) ou menor que 5.

Ex.:

```
] PRINT 1234567.766
1234567.77
] PRINT Ø.1234567894
.123456789
```

3.1.5. Se o número é muito próximo de zero, entre -3E-39 e 3E-39 (-3*1Ø^-39 e 3*1Ø^-39), é arredondado para zero.

Ex.:

```
] PRINT 2E-39 Ø
```

Desejando fugir das normas dos subitens 3.3.2 ao 3.3.5, empregue aspas.

Exemplos:

```
] PRINT "Ø57Ø.8ØØ"
Ø57Ø.8ØØ

] PRINT".ØØ1234567894"
Ø.Ø1234567894

] PRINT "1ØØØØØØØØØ"
1ØØØØØØØØØ

] PRINT "1234567.766"
1234567.766

] PRINT "Ø.1234567894"
Ø.1234567894

] PRINT "Ø.ØØØØØØØØØØØØØØØØØØØØØØØØØØØØØ2"
Ø.ØØØØØØØØØØØØØØØØØØØØØØØØØØØØØ2
```

Nota: Este expediente não se presta para cálculos. O computador somente registra como número o que é digitado entre aspas, se é empregada a função VAL (explicada no capítulo VII).

3.2. Noções de notação científica

A notação científica é um recurso empregado para resumir a grafia de números muito extensos. Nesta notação os números têm a seguinte forma:

± número E ± ee

Onde:

± = sinal positivo ou negativo do número;

E = *1Ø^ (multiplicado por 1Ø elevado ao expoente);

± = sinal positivo ou negativo do expoente;

ee = expoente.

Notação Usual	Notação Científica
1ØØØØØØØØØ	1E+9
-1ØØØØØØØØØ	-1E+9
Ø,ØØØØØØØØ1	1E-9
-Ø,ØØØØ123456	-1,23456E-5

O sinal + pode ser omitido tanto no número como no expoente.

3.3. Limites numéricos no TK //e

O TK //e faz operações com os números reais compreendidos na seguinte faixa: - 1E+38 a +1E+38.

Sendo efetuado um cálculo algébrico cujo resultado ultrapassa esta faixa, o TK //e emite uma mensagem de erro:

? ESTOURO ERRO

Para cálculos que empregam apenas as operações de adição e/ou subtração, a faixa estende-se, indo de: -1.7E38 a 1.7E38.

O TK //e não opera com números cujo número de dígitos seja superior a 38. Se você escreve um número com mais de 38 dígitos, recebe a mensagem: ESTOURO ERRO. Em tal hipótese, deve-se empregar a notação científica.

Limites Numéricos

Estouro	Erro				Estouro	Erro
-1.7E38	-1E38		Ø		1E38	1.7E38
soma			**3E-39**			**soma**
subtração			**-3E-39**			**subtração**

III - PROGRAMAÇÃO BASIC

Para que o computador execute qualquer tipo de operação, é necessário que lhe sejam fornecidas instruções detalhadas. Ao conjunto de instruções com as quais se informa o computador, para que realize tarefas, dá-se o nome de programa.

Os programas devem conter instruções escritas numa linguagem que o computador seja capaz de entender. Dentre as linguagens que o TK //e compreende, examinaremos o BASIC, que é de simples e rápida aprendizagem, e de considerável emprego na área de informática.

1. As Primeiras Instruções

Este item apenas apresenta algumas das instruções mais utilizadas na linguagem BASIC para desenvolver programas. Aqui, elas são descritas de maneira sucinta e empregadas no modo imediato. No item seguinte deste capítulo, você vai saber como utilizar essas instruções no modo programado.

1.1. NEW

Esta instrução faz com que todo conteúdo da memória RAM do TK //e (programa e dados) seja apagado. Terminando o processamento de algum programa, você deve empregá-la antes do início de outro, para garantir que não haja interferência das instruções anteriormente carregadas.

Deve-se digitar:

```
] NEW      return
```

1.2. PRINT

A instrução PRINT faz com que o programa apresente determinado conteúdo na tela. Ela pode ser empregada de quatro formas básicas diferentes:

1.2.1. PRINT "conteúdo"

Digitando-se entre aspas o conteúdo após PRINT, tal conteúdo aparece exatamente na forma em que foi digitado.

Digite os seguintes comandos e confira os resultados com os que apresentamos abaixo das linhas de comando de nossos exemplos:

```
] PRINT "COMPUTADOR TK //e"      return
COMPUTADOR TK //e
] PRINT "8 + 3"      return
8 + 3
```

No último exemplo, o TK //e não executou a operação algébrica indicada, simplesmente a apresentou na forma em que foi digitada.

Sempre que você quiser apresentar na tela um conjunto de caracteres alfabéticos (por exemplo, um nome) ou um conteúdo fixo (inalterável), digitado logo após um comando PRINT, deve fazê-lo entre aspas.

1.2.2. PRINT conteúdo

Esta forma é utilizada para a apresentação de um número preestabelecido ou para a obtenção do resultado de uma operação algébrica.

Experimente digitar:

```
] PRINT 888          return
888

] PRINT 8 + 8        return
16

] PRINT MENSAGEM     return
0
```

No último exemplo, em vez da impressão da palavra MENSAGEM, obtém-se o valor zero. Como vimos no subitem acima, grupos de caracteres alfabéticos devem ser digitados entre aspas. O valor zero aparece porque o computador considera a palavra MENSAGEM uma variável (falaremos sobre variáveis no próximo capítulo).

1.2.3. PRINT conteúdo, conteúdo, conteúdo, ...

Sendo digitada uma vírgula entre dois conteúdos, estes são exibidos numa tabulação padrão de quinze espaços.

Exemplo:

```
] PRINT 80, 64, 250          return
80              64              250

] PRINT "UM","DOIS","TRES"   return
UM              DOIS            TRES
```

1.2.4. PRINT conteúdo ; conteúdo

Sendo digitado ponto e vírgula entre dois conteúdos, estes são apresentados contiguamente, sem intervalo.

Exemplo:

```
] PRINT 50;84
5084
```

Exemplo da aplicação das quatro fórmulas básicas do comando PRINT:

```
] PRINT "8 +5 = ";8+5," 4 + 5 = ";4+5
8 + 5 = 13      4 + 5 = 9
```

Nota: Em vez de digitar a palavra PRINT, pode-se digitar simplesmente o ponto de interrogação [?].

Exemplos:

```
]?"SOU O TK //e"
SOU O TK //e

]?"6 + 6 = ";6+6
6 + 6 = 12
```

1.3. HOME

A instrução HOME faz com que todo o conteúdo da tela seja suprimido e que o cursor vá para a primeira posição da tela.

Digite:

]HOME [return]

Observe o resultado.

1.4. END

A instrução END faz com que a execução dum programa seja finalizada.

1.5. LIST

Esta instrução faz com que todas as linhas do programa sejam apresentadas, ou seja, listadas na tela.

Pode-se listar apenas um trecho específico do programa, usando:

LIST X,Y ou LIST X-Y ou LIST -Y ou LIST X- ou LIST, Y ou LIST X,

Onde: X deve ser o número da primeira linha e Y, o número da última linha a ser listada.

Necessitanto parar uma listagem muito longa, mantenha pressionada a tecla [control] e digite S.

Para dar prosseguimento à listagem interrompida, pressione qualquer tecla.

Desejando interromper definitivamente a listagem, tecle [control] e C, conjuntamente.

1.6. RUN

Esta é a instrução que "dá partida" à execução de um programa ou, como se costuma dizer, faz com que o programa "rode".

Para analisar o programa a partir de determinada linha, pode-se acionar o comando RUN XYZ. Onde: XYZ representa uma linha genérica de programa a partir da qual a execução é desejada.

2. O Modo Programado

Você já aprendeu a usar o TK //e através do modo imediato que, apesar de útil para algumas tarefas, impõe muitos limites à capacidade real do computador. Neste item você começa a ter contato com a programação. No modo programado em vez de solicitar tarefa por tarefa ao TK //e, você vai instruí-lo para que execute operações seqüencialmente. Seu computador irá combiná-las, compará-las, ordená-las etc., empregando todo o seu potencial.

2.1. A enumeração das linhas

No modo programado, devem-se enumerar as linhas do programa. Não há números preestabelecidos para tal, mas há algumas regras a observar:

a) Relacione cada linha a um número inteiro.

b) Enumere normalmente em ordem crescente. As linhas de comando sempre são executadas, em seqüência cronológica, do menor número para o maior.

c) Dê preferência à enumeração de dez em dez (10, 20, 30, 40). É muito freqüente a necessidade de incluir novas linhas de comando entre as já estabelecidas. Com essa seqüência de números não consecutivos, podem-se incluir até nove linhas entre cada uma das linhas originalmente ordenadas.

Exemplo:

Digite:

```
]10 PRINT "O TK //e CALCULA"        return
]20 PRINT "60 + 25 = ";60+25        return
```

Em seguida digite a instrução RUN (sem enumerar a linha) e pressione a tecla return.

A tela exibe:

```
O TK //e CALCULA
60 + 25 = 85 !
```

Podem-se incluir mais nove linhas entre cada uma das linhas de comando 10 e 20. Experimente fazer as seguintes inclusões:

```
]3 PRINT "ATENCAO"          return
]18 PRINT "EM MODO PROGRAMADO"          return
```

Agora digite RUN e tecle return:

```
ATENCAO
O TK //e CALCULA
EM MODO
PROGRAMADO
60+25 = 85
```

O TK //e permite utilizar até 239 caracteres (incluindo-se os espaços em branco) numa mesma linha de comando. Por isso, não se incomode se a instrução não couber numa só linha da tela. O computador vai entender que se trata da mesma linha de comando, somente tendo-a por terminada, quando você teclar return.

Observação: Sendo atribuído o mesmo número de linha a linhas diferentes, prevalece a última delas.

2.2. O uso de ponto e vírgula após o comando PRINT

O ponto e vírgula após o comando PRINT faz com que o conteúdo do próximo PRINT seja apresentado na mesma linha.

Digite os exemplos a seguir e note a diferença:

1.º = exemplo:

```
]NEW
]10 PRINT "PRIMEIRO"
]20 PRINT "SEGUNDO"
]30 END
]RUN
```

Resultado:

```
PRIMEIRO
SEGUNDO
]
```

2.º = exemplo:

```
]NEW
]10 PRINT "TERCEIRO";
]20 PRINT "QUARTO"
]30 END
]RUN
```

Resultado:

```
TERCEIROQUARTO
]
```

2.3. O primeiro programa

O objetivo deste subitem é mostrar-lhe que, com algumas instruções básicas, é possível montar um programa simples.

O programa proposto visa à construção da tabela de multiplicação de 1 a 10 dos números 2 e 3.

Todas as operações que você deve executar no TK //e são descritas abaixo, passo a passo (lembre-se de que, ao final de cada linha, deve pressionar a tecla return, para que as informações sejam registradas pelo computador):

a) Limpe a memória do TK //e, digitando:

```
]NEW
```

b) Prepare o título da tabela:

```
]10 PRINT "TABELA DE MULTIPLICACAO"
```

c) Forneça uma linha em branco entre o título e a tabela:

```
]20 PRINT
```

d) Apresente na primeira linha os multiplicandos:

```
]30 PRINT ,"X 2", "X 3"
```

e) Prepare agora a tabela:

```
]40 PRINT 1,1 * 2,1 * 3
]50 PRINT 2,2 * 2,2 * 3
]60 PRINT 3,3 * 2,3 * 3
]70 PRINT 4,4 * 2,4 * 3
]80 PRINT 5,5 * 2,5 * 3
]90 PRINT 6,6 * 2,6 * 3
]100 PRINT 7,7 * 2,7 * 3
]110 PRINT 8,8 * 2,8 * 3
]120 PRINT 9,9 * 2,9 * 3
]130 PRINT 10,10 * 2,10 * 3
```

f) Finalize o programa:

```
]140 END
```

g) Examine a listagem do programa:

```
]LIST
```

Não havendo erro de digitação nem esquecimento de alguma instrução, a tela deve apresentar:

```
10 PRINT "TABELA DE MULTIPLICACAO"
0 PRINT
30 PRINT, "X 2" , "X 3"
40 PRINT 1,1 * 2,1 * 3
50 PRINT 2,2 * 2,2 * 3
60 PRINT 3,3 * 2,3 * 3
70 PRINT 4,4 * 2,4 * 3
80 PRINT 5,5 * 2,5 * 3
90 PRINT 6,6 * 2,6 * 3
100 PRINT 7,7 * 2,7 * 3
110 PRINT 8,8 * 2,8 * 3
120 PRINT 9,9 * 2,9 * 3
130 PRINT 10,10 * 2,10 * 3
140 END
```

h) Inicie a execução do programa:

```
]RUN
```

Havendo algum erro, surge uma mensagem, indicando a linha da ocorrência:

```
? SINTAX ERRO EM ...
```

Nesta hipótese, digite novamente a linha corrigida, utilizando o mesmo número da linha original errada (lembre que, no caso de linhas com numeração idêntica, prevalece a que foi digitada por último).

Se tudo está correto, obtém-se na tela a seguinte tabela:

TABELA DE MULTIPLICACAO

	X 2	X 3
1	2	3
2	4	6
3	6	9
4	8	12
5	10	15
6	12	18
7	14	21
8	16	24
9	18	27
10	20	30

Está pronto e testado seu primeiro programa.

Suponha que, estando todas as linhas corretas, você lembre que o programa inicialmente deveria limpar a tela, e, somente então, executar a tabela, apresentando-a a partir da primeira linha/coluna do vídeo. Neste caso, inclua, em qualquer linha do programa de 1 a 9 (inclusive), uma instrução que apague os caracteres e posicione o cursor na primeira coluna da primeira linha:

```
]5 HOME
```

Suponha, ainda, que você queira que haja uma linha divisória entre os multiplicandos e multiplicadores. Para tanto, emita a ordem:

```
]35 PRINT" _ _ _ _ _ _ _ _ _"
```

Agora acione novamente o comando LIST, verifique a listagem do programa e faça as correções dos eventuais erros.

3. O Uso de Dois Pontos numa Linha de Programa

Empregando-se dois pontos (:), pode-se incluir mais de um comando numa única linha de programa. Por exemplo, no caso do programa apresentado no subitem acima, pode-se reunir as linhas 5 e 10, digitando:

```
]10 HOME : PRINT "TABELA DE MULTIPLICACAO"
```

Este artifício também pode ser utilizado no modo imediato.

IV - OUTROS COMANDOS

Neste capítulo você aprenderá novos comandos que lhe permitirão desenvolver programas um pouco mais sofisticados. Antes, porém, serão apresentados alguns conceitos muito úteis durante a programação: os dados e as variáveis.

1. Alguns Conceitos Importantes

1.1. Os dados

A principal função do computador é a de receber determinados dados, manipulá-los ou processá-los e, então, apresentar novos dados.

Na linguagem BASIC do TK //e existem dois principais tipos de dados: cadeias e números.

1.1.1. Cadeia (string)

Uma cadeia é qualquer seqüência de caracteres incluída entre aspas. Já utilizamos cadeias com o comando PRINT, para apresentar mensagens na tela.

As cadeias podem ser formadas por uma seqüência de Ø a 239 caracteres quaisquer, sem contarem-se as aspas.

1.1.2. Números

Você já recebeu explicações sobre os números no item 3 do capítulo V.

1.2. Variáveis

Os dados podem ser informados ao computador sob a forma de variáveis. Variável é uma etiqueta ou nome que se associa a determinadas constantes. Por exemplo, tomemos a letra X como sendo uma variável. A essa variável vamos associar o número 1: X = 1. Se lhe perguntássemos qual é o total da soma X + 2, você deveria responder 3, pois: sendo X = 1 → X + 2 = 1 + 2 = 3.

Agora vamos associar a variável X ao número 5: X = 5. Quanto vale agora X + 2? Resp.: X = 5 → X + 2 = 5 + 2 = 7

Atenção: Uma variável somente pode ser associada a uma constante por vez.

1.3. O nome que uma variável pode ter no BASIC do TK //e

Existem três tipos de variáveis, caracterizadas pela forma dos dados a elas associados: a variável tipo cadeia, correlacionada a qualquer conjunto de caracteres entre aspas; a variável inteira, associada somente a número sem parte fracionária; e a variável real, correspondente a qualquer número real, seja este inteiro, fracionário ou com parte inteira e fracionária.

O nome da variável pode ser de qualquer tamanho, mas o BASIC do TK //e somente reconhece os dois primeiros caracteres. Para determinar o nome, observe as seguintes regras:

a) O primeiro caractere do nome da variável obrigatoriamente deve ser uma letra.

b) Não é necessário mais de um caractere para definir o nome de uma variável. Pode-se, contudo, utilizar quaisquer outros caracteres (letra, número, símbolo etc.), além do primeiro.

c) Ao final do nome da variável, deve ser empregado um símbolo para definir o tipo:

$ - para variável do tipo cadeia
% - para variável do tipo inteira
Para variável do tipo real, não se emprega nenhum símbolo.

No próximo item, encontram-se exemplos que ilustram as regras acima.

2. LET

Neste item explicamos como relacionar um nome de variável a um valor constante, através do comando LET. Digite os exemplos abaixo e verifique os resultados após teclar return:

```
]LET BE$ = "Besouro"
]PRINT BE$
Besouro
```

Atribuímos à variável BE$ a cadeia Besouro. Quando pedimos ao computador que apresentasse a variável BE$, ele apresentou a cadeia Besouro.

```
]LET BE$ = "Agua"
]PRINT BE$
Agua
```

Agora fizemos com que a variável BE$ passasse a se relacionar com AGUA. Quando ordenamos que o TK //e apresentasse a variável BE$, surgiu a cadeia AGUA.

Note que o sinal $, no final do nome da variável, é necessário por se tratar de uma cadeia. E observe que uma variável somente pode ser associada a um valor por vez, sendo desprezado o antigo valor sempre que se atribui um novo.

Dissemos, no subitem 1.3, que o computador somente registra os dois primeiros caracteres do nome-de-variável. Note o exemplo abaixo:

```
]LET Casa = 5
]PRINT Ca
5
```

Você deve tomar cuidado ao programar:

```
]10 LET Casa = 5
]20 LET Caixa = 3
]30 PRINT 4 + Casa
]RUN
7
```

De acordo com a regra vista no subitem 1.3.2, podemos utilizar apenas uma letra para definir o nome de uma variável:

```
]LET C$ = "MICROCOMPUTADOR"
]PRINT C$
MICROCOMPUTADOR
```

Na realidade, o comando LET é opcional, de forma que podemos omiti-lo para executar a associação entre a variável e o seu valor. Assim, por exemplo, é indiferente digitar:

```
]LET A$ = "COMPUTADOR"   ou   ]A$ = "COMPUTADOR"
```

Para poupar tempo durante a digitação, e para economizar memória, recomendamos-lhe não empregar LET.

Veja o que acontece quando as regras sobre variáveis não são obedecidas:

a)
```
]LET 5F$ = "Verde"
? SINTAX ERRO
```

O primeiro caractere do nome é um número.

```
]5F$ = "Verde"
]PRINT 5F$
]5
```

Não se empregando LET, o TK //e entende o 5 como número de linha de programa.

b)
```
]LET X$ = 7
?INCOMPATIVEL ERRO
```

Símbolo de variável tipo cadeia em associação sem aspas.

c)
```
]LET X% = 0.2
]PRINT X%
0

]LET X% = 1.2
]PRINT X%
1
```

Símbolo de variável inteira associado a números fracionários. A parte fracionária é desprezada.

```
]LET X = A
]PRINT X
0
```

Variável tipo real associada a um caractere. O computador interpretou-o como uma variável.

3. GOTO

Este comando faz com que o programa seja desviado para uma determinada linha. Deve ser digitado da seguinte forma:

GOTO número da linha

Vamos ilustrar o emprego deste comando:

Suponha que você queira fazer um programa que apresente o nome MICRODIGITAL no início de todas as linhas da tela.

Para tanto, digite:

```
]NEW
]10 HOME
]20 PRINT "MICRODIGITAL"
]30 GOTO 20
]RUN
```

Note o resultado na tela.

A figura abaixo esquematiza o fluxo do programa:

RUN (dá início à execução do programa)
HOME (limpa a tela e posiciona o cursor em seu início)
PRINT "MICRODIGITAL" (apresenta MICRODIGITAL na tela)
GOTO 20 (desvia o programa para a linha 20)

Quando quiser interromper o programa, digite o conjunto de teclas control C.

Vamos agora modificar um pouco o programa:

```
NEW
]10 HOME
]20 PRINT "MICRODIGITAL"
]25 PRINT
]30 GOTO 10
]RUN
```

O nome MICRODIGITAL fica piscando na primeira linha da tela.

Enquanto não se interrompe o programa (através de um control C), ele continua a rodar indefinidamente, pois sempre que alcança a instrução GOTO, é desviado para a instrução anterior a ela. Neste caso, não adianta incluir uma instrução END após GOTO: o programa nunca a atingirá.

4. INPUT

Este comando, assim como LET, permite uma associação entre um nome-de-variável e seu valor.

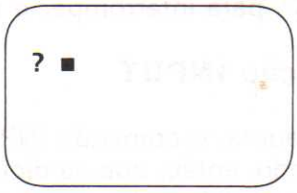
O formato de INPUT é:

INPUT nome-de-variável, (nome-de-variável, ...)

Digite:

```
]NEW
]10 HOME
]20 INPUT A$
]30 PRINT A$
]40 GOTO 30
]RUN
```

No canto esquerdo da tela, surge um ponto de interrogação e o cursor, como no desenho:



A instrução "20 INPUT A$" executa as seguintes operações:

a) apresenta um ponto de interrogação na tela;

b) posiciona o cursor;

c) espera pela digitação de alguma cadeia. Quando a cadeia é digitada e a tecla return pressionada, INPUT faz com que a cadeia seja associada à variável A$ e, somente então, permite que o programa prossiga para a próxima linha de comando.

Dando continuidade ao programa, digite seu nome e pressione a tecla return. Observe o resultado na tela (para interromper o programa, pressione control C).

Havendo diversas variáveis num INPUT, deve-se digitar o atributo de cada variável por vez, ou, então, digitá-los separados por vírgula.

Vamos agora a mais um exemplo:

```
]NEW
]10 HOME
]20 INPUT N$
]30 PRINT N$
]40 GOTO 20
]RUN
```

A figura a seguir esquematiza o fluxograma do exemplo:

RUN (dá início ao programa)
INPUT N$ (apresenta na tela o ponto de interrogação e o cursor, e aguarda a digitação de uma cadeia, para associar ao nome-de-variável N$).
PRINT N$ (apresenta na tela o valor associado a N$)
GOTO 20 (desvia o programa para a linha da instrução INPUT).

Após o ponto de interrogração, digite o nome SONIA:

```
?SONIA
```

Agora pressione a tecla [return]. A tela passa a apresentar:

```
?SONIA
SONIA
? ■
```

o computador apresenta as cadeias digitadas após o ponto de interrogação, até que se pressione o conjunto [control] [C], para interromper o programa.

4.1. Incrementando a instrução INPUT

De acordo com o que vimos até agora, o comando INPUT apresenta na tela o ponto de interrogação e o cursor, aguardando, então, que se digite uma cadeia e a tecla return. É interessante que seja apresentada também uma mensagem ou um incremento, indicando qual cadeia é desejada.

Modifique o programa anterior, digitando:

```
]20 INPUT "Qual o seu nome?"; N$
]40 GOTO 30
```

Vamos analisar o que a instrução da linha 20 executa:

a) apresenta na tela a frase "Qual o seu nome?";
b) aguarda a digitação da cadeia e da tecla [return];
c) associa a cadeia digitada ao nome-de-variável N$.

A forma do comando INPUT incrementado é a seguinte:

INPUT " texto " ; variável ; variável ; ...

Quando utilizamos o incremento, temos de acrescentar o sinal de interrogação dentro deste, pois o TK //e, neste caso, não imprime a interrogação automaticamente.

Sendo digitado o comando RUN e a tecla [return], surge na tela:

```
QUAL O SEU NOME ? ■
```

Digitando agora o seu nome (suponhamos, PAULO), o início de todas as linhas da tela é preenchido com ele:

```
PAULO
PAULO
PAULO
PAULO
```

Para interromper a execução, use [control] [C].

4.2. Uso de INPUT com variáveis numéricas

Também podemos usar a instrução INPUT para relacionar variáveis numéricas, como se vê no exemplo a seguir:

```
]NEW
]10 HOME
]20 INPUT A
]30 PRINT A
]40 PRINT
]50 GOTO 20
```

Vamos examinar o fluxo deste programa:

HOME (limpa a tela)
INPUT (apresenta o ponto de interrogação e o cursor, aguardando a digitação de um número real a ser relacionado ao nome-de-variável A)
PRINT A (apresenta o valor real associado à variável A)
PRINT (deixa uma linha em branco)
GOTO 20 (desvia o programa para a instrução INPUT A)

Após o comando RUN, o ponto de interrogação e o cursor surgem na tela. Digite o número 601 e pressione [return]. A linha seguinte apresenta este número. Uma linha depois, o ponto de interrogação e o cursor surgem novamente na tela. Você pode, em seguida, digitar outros números como: -84, 6.02 etc.

A tela mostra:

```
?601
601
?84
-84
?6.82
6.02
?"2"
?REDIGITE
?32
32
?
```

5. REM

Esta instrução permite incluir lembretes ou comentários em linhas do programa. O computador ignora estas linhas quando o programa é executado.

A instrução REM deve ser digitada da seguinte forma:

```
REM lembrete
```

Exemplo:

```
]NEW
]10 REM QUADRADO DE UM NUMERO
]20 INPUT "Qual e o numero?"; A
]30 PRINT
]40 PRINT A^2
]50 PRINT
]60 GOTO 20
]RUN
```

Durante a execução do programa, surge apenas a pergunta "Qual e o numero?", e, sendo digitado o número e a tecla return, aparece o valor do número ao quadrado. Todavia, sempre que se lista o programa (através do comando LIST), a linha 10 indica que a finalidade dele é elevar um número ao quadrado.

Os programadores experientes utilizam freqüentemente este recurso.

6. Instrução FOR ... NEXT

6.1. FOR ... NEXT

Esta instrução faz com que sejam repetidas, um certo número de vezes, as instruções que se encontram entre os comandos FOR e NEXT.

Por exemplo, digite:

```
]NEW
]10 HOME
]20 FOR I=1 TO 5
]30 PRINT "TK //e"
]40 NEXT I
]RUN
```

O nome " TK //e" é apresentado cinco vezes na tela.

Vamos analisar o formato geral desta instrução:

```
FOR variável= valor inicial da variável TO valor final
.
.
.
linhas de instrução a serem repetidas
.
.
.
NEXT variável
```

Saiba como funciona o programa que você acabou de rodar:

a) O computador coloca o valor inicial na variável.

b) O computador verifica se o valor inicial da variável "I" é menor que o valor final (5).

Se menor, faz com que o programa prossiga normalmente até encontrar o comando NEXT I (linha 40). Neste momento, o valor inicial da variável é acrescido de uma unidade.

c) O novo valor de I (agora I+1) é novamente comparado com o valor final da variável e, se ainda for menor, o programa salta para a linha seguinte à de FOR.

d) O ciclo se repete até que o valor final de I iguale-se a 5. Ocorre então o último ciclo. Na passagem por NEXT, I vale 6 e o computador, ao comparar 6 com o valor final determinado para I, faz com que o "loop" seja abandonado, prosseguindo o programa na primeira instrução após o NEXT.

Nota: Pode-se inserir uma instrução FOR ... NEXT numa única linha, tal como:

```
20 FOR I=0 TO 100: PRINT I : NEXT I
```

6.2. Uso de FOR ... NEXT como artifício matemático

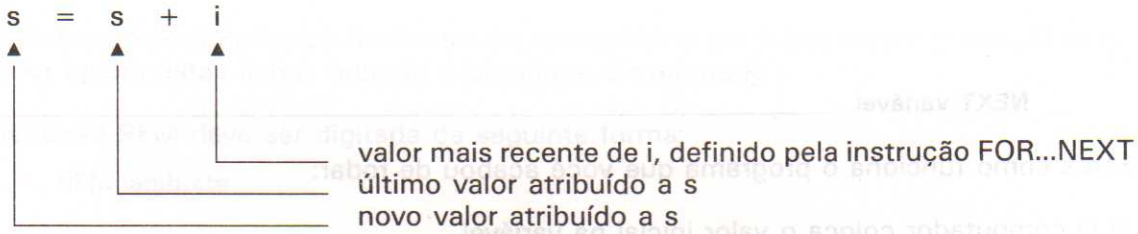
O potencial desta instrução é muito grande. Neste item explicamos a aplicação de FOR ... NEXT como artifício matemático dentro de um programa.

O programa-exemplo abaixo tem por função computar a soma dos inteiros positivos entre 1 e N.

Digite:

```
]NEW
]5 REM Calculo da soma de inteiros positivos entre 1 e n
]10 HOME
]20 INPUT "n= "; n
]30 LET s=0
]40 FOR I= 1 TO n
]50 LET s = s + i
]60 NEXT I
]70 PRINT "A soma e      ";s
]80 GOTO 20
```

A variável "s" (que, no caso, indica soma) é usada nas linhas 30, 50 e 70. Na linha 30, que é processada antes do comando FOR, o valor zero é atribuído a ela. A linha 50, entre os comandos FOR e NEXT, é executada para i=1, i=2, i=3 e assim por diante, até i=n. O novo valor de "s" passa a ser:



Por exemplo, suponha que se atribua o valor 3 à variável "n"; então, "s" assume os valores:

linha 30	S = 0
1.ª execução da linha 50	s = s+i = 0+1 = 1
2.ª execução da linha 50	s = s+i = 1+2 = 3
última execução da linha 50	s = s+i = 3+3 = 6

Execute o programa diversas vezes, variando sempre o valor de n.

6.3. STEP

Esta instrução faz com que se altere o "passo" da instrução FOR ... NEXT de 1 para o número desejado.

Recorde que cada vez que o programa passava em NEXT, o valor da variável indicada na ordem FOR era acrescido de uma unidade, até que se igualasse ao valor final ou superasse este. Pois bem, se junto a FOR houver a ordem STEP, a cada instrução NEXT, a variável passa a ser incrementada com o valor indicado após a ordem STEP. Este valor pode ser inteiro ou real, positivo ou negativo.

Observe o exemplo:

```
]NEW
]10 HOME
]20 FOR j=0 TO 10 STEP 2
]30 PRINT j
]40 NEXT j
]RUN
```

Quando o incremento é negativo, o programa continua o ciclo enquanto a variável determinada pelo FOR for maior que ou igual ao limite. Veja o programa a seguir:

```
]NEW
]10 HOME
]20 FOR j = 10 TO 0 STEP -0.5
]30 PRINT j
]40 NEXT j
```

Deve-se ter cuidado ao fazer dois ou mais "loops" FOR...NEXT ao mesmo tempo, ou seja, "loops aninhados". O programa abaixo mostra isto:

```
]NEW
]10 FOR m=0 TO 10
]20 FOR n=0 TO m
]30 PRINT m;":";n;":";
]40 NEXT n
]50 PRINT
]60 NEXT m
```

O "loop" n está inteiramente dentro do "loop" m, portanto, estão enquadrados de maneira correta. É imperativo que os "loops" não se cruzem, pois se isto acontece, o programa pode não funcionar normalmente.

Observe os seguintes exemplos:

a) Os loops não se cruzam:

```
10 FOR i=1 TO 10
20 FOR j=1 TO 10
30 PRINT i; "x";j;"="; i*j
40 NEXT j
50 NEXT i
```

b) Os loops se cruzam:

```
10 FOR i=1 TO 10
20 FOR j=1 TO 10
30 PRINT i; "x";j; "="; i*j
40 NEXT i
50 NEXT j
```

Nota: O TK //e envia a mensagem de erro "NEXT SEM FOR" sempre que NEXT é introduzido e FOR não.

6.4. Uso de FOR ... NEXT como tempo de espera

Execute os dois exemplos a seguir (para interrompê-los, digite control C):

```
A - ]NEW
    ]10 HOME
    ]20 PRINT "TK //e"
    ]30 GOTO 10
     ]RUN
```

```
B - ]NEW
    ]10 HOME
    ]20 FOR i = 1 TO 500
    ]30 NEXT i
    ]40 PRINT "TK //e"
    ]50 FOR i = 1 TO 500
    ]60 NEXT i
    ]70 GOTO 10
    ]RUN
```

Obs.: Após digitar RUN, aguarde um instante para que surja o nome TK //e na tela.

No exemplo A, quase não se consegue ler o nome TK //e, que pisca na tela. Já no exemplo B, os intervalos de tempo entre a apresentação da tela em branco e do nome TK //e são bem maiores.

Na linha 20, a ordem FOR i = 1 TO 500, seguida da instrução NEXT, faz com que o computador "conte", internamente, de 1 a 500, para somente então prosseguir o programa. O mesmo acontece na linha 50. O tempo de contagem é muito rápido (menos de um segundo), mas é perceptível.

O diagrama a seguir representa o fluxo do programa:

10 HOME (apaga a tela e posiciona o cursor em seu início)
20 FOR i = 1 TO 500 (atribui inicialmente o valor 1 a i e compara-o com 500. Esta comparação é repetida para os novos valores de i até que i = 500. Depois, a próxima ordem NEXT é ignorada)
30 NEXT i (acrescenta 1 à variável e desvia o programa para a ordem FOR anterior - linha 20)
40 PRINT TK //e (apresenta o nome TK //e)
50 FOR i = 1 TO 500 (similar à ordem da linha 20)
60 NEXT i (similar à ordem 30, mas agora a ordem FOR anterior encontra-se na linha 50)
70 GOTO 10 (desvia o programa para a linha 10)

Se você quiser que o nome TK //e pisque lentamente, basta aumentar as contagens efetuadas pelo programa, substituindo as linhas 20 e 50. Exemplo:

```
]20 FOR i = 1 TO 1000
]50 FOR i = 1 TO 1000
```

De modo inverso, para que TK //e pisque mais rapidamente, diminua as contagens, alterando as linhas 20 e 50.

Exemplo:

```
]20 FOR i = 1 TO 100
]50 FOR i = 1 TO 100
```

Finalizando, execute o programa abaixo e observe o resultado:

```
]NEW
]10 REM Educando o TK //e
]20 HOME
]30 PRINT ''Meu nome e TK //e''
]40 FOR b = 1 TO 1500 : NEXT b
]50 REM Obtendo o seu nome
]60 HOME
]70 INPUT ''Qual e o seu nome ?'';n$ : PRINT
]80 PRINT ''Prazer em conhece-lo ''; n$
]90 FOR c = 1 TO 1500 : NEXT c
]100 GOTO 20
] RUN
```

Desejando parar a execução, use control e return.

V - COMANDOS DE EDIÇÃO

1. Corrigindo e Alterando a Digitação dos Programas

Você já tomou conhecimento de formas através das quais pode corrigir erros em linhas de programa antes de digitar a tecla [return]. Vamos reuni-las e revê-las rapidamente:

a) A tecla [←] movimenta o cursor para a esquerda, de forma que os caracteres que ficam à direita dele são ignorados, ou seja, embora possam permanecer na tela, não são registrados na memória do TK //e depois que se pressiona a tecla [return].

b) A tecla [→] movimenta o cursor para a direita, tornando os caracteres de uma linha válidos, à medida que o cursor passa por eles.

c) o conjunto de teclas [control] [X] faz com que a linha onde se encontra o cursor seja ignorada.

d) O comando HOME apaga toda a tela e leva o cursor à primeira posição da primeira linha.

Você encontra a seguir outras técnicas de edição especialmente úteis no modo programado.

2. A Supressão de Linhas do Programa

2.1. Supressão de uma linha

Para suprimir uma linha inteira do programa, basta digitar o número desta linha e, em seguida, pressionar a tecla [return].

Exemplo:

```
] 10 REM PROGRAMA EXEMPLO
] 20 PRINT "Este e o computador TK //e"
] 30 PRINT "MICRODIGITAL ELETRONICA"
] 60 REM Ultima linha a ser suprimida
] 70 END
```

Para suprimir a linha 30, digite:

```
] 30 [return]
```

Use o comando LIST e observe que a linha 30 sumiu.

2.2. Supressão de um conjunto de linhas (DEL)

Para suprimir um conjunto de linhas, usa-se o comando DEL.

Exemplo:

```
] DEL 20,60
] LIST
10 REM Programa-exemplo
70 END
]
```

O comando DEL 20,60 elimina todas as linhas de programa entre as linhas 20 e 60 (inclusive). Ainda que a linha 20 não existisse, todas as linhas com numeração entre 20 e 60 seriam suprimidas.

Nota: Não confundir o comando DEL com a tecla delete.

3. A Permuta e a Supressão de Caracteres

As operações de permuta e de supressão dos caracteres da linha do programa são possíveis apenas enquanto a tecla [return] não é pressionada. Sempre que se aciona [return] numa linha, não se pode mais alterar isoladamente os caracteres nela registrados. É necessário reescrever a linha toda, a não ser que se entre no modo ESCAPE, descrito no item 4 deste capítulo.

3.1. Permuta de caracteres

A permuta de caracteres de uma linha é bastante simples: basta movimentar o cursor, através das teclas adequadas ([→] e [←]), para a parte que se deseja alterar e, então, digitar os novos caracteres sobre os antigos.

Atenção: Antes de teclar [return], leve o cursor até o final da linha, com o auxílio da tecla [→]. Do contrário, você perde tudo o que está entre o cursor e o final da linha!

3.2. Supressão de caracteres

Podem-se suprimir caracteres isoladamente, movendo o cursor e digitando um espaço em branco sobre eles. Note que no BASIC os espaços em branco a mais são ignorados, a não ser que sejam inseridos entre aspas.

4. Utilizando o Modo ESCAPE (esc)

O modo ESCAPE permite a movimentação livre do cursor para qualquer ponto da tela, sem alterar-lhe o conteúdo. Para tanto, basta que se pressione uma vez a tecla [esc] e utilizem-se as setas [↑], [↓], [←] e [→] ou as teclas [I], [M], [J], [K], respectivamente, para posicionar o cursor onde se deseja. As letras D, C, B e A também movimentam o cursor, mas sempre é necessário pressionar [esc] antes de usá-las.

4.1. Reeditando uma linha de programa

Digite este pequeno programa contendo erros intencionais a serem corrigidos posteriormente:

```
] NEW
] 10 PRUNT "O SABIA"
] 20 PRINT "CANTA"
] 30 PRINT "NAMATA"
```

Os erros de digitação aparecem na tela sem serem alterados.

Liste o programa. Ele deve aparecer assim:

```
10 P RUN T"O SABIA"
20 PRINT "CANTA"
30 PRINT "NAMATA"
```

Observe a listagem da linha 10. O "PRUNT" que você digitou propositadamente errado, transformou-se em "P RUN T", pois sendo RUN uma palavra reservada do BASIC, o TK //e encarrega-se de colocar espaços antes e depois dela.

Tente agora executar o programa. Digite RUN e pressione return.

Como já era previsto, o TK 3000 //e constatou o erro e emitiu a mensagem:

```
?SINTAX ERRO EM 10
]
```

Modifique a linha 10, com os seguintes passos:

a) Liste-a para facilitar o acesso do cursor a ela:

```
]LIST 10
```

Você obtém:

```
10 P RUN T"O SABIA"
```

b) Pressione esc uma vez, para entrar no modo ESCAPE.

c) Utilizando as setas direcionais, posicione o cursor no extremo esquerdo da linha (sobre o número 1 de 10).

d) Saia do modo ESCAPE, pressionando a tecla esc novamente ou a barra de espaço.

e) Utilizando a seta →, movimente o cursor até posicioná-lo sobre o U e digite sobre ele a letra I.

f) Prossiga com a seta → até o final da linha e então tecle return.

Para confirmar que a correção foi efetuada, liste o programa novamente. Desta vez você vê:

```
10 PRINT "O SABIA"
20 PRINT "CANTA"
30 PRINT "NAMATA"
```

Embora ainda existam erros, tente rodar o programa:

```
] RUN
```

Desta feita, o resultado é:

```
?SINTAX ERRO EM 20
```

Apesar de ter iniciado a execução do programa, o TK //e deteve-se na linha 20, pois outro erro foi encontrado: as aspas utilizadas (na verdade dois apóstrofos de cada lado da palavra CANTA).

Liste a linha 20 e proceda de maneira similar à utilizada para a correção da linha 10. Faça o mesmo na linha 30, separando "NAMATA". Após as mudanças, execute o programa, que deve gerar:

```
O SABIA
CANTA
NA MATA
```

4.2. Regras para utilização do modo ESCAPE

As seguintes regras devem ser aplicadas sempre que o modo ESCAPE é usado:

- Para entrar no modo ESCAPE, pressione [esc]; para sair dele, pressione [esc] novamente ou a barra de espaço.
- Para corrigir ou modificar algo, utilizando o modo ESCAPE, você deve posicionar o cursor sobre o primeiro dígito do número de linha, e digitar novamente a partir desta posição.
- O modo ESCAPE permite a correção de apenas uma linha de programa por vez.
- Após alterar uma linha, é importante copiar o restante dela, por meio da tecla [→] (passando o cursor por sobre os caracteres a serem copiados).

4.3. Outros comandos do modo ESCAPE

Existem três outros comandos no modo ESCAPE, destinados a agilizar a edição:

- [esc] [@] (para obter o símbol @, tecle [shift] [2])
  Limpa a tela inteira e posiciona o cursor no canto superior esquerdo do vídeo, exatamente como o comando HOME.

- esc E
  Apaga todos os caracteres desde a posição do cursor até o final da linha.

- esc F
  Apaga todos os caracteres desde a posição do cursor até o final da tela.

4.4. Os limites do modo ESCAPE

Agora que você conheceu os comandos do modo ESCAPE, pretique-os, movendo o cursor para todos os lados da tela. Verifique o que acontece quando ele é movido para além das margens do vídeo. Descubra o que acontece quando você pressiona uma tecla qualquer. Quanto mais se familiarizar com este modo, menos dificuldades você terá durante a edição de seus programas.

Nota: Se você não posicionar o cursor no primeiro dígito do número da linha a ser editada, parte dela será perdida.

4.5. Inserindo caracteres numa linha

Por vezes é necessaria a inserção de algum complemento numa linha de programa já existente. Quando se trata de uma linha um tanto longa, é preferível editá-la e acrescentar o que desejamos, a digitar tudo novamente.

Suponha que a linha já existente seja: 10 PRINT "COMO INSERIR CARACTERES EM UMA LINHA" e que você queira acrescentar ALGUNS antes de CARACTERES.

Para tanto, após ter digitado a linha 10, deve agir da seguinte maneira:

a) Liste-a:

```
] LIST 10 return
```

É exibido:

```
10 PRINT "COMO INSERIR CARACTER
   ES EM UMA LINHA"
```

b) Entre no modo ESCAPE (tecle esc uma vez).

c) Tecle ↑ três vezes, para posicionar o cursor na linha a ser corrigida, e pressione a seta ← até alcançar o primeiro dígito do número de linha.

d) Saia do modo ESCAPE (pressione a barra de espaço ou esc).

e) Use [→] até que o cursor alcance o espaço entre INSERIR e CARACTERES:

```
10 PRINT "COMO INSERIR ■ CARACTER
   ES EM UMA LINHA"
```

f) Acione [esc] para entrar no modo ESCAPE.

g) Pressione [↑], para subir o cursor uma linha.

h) Pressione a barra de espaço, para voltar ao modo normal.

i) Digite: (um espaço) ALGUNS (um espaço). Veja como deve ter ficado a tela

```
                    ALGUNS ■
10 PRINT "COMO INSERIR CARACTER
   ES EM UMA LINHA"
```

j) Entre no modo ESCAPE novamente.

l) Posicione o cursor, por meio das setas direcionais, sobre o "C" de CARACTER.

m) Saia para o modo normal, usando a barra de espaço.

n) Mova o cursor até uma posição imediatamente após o "R" final de CARACTER.

o) Entre em modo ESCAPE e aperte a seta [→], até atingir o S da linha seguinte.

p) Volte ao modo normal.

q) Corra o cursor até passar o último caractere da linha 10, por meio da seta [→].

r) Tecle [return], finalizando a edição.

s) Liste a linha 10, que deve estar assim:

```
10 PRINT "COMO INSERIR ALGUNS C
   ARACTERES EM UMA LINHA"
```

O passo "o" foi necessário para evitar que os espaços em branco entre CARACTER e "ES" da linha seguinte fossem incluídos na inserção. Estes espaços são criados pelo próprio sistema, bem como o formato irregular das linhas durante a listagem.

Nota: Para que as listagens dos programas saiam "compactadas" (sem espaços inúteis), utiliza-se um recurso, que reduz o tamanho da tela de 40 para 33 colunas, deslocando a margem direita 7 posições para a esquerda. Basta digitar POKE 33,33 e return. Para repor a margem direita na coluna 40, digita-se POKE 33,40 e return.

5. Resumo dos Comandos de Edição

5.1. Modo ESCAPE

Para entrar no modo ESCAPE	pressione esc
Para sair do modo ESCAPE	pressione a barra de espaço ou esc
Para mover o cursor para cima	use ↑ , I , D ou control K
Para mover o cursor para baixo	use ↓ , M , C ou control J
Para mover o cursor para a esquerda	use ← , J , B ou control H
Para mover o cursor para a direita	use → , K , A ou control U
Para apagar a partir do cursor até o final da linha	pressione esc E
Para apagar a partir do cursor até o final da tela	pressione esc F
Para apagar a tela inteira	pressione esc @

5.2. Modo texto

Para copiar um caractere	pressione →
Para eliminar uma linha antes de return ter sido pressionada	pressione control X

6. Instruções Relativas à Formatação

6.1. TAB

A instrução TAB é usada para deslocar o cursor horizontalmente. Este comando pode ser utilizado para iniciar a impressão de dados a partir de uma coluna qualquer. Normalmente, associa-se TAB a PRINT.

No exemplo abaixo, TK e OK são apresentados na 10.ª e na 15.ª colunas, respectivamente.

```
]PRINT TAB(10); "TK": PRINT TAB(15);"OK"
```

Nota: Não confundir o comando TAB com a tecla tab.

6.2. VTAB e HTAB

Através de VTAB pode-se mover o cursor para qualquer linha do vídeo. Este movimento sempre é executado no sentido vertical, para baixo ou para cima, sendo que a posição é dada relativamente ao limite superior da tela (linha 1) dentro dos limites 1 a 24. Qualquer valor fora dos limites citados faz com que o TK //e emita a mensagem de erro:

```
? VALOR ILEGAL ERRO
```

A instrução HTAB, é usada de forma semelhante a VTAB, porém move o cursor no sentido horizontal. Neste caso, os limites utilizáveis vão de 1 a 255, relativamente à margem esquerda da tela. Uma vez que a tela só possui 40 posições horizontais, os números 41 a 80 passam a posicionar o cursor na linha seguinte e assim por diante.

Confira no exemplo a seguir as funções destas instruções:

```
]NEW
]10 HOME
]20 VTAB 12
]30 HTAB 18
]50 PRINT "TK //e"
]60 END
```

O nome TK //e é apresentado a partir da linha 12, coluna 18 do vídeo.

6.3. SPC

SPC é a abreviação de espaço (SPaCe). Esta instrução faz com que o cursor seja movido determinados espaços, horizontalmente.

Observe, no próximo exemplo, a difereça entre os efeitos das instruções SPC e TAB:

```
]NEW
]10 PRINT "AAAA"; TAB(15); "BB"
]20 PRINT "CCCC"; SPC(15); "DD"
```

Sendo dado o comando RUN, a tela apresenta:

```
AAAA          BB
CCCC               DD
```

O primeiro B encontra-se na 15.ª coluna, contada a partir do início da tela; enquanto o primeiro D situa-se 15 posições após o último C.

6.4. POS

POS é a abreviatura de posição. Esta instrução faz com que seja apresentada a última posição ocupada pelo cursor numa linha. Normalmente, deve-se colocar um número qualquer entre parênteses depois de POS, por exemplo, POS (Ø) e POS (15) (ambos têm o mesmo significado).

Acompanhe o exemplo abaixo para melhor entender a função deste comando:

```
]NEW
]1Ø PRINT "AAA"; POS (Ø)
]2Ø PRINT "BBBB"; POS(12)
]3Ø PRINT "AAA"; TAB(1Ø); "BBBB"; POS(Ø)
]4Ø PRINT "CCC"; SPC (1Ø); "DDDD"; POS (Ø)
]5Ø PRINT TAB(23); POS (Ø)
]6Ø PRINT SPC(23); POS (Ø)
```

Após o comando RUN, tem-se:

```
AAA3
BBBB4
AAA       BBBB13
CCC         DDDD17       22
                                23
```

Observações: a) Nas linhas 1Ø e 2Ø a instrução POS faz com que seja indicado estritamente o número de caracteres da linha, uma vez que a apresentação destes começa no início da linha, e que não há espaços entre eles.

b) Se você não entendeu o resultado da execução das linhas 3Ø e 4Ø, releia os subitens 6.1 e 6.3 (instruções TAB e SPC, respectivamente).

c) Através dos resultados da execução das linhas 5Ø e 6Ø, pode-se concluir que não é necessário que seja apresentado qualquer caractere na tela, para que a última posição ocupada pelo cursor, em determinada linha, seja diferente de zero.

VI - GRUPOS DE COMANDOS E INSTRUÇÕES

Neste capítulo se encontram novas instruções e comandos. São apresentados dentro de grupos, de acordo com a finalidade a que se destinam. Após sua leitura, você estará a par da maioria das instruções elementares do BASIC e contará com recursos consideráveis para começar a desenvolver seus próprios programas.

1. A Operação com Cadeias

1.1. LEN

Esta instrução faz com que o computador conte o número de caracteres de um string. Ela pode ser usada em dois formatos distintos:

LEN (string) **ou** **LEN (variável string)**

Exemplo:

```
]PRINT LEN (''ANTENA'')
        ou
]A$ = ''ANTENA''
]PRINT LEN (A$)
```

Nos dois formatos o resultado é o mesmo, ou seja, a apresentação do número 6 na tela, visto que ANTENA possui seis caracteres.

Entre o seguinte programa no computador:

```
]NEW
]10 REM CONTAGEM DE CARACTERES
]20 HOME
]30 INPUT ''Qual a palavra ?''; X$
]40 N = LEN (X$)
]50 PRINT ''O numero de letras e '';N
]60 PRINT
]70 GOTO 30
]RUN
```

Note que, neste exemplo, o número de caracteres da cadeia foi atribuído à variável N.

Abaixo, um possível resultado da execução deste programa (para interrompê-lo, pressione Control C ou reset):

```
Qual a palavra ? HOJE
O numero de letras e 4

Qual a palavra ? CARAMBOLA
O numero de letras e 9

Qual a palavra ? ■
```

1.2. LEFT$

Esta instrução faz com que o TK //e selecione caracteres de uma cadeia, a partir da esquerda.

Exemplo:

```
]A$ = "Como vai voce?"
]PRINT LEFT$ (A$,4)
```

Ao pressionar return, temos como resultado os quatro primeiros caracteres da cadeia apresentados na tela, ou seja, a palavra Como.

Experimente rodar o programa a seguir:

```
]NEW
]5 HOME
]10 A$ = "Como vai ?"
]20 FOR I = 1 TO LEN (A$)
]30 PRINT LEFT$ (A$,I)
]40 NEXT I
]50 END
]RUN
```

Obtém-se uma tela assim:

```
C
Co
Com
Como
Como
Como v
Como va
Como vai
Como vai
Como vai ?
```

A explicação do programa é dada abaixo:

HOME (limpa a tela e posiciona o cursor em seu início)
A$ = "Como vai ?" (associa à variável A$ a cadeia "Como vai ?")
FOR I = 1 TO LEN (A$) (define a instrução FOR ... NEXT com o valor inicial da variável I = 1 e o valor final com o número de caracteres da cadeia. Uma vez que neste caso particular a cadeia tem 10 caracteres, esta instrução corresponde a FOR I = 1 TO 10)
PRINT LEFT$ (A$,I) (apresenta os I primeiros caracteres da cadeia)
NEXT I (soma 1 à variável I e desvia o programa para a ordem FOR, até que I atinja seu valor final - no caso, 10)

É importante notar que, para o computador, o espaço em branco também é processado como qualquer outro caractere, por isso, quando a instrução LEN (A$) foi utilizada, ele contou 10: incluiu os dois espaços em branco.

1.3. RIGHT$

A instrução RIGHT$, opera de forma similar à LEFT$ fazendo, porém, com que o TK //e selecione caracteres da direita para a esquerda.

Exemplo:

```
]A$ = "Como vai voce?"
]PRINT RIGHT$ (A$,5)
```

Acionado [return], temos a palavra "voce?" apresentada no vídeo.

Execute agora o programa:

```
]5 HOME
]10 A$ = "Como vai ?"
]20 FOR I = 1 TO LEN (A$)
]30 PRINT RIGHT$ (A$,I)
]40 NEXT I
]RUN
```

A tela apresenta:

```
?
 ?
i ?
ai ?
vai ?
 vai ?
o vai ?
mo vai ?
omo vai ?
Como vai ?
```

O fluxo deste programa é idêntico ao apresentado no subitem anterior, exceto na instrução PRINT RIGHT$ (A$,I), através da qual são apresentados os I últimos caracteres da cadeia.

1.4. MID$

MID$ faz com que o computador selecione caracteres intermediários de uma cadeia. Por exemplo, a instrução MID$ (A$,3,5) permite a seleção de cinco caracteres a partir do terceiro caractere da cadeia A$.

Digite agora:

```
]PRINT MID$ (''Como vai?'',2,3)
```

Pressionando return, você vê os caracteres "omo", ou seja, três caracteres a partir do segundo da cadeia.

Entre o programa a seguir:

```
]NEW
]5 HOME
]10 A$ = ''Como vai ?''
]20 FOR I = 1 TO LEN (A$)- 5
]30 PRINT MID$ (A$,6,I)
]40 NEXT I
]RUN
```

A tela mostra:

```
v
va
vai
vai
vai ?
```

Perceba que, na instrução FOR I = 1 TO LEN(A$)- 5, foi subtraído 5, porque somente existem cinco caracteres a serem apresentados. Assim, esta instrução é equivalente a:

```
FOR I = 1 TO 10-5        ou        FOR I = 1 TO 5
```

Para testar sua aprendizagem das instruções tratadas neste capítulo, digite o programa abaixo e, sem olhar a figura que segue a listagem do programa, procure prever os resultados:

```
]NEW
]5 HOME
]10 A$ = ''Seu comodo''
]20 PRINT A$
]30 PRINT LEN (A$)
]40 PRINT LEFT$ (A$,2)
]50 PRINT RIGHT$ (A$,3)
]60 PRINT MID$ (A$,2,6)
```

Dê o comando RUN e confira:

```
Seu comodo
10
Se
odo
eu com
```

2. A Operação com Dados Numéricos

As operações com dados numéricos são apresentadas de forma mais ampla em capítulos posteriores. Por ora, damos apenas uma introdução a este assunto, examinando RND e INT.

2.1. RND

O nome desta instrução corresponde à abreviatura de randômico. Um conjunto de números randômicos é um conjunto de números sem qualquer relação entre si, aleatórios. Não existe função que possa representá-los. Se você não tem formação matemática, pode ser-lhe difícil entender este conceito, mas o que importa é observar sua ação e saber que, em determinadas aplicações como, por exemplo, jogos, RND é uma instrução especialmente útil.

RND(1) faz com que o TK //e selecione números randômicos entre Ø e 1 (o argumento pode ser qualquer valor inteiro maior que zero).

Digite o exemplo:

```
]NEW
]5 HOME
]10 FOR I = 1 TO 10
]20 PRINT RND(1),
]30 NEXT I
```

Na primeira vez em que você roda este programa, o TK //e apresenta os resultados indicados a seguir:

```
]RUN
.973136996      .103117626
.0177148333     .779343355
.551834438      .617419111
.960296981      .547150891
.802192734      .814107273
```

Observe que, cada vez que o programa é novamente rodado, os resultados são diferentes, mas sempre com números entre Ø e 1.

Notas: 1) Sendo zero o argumento de RND, o resultado é igual ao do último RND fornecido.

2) O argumento negativo fornece o mesmo valor todas as vezes em que programa é rodado. Um argumento positivo usado posteriormente, fornecerá uma lista de valores aleatórios, que será apresentada outra vez quando o programa for rodado novamente.

Para obter seqüências de dez números entre 0 e 10, basta digitar:

```
]NEW
]5 HOME
]10 FOR I = 1 TO 10
]20 PRINT 10 * RND (1),
]30 NEXT I
```

Rode este programa algumas vezes e observe que os resultados dificilmente são repetidos.

2.2. INT

A função INT faz com que o computador despreze a parte fracionária de um número real maior que ou igual a zero. Se o número é negativo, fornece o inteiro imediatamente inferior.

Exemplo:

```
]PRINT INT(1.896)
]PRINT INT(2.571)
]PRINT INT(0.62528)
]PRINT INT(-3.587)
```

Os comandos acima dão como resultado, respectivamente: 1, 2, 0 e -4.

É bastante freqüente a necessidade de gerarem-se números randômicos inteiros. Para tal, utilizamos as instruções INT e RND em conjunto.

Programa-exemplo para a seleção de cinco números (entre 0 e 99):

```
]NEW
]5 HOME
]10 PRINT "Os numeros selecionados sao: ":PRINT
]20 FOR I = 1 TO 5
]30 PRINT INT( 100 * RND(1)),
]40 NEXT I
```

Observe que, cada vez que se executa o programa, obtém-se um conjunto diferente de números.

Através dos exemplos abaixo, entenda porque o RND(1) foi multiplicado por 100:

RND(1)	100*RND(1)	INT(100*RND(1))
0.00125	0.125	0
0.99978	99.978	99

Para gerar 10 números inteiros entre 1 e 10, pode-se lançar mão do seguinte programa:

```
]NEW
]10 FOR I = 1 TO 10
]20 PRINT INT ( 10 * RND(1)) + 0.49
]30 NEXT I
```

3. Os Comandos Relacionados à Execução de Programas

3.1. STOP, CONT e END

Tanto o comando STOP como END fazem com que o TK //e interrompa a execução de um programa. STOP pode ser usado várias vezes num programa. END também pode ser usado várias vezes, no entanto, para facilitar a leitura da listagem, costuma-se empregar END uma única vez, somente na finalizacão do processamento.

3.1.1. STOP

Quando o computador executa um STOP, apresenta a mensagem: PAROU EM xx, onde xx representa o número da linha de programa em que o processamento foi interrompido.

Exemplo:

```
]NEW
]10 FOR I = 1 TO 10
]20 PRINT I;
]30 NEXT I
]40 STOP
]RUN
```

A tela exibe:

```
12345678910
PAROU EM 40
```

3.1.2. CONT

Sendo o programa interrompido com STOP, pode-se dar prosseguimento à sua execução através do comando CONT.

Exemplo:

```
]NEW
]10 FOR I = 1 TO 2
]20 PRINT I;
]30 NEXT I
]40 STOP
]50 FOR I = 1 TO 5
]60 PRINT I;
]70 NEXT I
```

Depois do comando RUN a tela mostra:

```
12
PAROU EM 40
```

Sendo introduzido o comando CONT, o programa passa a executar as linhas 50 a 70:

```
]CONT    return
12345
```

3.1.3. END

A instrução END pode ser usada da mesma forma que STOP, porém nenhuma mensagem é apresentada.

```
]NEW
]10 FOR I = 1 TO 2
]20 PRINT I;
]30 NEXT I
]40 END
]50 FOR I = 1 TO 5
]60 PRINT I;
]70 NEXT I
]RUN
```

Resultado: 12

O comando CONT também pode ser usado quando o programa é interrompido pela instrução END.

Digitando CONT, obtém-se:

```
]CONT
12345
```

3.2. Control-C

A teclas control e C, pressionadas em conjunto, fazem com que o TK //e execute função idêntica à que realiza através da instrução STOP: a interrupção do programa e a indicação da linha onde se deu a pausa. Mas, diferentemente de STOP, pode-se acionar control C em qualquer ponto do programa.

Exemplo:

```
]NEW
]10 PRINT "Bom dia"
]20 GOTO 10
]RUN
```

Rode o programa e interrompa-o com control C:

```
Bom dia
Bom dia
Bom dia
Bom dia
PAROU EM 10
```

A execução do programa pode ter prosseguimento através do comando CONT.

Nota: control C também "limpa" o buffer de entrada.

3.3. TRACE e NOTRACE

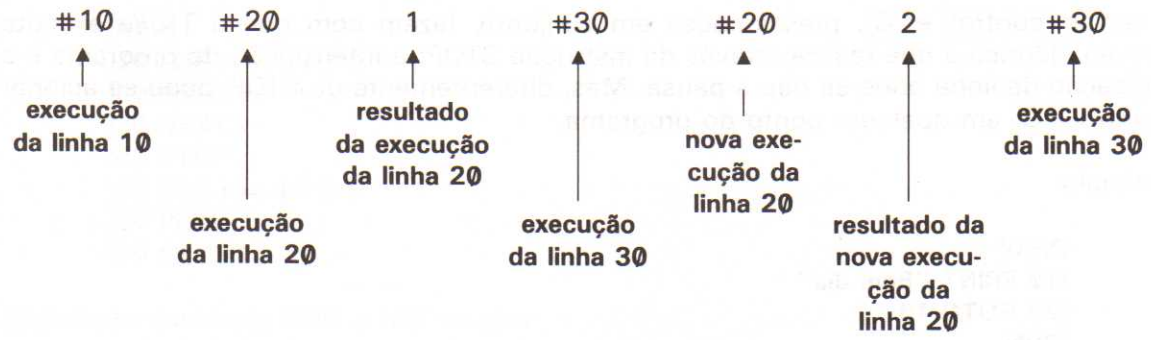
Para acompanhar o processo de execução de um programa, pode-se usar TRACE. Antes de executar o programa, deve-se digitar TRACE e teclar return. Quando o programa é rodado, o computador apresenta o número de cada linha à medida em que a executa.

Siga o exemplo:

```
]NEW
]10 FOR I = 1 TO 2
]20 PRINT I;
]30 NEXT I
]TRACE
]RUN

 #10  #20  1 #30  #20  2 #30
```

Análise do resultado deste programa:



A principal função de TRACE é auxiliar o programador a descobrir erros durante a elaboração de um programa.

O único comando que desativa o modo TRACE é NOTRACE. Uma vez que TRACE esteja acionado, NEW ou reset não são capazes de fazer cessar sua atividade.

3.4. CLEAR

O comando CLEAR faz com que se "limpem" da memória todas as variáveis existentes.

Exemplo:

```
]A = 7
]PRINT A
```

Como resultado, obtém-se o número 7.

Agora faça:

```
]CLEAR
]PRINT A
```

Em vez de 7, passa a ser apresentado Ø (o zero é apresentado sempre que, ao se usar o PRINT, não haja valor algum atribuído à variável numérica utilizada).

3.5. FRE(Ø)

Esta função reorganiza a memória e informa a área ou a quantidade de memória RAM disponível, ou seja, a área ainda não ocupada pelo programa em BASIC e pelas variáveis.

A área dedicada a páginas de alta-resolução é considerada livre, mesmo que esteja sendo usada.

FRE(Ø) deve ser usado juntamente com PRINT.

O resultado é negativo se a área ocupada é menor que 32.767.

Seu formato é:

```
]PRINT FRE(Ø)
```

Caso o valor seja negativo, para obter o tamanho real da memória disponível, basta somar 65.536 ao número apresentado através da instrução acima.

O uso mais importante do comando FRE(Ø) faz-se na reorganização da memória, quando se usam muitas variáveis string (sobre economia de memória, ver apêndice C).

3.6. INVERSE E NORMAL

O comando INVERSE faz com que os caracteres apareçam no vídeo em escuro sobre fundo claro.

O comando NORMAL faz com que a aparência dos caracteres volte à situação original: em claro sobre fundo escuro.

3.7. FLASH

O comando FLASH tem como função fazer com que uma determinada mensagem fique piscando na tela.

O comando NORMAL faz com que as próximas mensagens a serem exibidas apareçam sem piscar.

3.8. SPEED

Através de SPEED, pode-se determinar a velocidade com a qual os caracteres são apresentados na tela. Se SPEED = Ø, a velocidade é a mais baixa possível; se SPEED = 255, a velocidade é a mais alta possível. Sendo escolhido um número maior que 255, ocorre uma mensagem de erro:

```
? VALOR ILEGAL ERRO
```

Note a diferença de velocidade na contagem de 1 a 5 e na de 6 a 1Ø, através do exemplo a seguir:

```
]NEW
]1Ø SPEED = 1
]2Ø FOR I = 1 TO 5
]3Ø PRINT I;" ";
]4Ø NEXT I
]5Ø SPEED = 255
]6Ø FOR I = 6 TO 1Ø
]7Ø PRINT I;" ";
]8Ø NEXT I
]RUN
```

3.9. Control-X

O conjunto das teclas [control] e [X] faz com que um sinal de barra \ seja colocado na linha onde se encontra o cursor, tornando a linha inexistente para o computador.

Digite:

```
]NEW
]10 PRINT "BARRA" (não tecle [return])
```

E acione [control] [X]. A linha acima toma a seguinte forma:

```
]10 PRINT "BARRA" \
```

Agora digite RUN e observe que a linha é anulada, não produzindo qualquer efeito.

4. Acesso à Memória RAM

A utilização dos comandos POKE e PEEK requer experiência em programação. Se você não tem uma certa prática e algum conhecimento do hardware do microcomputador, não é necessário que se detenha muito neste item.

4.1. POKE

POKE serve para introduzir dados diretamente na memória RAM do computador. Para efetuar a introdução, devem-se indicar duas coisas:

a) o endereço da memória RAM para o qual se pretende enviar o dado desejado;

b) o dado a ser introduzido.

Exemplo:

```
]POKE 1500 , 56
     endereço dado
     (decimal) (decimal)
```

POKE 1500,56 indica a introdução do valor 56, no endereço 1500.

Como os dados são armazenados em bytes (8 bits), o dado a ser introduzido diretamente na memória deve estar compreendido entre 0 e 255. A tentativa de introdução de um número fora destes limites provoca a ocorrência da mensagem:

```
?VALOR ILEGAL ERRO
```

Esta mensagem também é apresentada se o endereço é maior que 65.535, o maior endereço da memória RAM do TK //e.

4.2. PEEK

Desejando-se ter acesso direto a um dado de um determinado endereço da memória RAM do TK //e, usa-se a instrução PEEK, juntamente com o endereço desejado.

Por exemplo:

```
]10 A = PEEK (120)
]20 PRINT A
```

Ao se executar o programa acima, o conteúdo do endereço 120 é apresentado na tela.

Note que, se você executou o exemplo apresentado no início deste capítulo (POKE 1500, 56), ao digitar:

```
]10 A = PEEK(1500)
]20 PRINT A
]RUN
```

Obtém na tela o número 56. Justamente o conteúdo introduzido no endereço 1500 através da instrução POKE.

VII - OPERAÇÕES NUMÉRICAS, COMPARATIVAS E LÓGICAS

Seu microcomputador realiza operações com números e também com variáveis, sejam estas numéricas ou do tipo cadeia.

O TK //e realiza as seguintes operações numéricas:

a) operações algébricas, tais como: +, -, *, /, ^;

b) operações comparativas, tais como: >, <, =, >=, <=;

c) operações lógicas, tais como: NOT, AND e OR.

1. As Operações Numéricas

Você já estudou as operações algébricas no item 2 do capítulo II deste manual. Aprendeu a fazer cálculos no modo imediato. No modo programado, emprega-se aquela mesma simbologia, para todas as operações.

2. As Operações Comparativas

Através do TK //e podem-se realizar operações comparativas entre números.

A tabela a seguir indica os tipos de operações comparativas possíveis, bem como os respectivos símbolos usados pelo TK //e:

OPERAÇÃO	SÍMBOLO
menor que	<
maior que	>
igual a	=
diferente de	<>
maior que ou igual a	>=
menor que ou igual a	<=

Sendo a comparação verdadeira, o TK //e fornece o valor 1; sendo falsa, fornece o valor Ø.

Observe o programa:

```
]NEW
]10 PRINT 5>4
]20 PRINT 4=5-1
]30 PRINT 10=3*4
]40 PRINT 3<=6/2
]50 PRINT 9-8>=5
]60 PRINT 8-5<4
]70 PRINT 8-3<>15/3
```

Antes de rodar o programa, vamos prever os resultados:

linha	verdadeiro/falso	resultado
10	V	1
20	V	1
30	F	0
40	V	1
50	F	0
60	V	1
70	F	0

3. As Operações Lógicas

Veja na tabela abaixo as operações lógicas disponíveis e os respectivos símbolos:

OPERAÇÃO	SÍMBOLO
E	AND
OU	OR
NEGAÇÃO	NOT

Cada um destes operadores lógicos tem sua tabela verdade, como você vê a seguir:

TABELA VERDADE DA OPERAÇÃO E	
Entrada	Saída
1 AND 1	1
1 AND 0	0
0 AND 1	0
0 AND 0	0

TABELA VERDADE DA OPERAÇÃO OU	
Entrada	Saída
1 OR 1	1
1 OR Ø	1
Ø OR 1	1
Ø OR Ø	Ø

TABELA VERDADE DA OPERAÇÃO NOT	
Entrada	Saída
NOT 1	Ø
NOT Ø	1

Acompanhe alguns exemplos do emprego das operações comparativas combinadas com as operações lógicas:

1.º)]**PRINT (5>4) AND (3>2)**
1

A operação acima pode ser desdobrada em duas partes a das operações comparativas e a da operação lógica:

Operações Comparativas		
5>4	verdadeiro	1
3>2	verdadeiro	1

Operação Lógica		
1 AND 1	ver tabela verdade AND	1 1

2.º]**PRINT (3<2) OR (1<Ø)**
Ø

Vamos esquematizar da mesma forma que no exemplo anterior:

Operações Comparativas		
3<2	falso	Ø
1<Ø	falso	Ø

Operação Lógica		
Ø OR Ø	ver tabela verdade OR	Ø

3.º)]PRINT NOT (5>4)
0

Utilizando o mesmo esquema dos exemplos anteriores, temos:

Operação Comparativa		
5>4	verdadeiro	1

Operação Lógica		
NOT 1	ver tabela NOT	0

Agora digite o programa a seguir, procurando prever seus resultados antes de rodá-lo:

```
]NEW
]10 PRINT NOT ((3 + 4) >= 4)
]20 PRINT (9=8) OR (4 * 6 > 3 * 4)
]30 PRINT (9 > 8) AND (5 >3)
]40 PRINT (9 * 2 > 3 * 5) AND (2 * 3 > 54 * 2)
]50 PRINT NOT (3 = 2)
]60 PRINT (2 = 3) OR (21 = 7 * 3)
```

4. Outras Operações com Cadeias

Como você já sabe, o símbolo $ deve ser posposto à variável tipo cadeia. No TK //e as operações possíveis com tal variável são as indicadas abaixo:

. LEN (A$)	. STR$ (X)	. VAL (A$)
. CHR$ (X)	. ASC (A$)	. LEFT$ (A$,X)
. RIGHT$ (A$,X)	. MID$ (A$,X,Y)	. +

Dentre estas há cinco operações que ainda não foram estudadas e que são apresentadas neste item: STR$ (X), + , VAL (A$), CHR$ (X) e ASC (A$).

4.1. STR$ (X) e +

STR$ representa cadeia (string). Em STR$ (X) o valor do número X, entre parênteses, é convertido numa cadeia.

Exemplo:

```
]NEW
]10 A = 123
]20 B = 456
]30 A$ = STR$(A)
]40 B$ = STR$(B)
]50 PRINT A$ + B$
```

Após o comando RUN tem-se:

```
123456
```

Os dados 123 e 456 não foram tratados como números (cuja soma seria 579). Foram tratados como cadeias. O sinal + fez com que fossem apresentados em ordem seqüencial: 123456.

O programa seguinte distribui sete números em dois grupos, separados por um hífen, como se usa na notação dos números telefônicos:

```
]NEW
]10 HOME
]20 INPUT "Numero do tel. ";A
]30 B$ = STR$(A)
]35 IF LEN (B$) <>7 THEN GOTO 80
]40 C$ = LEFT$(B$,3)
]50 D$ = RIGHT$(B$,4)
]60 PRINT C$ + "-" + D$
]70 GOTO 20
]80 HTAB1: VTAB20: PRINT "Digite 7 digitos": FOR S = 1 TO 1500: NEXT S:
GOTO 10
```

Antes de rodar este programa, analisemos sua seqüência de instruções:

HOME (limpa a tela e posiciona o cursor em seu início)
INPUT "Numero do tel.: ";A (apresenta o texto "Numero do tel.:" e aguarda a digitação de um número de 7 dígitos a ser associado à variável numérica A)
B$ = STR$(A) (associa a variável numérica A à variável tipo cadeia B$)
C$ = LEFT$(B$,3) (associa os três primeiros caracteres da cadeia B$ à variável tipo cadeia C$)
D$ = RIGHT$(B$,4) (associa os quatro últimos caracteres à variável tipo cadeia D$)
PRINT C$ + "-" + D$ (faz com que sejam apresentados em ordem seqüencial: a cadeia C$, o caractere hífen (-) e a cadeia B$)
GOTO 20 (desvia o programa para a linha 20)

A figura abaixo apresenta um exemplo de tela que poderia ser gerada por este programa:

```
Numero do tel.: 2879741
287-9741
Numero do tel.: 5475127
547-5127
Numero do tel.: 8818969
881-8969
Numero do tel.:
```

4.2. VAL (A$)

VAL é a abreviatura de valor. A instrução VAL(A$) executa função inversa à da instrução STR$. Observe o exemplo abaixo:

```
]NEW
]10 A$ = "12"
]20 B$ = "13"
]30 X$ = A$ + B$
]40 Y = VAL(A$) + VAL(B$)
]50 Z = VAL(A$) * VAL(B$)
]60 PRINT X$,Y,Z
```

Após o comando RUN, são apresentados os números:

```
1213          25          156
```

Repare que, enquanto X$ corresponde à soma das cadeias "12" e "13", Y vale a soma dos números 12 e 13.

4.3. O Código ASCII

ASCII é a abreviação de "American Standard Code for Information Interchange", ou seja, Código Americano Padrão para Intercâmbio de Informações.

Como o computador opera com números binários, podemos representar os caracteres alfabéticos (A,B,C...), os numéricos (0,1,2,...) ou os simbólicos (+,-,*,/,...) por meio de um código binário, como o ASCII.

No código ASCII, a letra A é representada pelo número binário 01000001 (= 65 no sistema decimal), e a letra L é representada por 01001100 (= 76 na base decimal). Cada caractere alfabético, numérico ou simbólico corresponde a um código ASCII específico.

A tabela a seguir apresenta o código ASCII, na base decimal, dos caracteres alfabéticos maiúsculos:

CARACTERES	ASCII NA BASE
A	65
B	66
C	67
D	68
E	69
F	70
G	71
H	72
I	73
J	74
K	75
L	76
M	77

CARACTERES	ASCII NA BASE
N	78
O	79
P	80
Q	81
R	82
S	83
T	84
U	85
V	86
W	87
X	88
Y	89
Z	90

Na linguagem BASIC, usam-se duas instruções para converter valores ASCII em caracteres e vice-versa, conforme explicamos nos próximos subitens.

4.4. ASC(A$)

ASC é abreviação de ASCII. O formato básico desta instrução é:

```
ASC (cadeia)
```

Note que, embora se possa incluir uma cadeia entre parênteses após a instrução ASC, apenas o primeiro caractere desta tem seu valor calculado. Execute os comandos abaixo para comprovar tal fato:

```
]PRINT ASC("A")
65

]PRINT ASC("AB")
65
```

O TK //e apresenta 65 nos dois exemplos. Assim, concluímos que o caractere B não foi computado. Constatamos, ainda, que o valor foi apresentado no código ASCII, em base decimal.

Os parênteses após ASC podem ser usados também com outras operações com cadeias. Veja nos exemplos a seguir que os resultados são os mesmos. As duas ordens das linhas 20 e 30 do primeiro programa são as da linha 20 do segundo programa:

1.º)
```
]NEW
]10 A$ = "MICRO"
]20 B$ = MID$(A$,3,1)
]30 PRINT B$ ; "=" ; ASC(B$)
```

2.º)
```
]NEW
]10 A$ = "MICRO"
]20 PRINT MID$(A$, 3, 1);"="; ASC( MID$( A$, 3, 1))
```

Nos exemplos apresentados, sendo A$ relacionado à cadeia MICRO, obtemos o terceiro caractere através da instrução MID$(A$, 3, 1). O resultado apresentado é 67, que é o código ASCII decimal de C.

Para fixar o conceito, execute o programa a seguir:

```
]NEW
]10 A$ = "MICRO"
]20 B = LEN (A$)
]30 FOR I = 1 TO B
]40 PRINT MID$ (A$,I,1), ASC (MID$(A$,I,1))
]50 NEXT I
```

Certamente você consegue prever o resultado deste programa. Confira:

```
M     77
I     73
C     67
R     82
O     79
```

4.5. CHR$(X)

CHR$ representa caractere (CHaRacter). CHR$ é empregado para transformar o código ASCII decimal em caracteres ASCII.

Exemplo:

```
]PRINT CHR$(65)
A
```

Observe que esta é a operação inversa de PRINT ASC("A").

Construa uma tabela de valores decimais e seus correspondentes símbolos ASCII, executando o próximo programa (o programa começa com o valor decimal 32, porque antes deste número há alguns códigos de tela que podem influenciar a operação do computador; termina em 127 porque, a partir deste número, os caracteres repetem-se até o final do código, que é 255):

```
]NEW
]10 HOME
]20 FOR I = 32 TO 127
]30 PRINT I;" = "; CHR$ (I),
]40 FOR K = 1 TO 500: NEXT K
]50 NEXT I
```

Após o comando RUN, observa-se a formação da tabela:

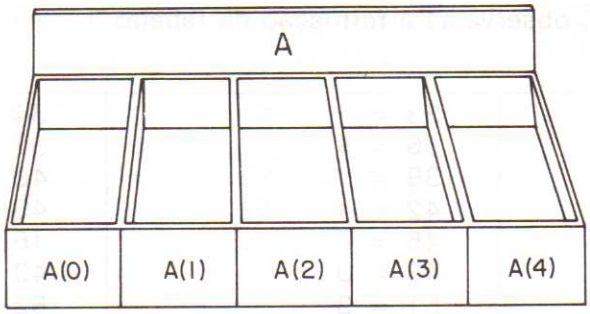
32 =	33 =	34 = ''
35 = #	36 = $	37 = %
38 = &	39 = '	40 = (
41 =)	42 = *	43 = +
44 = ,	45 = -	46 = .
47 = /	48 = 0	49 = 1
50 = 2	51 = 3	52 = 4
53 = 5	54 = 6	55 = 7
56 = 8	57 = 9	58 = :
59 = ;	60 = <	61 = =
62 = >	63 = ?	64 = @
65 = A	66 = B	67 = C
68 = D	69 = E	70 = F
71 = G	72 = H	73 = I
74 = J	75 = K	76 = L
77 = M	78 = N	79 = O
80 = P	81 = Q	82 = R
83 = S	84 = T	85 = U
86 = V	87 = W	88 = X
89 = Y	90 = Z	91 = [
92 = \	93 =]	94 = ^
95 = _	96 = `	97 = a
98 = b	99 = c	100 = d
101 = e	102 = f	103 = g
104 = h	105 = i	106 = j
107 = k	108 = l	109 = m
110 = n	111 = o	112 = p
113 = q	114 = r	115 = s
116 = t	117 = u	118 = v
119 = w	120 = x	121 = y
122 = z	123 = {	124 = !
125 = }	126 = ~	127 = ⊡

Note que o caractere correspondente ao número 32 está em branco. Para o computador, o espaço em branco também é considerado um caractere (com valor ASCII decimal = 32).

Aperte a tecla mode, entrando em português e veja o que acontece!

5. As Matrizes

Até aqui você conheceu variáveis relacionadas a somente um único valor por vez. Através das matrizes, pode-se associar mais de um valor a uma variável. Por exemplo, suponha que haja uma fileira de cinco caixas. Podemos nos referir a esta fileira nome A e a cada caixa, individualmente, pelos nomes A(0), A(1), A(2), A(3) e A(4), conforme ilustra a figura abaixo:

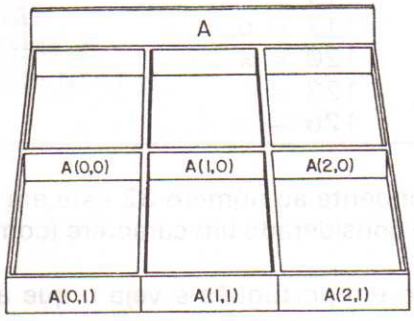


A matriz representada acima é chamada de unidimensional. Os números apresentados entre parênteses especificam cada um dos elementos da matriz e são chamados de índices.

De modo semelhante, podemos ter matrizes bidimensionais. Assim, os preços de veículos, de acordo com o ano de fabricação, podem ser registrados numa matriz bidimensional. Para saber o preço de determinado veículo, devemos fornecer o modelo do veículo (na coluna) e o ano de fabricação (linha):

MODELO	1986	1985	1984...
Passat LS	89.740,32	69.191,78	59.732,52
Ford Corcel L	81.605,71	62.640,12	58.631,67
Belina LDO	87.934,50	65.697,15	58.141,12
Monza SL	95.245,20	89.937,28	73.234,54
Fiat L	48.165,29	40.341,52	35.437,26

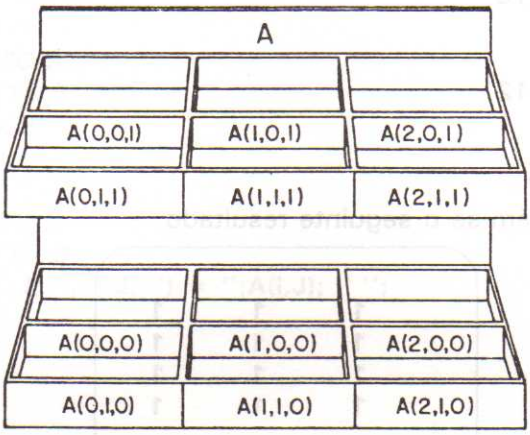
A figura a seguir representa uma matriz bidimensional:



A matriz representada tem dimensões de 2 X 3, ou seja, duas linhas por três colunas.

Numa matriz tridimensional, além de colunas e de linhas, temos níveis. Suponha que não saibamos a numeração dos apartamentos de um prédio. Poderemos localizar um determinado apartamento dizendo: "3.º andar (nível), de frente para a rua (linha), no canto direito (coluna)".

Na figura a seguir é representada uma matriz tridimensional, cujas dimensões são 2 X 3 X 2 (2 linhas, por 3 colunas, por 2 níveis).



Poderíamos, ainda, ter matrizes com mais de três dimensões, mas seria impossível representá-las visualmente.

No TK //e as matrizes podem pertencer a três grupos distintos:

a) variáveis-tipo-matriz inteiras, como: A%(8,5), B%(2,4);

b) variáveis-tipo-matriz reais, como: A(9,9), D(5,9);

c) variáveis-tipo-matriz cadeias, como: A$(1Ø,5), C$(6,8).

5.1. DIM

A instrução DIM é imprescindível para o uso de matrizes num programa. Esta instrução permite que se dimensione, na memória do TK //e, a área necessária para conter todos os dados de cada matriz.

5.2. Matrizes unidimensionais

DIM A(2Ø) é uma matriz unidimensional. Indica que o nome da matriz é A e que existem ao todo 21 variáveis, de A(Ø) até A(2Ø). O método de acesso a estas 21 variáveis é idêntico ao normalmente usado para dar acesso a qualquer variável comum. Toda variável contida numa matriz, de qualquer dimensão, sempre é acessível.

Por exemplo, pode-se usar uma ordem como: A(5) = 32 + A(6). Esta ordem faz com que, após adicionar 32 ao número associado à variável A(6), coloque-se o resultado em A(5). Os números 5 e 6 são os índices. Os índices são usados para se ter acesso a uma determinada variável (ou elemento) de uma matriz.

O programa a seguir atribui a todos os elementos de uma matriz o valor 1:

```
]NEW
]10 DIM A(12)
]20 FOR I = 1 TO 12
]30 A(I) = 1
]40 NEXT I
]50 FOR I = 1 TO 12
]60 PRINT A(I),
]70 NEXT I
```

Ao rodar o programa, obtém-se o seguinte resultado:

```
 1    1    1
 1    1    1
 1    1    1
 1    1    1
```

Vamos analisar o fluxo deste programa:

DIM A(12) (dimensiona a matriz A como unidimensional de 13 elementos)
FOR I = 1 TO 12 (faz com que I varie de 1 a 12)
A(I) = 1 (atribui ao elemento (I) da matriz A o valor 1)
NEXT I (acrescenta 1 à variável e volta à linha 30. Se I = 12, segue o programa)
FOR I = 1 TO 12 (faz com que I varie de 1 a 12)
PRINT A(I), (apresenta o elemento (I) da matriz A)
NEXT I (acrescenta 1 à variável e volta à linha 60. Se I = 12, segue o programa)

Agora execute o programa:

```
]NEW
]10 DIM A%(9)
]20 A%(0) = 0
]25 PRINT "A%(";0;")=";A%(0),
]30 FOR I = 1 TO 9
]40 E = I - 1
]50 A%(I) = A%(E) + I
]60 PRINT "A% (";I;") = ";A%(I),
]70 NEXT I
```

Resultado:

```
A%(0) = 0          A%(1) = 1
A%(2) = 3          A%(3) = 6
A%(4) = 10         A%(5) = 15
A%(6) = 21         A%(7) = 28
A%(8) = 36         A%(9) = 45
```

5.3. Matrizes bidimensionais

O programa seguinte mostra-lhe como utilizar matrizes de duas dimensões:

```
]NEW
]10 DIM A(3,3)
]20 FOR I = 0 TO 3
]30 FOR J = 0 TO 3
]40 A(I,J) = I * J
]50 NEXT J
]60 NEXT I
]70 FOR I = 0 TO 3
]80 FOR J = 0 TO 3
]85 PRINT "A("; I; ",("; J; ") = ";A(I,J); " ";
]90 NEXT J
]100 PRINT
]110 NEXT I
]120 END
```

Resultado:

```
A(0,0) = 0    A(0,1) = 0    A(0,2) = 0    A(0,3) = 0
A(1,0) = 0    A(1,1) = 1    A(1,2) = 2    A(1,3) = 3
A(2,0) = 0    A(2,1) = 2    A(2,2) = 4    A(2,3) = 6
A(3,0) = 0    A(3,1) = 3    A(3,2) = 6    A(3,3) = 9
```

5.4. Matrizes tridimensionais

Observe, no exemplo a seguir, o uso de matrizes de três dimensões:

```
]NEW
]10 DIM A(3,3,3)
]20 FOR I = 0 TO 3
]30 FOR J = 0 TO 3
]40 FOR K = 0 TO 3
]50 A(I,J,K) = I * J * K
]60 NEXT K : NEXT J : NEXT I
]70 FOR I = 0 TO 3
]80 FOR J = 0 TO 3
]90 FOR K = 0 TO 3
]100 PRINT "A("; I; ",("; J; ",("; K; ") = "; A(I,J,K); " ";
]110 NEXT K
]130 PRINT
]140 NEXT J : NEXT I
```

Verifique você mesmo o resultado deste programa.

VIII - INSTRUÇÕES DE ENTRADA E SAÍDA

As instruções descritas neste capítulo têm por função a introdução e a obtenção de dados num programa, ou a apresentação de dados na tela.

Uma instrução de entrada e uma de saída já foram utilizadas em capítulos anteriores; são elas respectivamente: INPUT e PRINT. Os itens a seguir apresentam as novas instruções.

1. DATA ... READ

As instruções DATA e READ são sempre usadas em conjunto. Se um programa contém uma ou mais instruções DATA, deve ter pelo menos uma instrução READ.

A instrução DATA permite definir um conjunto de dados separados por vírgulas. Tais dados devem ser lidos seqüencialmente pela instrução READ. Normalmente, por uma questão de organização, a instrução DATA é incluída no início ou no final do programa, porém nada impede que seja colocada em qualquer outra parte.

Os exemplos a seguir esclarecem a utilização destas instruções:

1.º)
```
]NEW
]10 DATA "GIRAFA", "EMA", "TATU", "PACA"
]20 HOME
]30 READ A$
]40 READ B$
]50 READ C$
]60 READ D$
]70 PRINT A$
]80 PRINT B$
]90 PRINT C$
]100 PRINT D$
```

A figura apresenta os resultados deste programa:

```
GIRAFA
EMA
TATU
PACA
]
```

Nota: As aspas da linha 10 são dispensáveis.

2.º) O resultado do primeiro exemplo também é obtido se se usa apenas uma vez a instrução READ:

Mude a linha 30 para:

```
]30 READ A$,B$,C$,D$
```

E elimine as linhas 40, 50 e 60.

3.º) Pode-se usar apenas um nome de variável:

```
]30 FOR I = 1 TO 4
]40 READ A$
]50 PRINT A$
]60 NEXT I
```

4.º) Os dados utilizados podem ser distribuídos em mais de uma instrução DATA. Esta modalidade é particularmente útil quando a quantidade de dados é muito grande:

```
]NEW
]10 DATA "GIRAFA"
]20 DATA "EMA", "TATU"
]30 DATA "PACA"
]40 HOME
]50 FOR I = 1 TO 4
]60 READ A$
]70 PRINT A$
]80 NEXT I
```

5.º) Instrução DATA utilizada no final do programa:

```
]NEW
]10 HOME
]20 FOR I = 1 TO 4
]30 READ A$
]40 PRINT A$
]50 NEXT I
]60 DATA "GIRAFA", "EMA", "TATU", "PACA"
```

6.º) Instrução DATA incluída no desenvolvimento do programa:

```
]NEW
]10 HOME
]20 DATA "GIRAFA", "EMA", "TATU", "PACA"
]30 FOR I = 1 TO 4
]40 READ A$
]50 PRINT A$ ]60 NEXT I
```

7.º) Os dados contidos na instrução DATA podem ser cadeias, inteiros ou reais. Deve-se apenas ter o cuidado de associar ao dado o devido nome-de-variável. O programa abaixo utiliza dados tipo cadeia e reais na instrução DATA:

```
]NEW
]10 DATA 2,3,2,4,"GIRAFAS", "EMAS"
]20 DATA "TATUS", "PACAS"
]30 HOME
]40 READ A,B,C,D,A$,B$,C$,D$
]50 PRINT A;" ";A$
]60 PRINT B;" ";B$
]70 PRINT C;" ";C$
]80 PRINT D;" ";D$
```

Após o comando RUN, tem-se uma tela assim:

```
2 GIRAFAS
3 EMAS
2 TATUS
4 PACAS
```

8.º) Os dados lidos pela instrução READ e associados a nomes de variáveis podem ser usados para qualquer tipo de operação:

```
]NEW
]10 DATA 4,2,8,3
]20 READ A,B,C,D
]30 E = A ^ B / C + D
40 PRINT E
```

O resultado deste programa-exemplo é a apresentação do número 5 na tela (4 ^ 2 / 8 + 3 = 16 / 8 + 3 =2 + 3 = 5).

9.º) O emprego das instruções DATA .. READ em conjunto com as matrizes é uma importante ferramenta de programação:

```
]NEW
]10 DIM A(8)
]20 FOR I = 1 TO 8
]30 READ A(I)
]40 NEXT I
]45 FOR I = 1 TO 8
]50 PRINT "A(";I;") = ";A(I),
]60 NEXT I
]70 DATA 12,13,43,21,56,78,9,102
```

Ao comando RUN, obtém-se:

```
A(1) = 12     A(2) = 13
A(3) = 43     A(4) = 21
A(5) = 56     A(6) = 78
A(7) = 9      A(8) = 102
```

10.º) Se todos os dados da instrução DATA não são utilizados, o computador ignora os dados restantes:

```
]NEW
]10 DATA 51,62,74,55,92,11
]20 HOME
]30 READ A,B,C
]40 PRINT A,B,C
]50 END
```

Resultado deste programa:

```
51      62      74
```

Ao empregar a instrução READ, atente ao seguinte:

a) associe o nome-de-variável correto a cada tipo de dado (cadeia, real ou inteiro);

b) não utilize a instrução READ mais vezes do que o número de dados contidos na(s) instrução(s) DATA.

No exemplo abaixo, a instrução READ está relacionada a dado não contido em DATA:

```
]NEW
]10 DATA "GIRAFA","EMA","TATU","PACA"
]20 FOR I = 1 TO 10
]30 READ A$: PRINT A$
]40 NEXT I
```

Tal tipo de erro faz com que o TK //e apresente a mensagem:

```
?FALTA DADOS ERRO EM 30
```

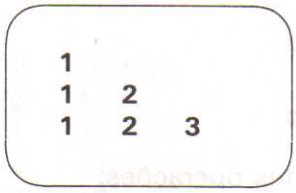
2. RESTORE

A instrução RESTORE faz com que os dados da instrução DATA voltem a ser lidos de maneira que possam ser novamente manipulados.

Observe o exemplo abaixo:

```
]NEW
]10 DATA 1,2,3
]30 READ A
]40 PRINT A
]50 RESTORE
]60 READ A,B
]70 PRINT A,B
]80 RESTORE
]90 READ A,B,C
]100 PRINT A,B,C
```

Resultado:



Cada vez que surge a instrução RESTORE, os dados de DATA voltam a ser lidos desde o início, pela próxima READ.

O programa a seguir apresenta as notas de quatro bimestres de uma determinada matéria da escola. Em seguida, aplicando os pesos de cada bimestre, tira a média anual desta matéria.

Suponhamos que as notas sejam 8, 7, 5 e 9; e os pesos dos bimestres; 1, 2, 3 e 4.

Pressione a tecla [mode], para trabalhar com o teclado em português.

```
NEW
ç10 REM NOTAS DOS BIMESTRES
ç20 DATA 8,7,5,9
ç30 REM PESOS DOS BIMESTRES
ç40 DATA 1,2,3,4 ç45 HOME
ç50 PRINT ''As notas são'':
ç60 READ A,B,C,D
ç70 PRINT ''Primeiro bimestre- '';A
ç80 PRINT ''Segundo bimestre- '';B
ç90 PRINT ''Terceiro bimestre- '';C
ç100 PRINT ''Quarto bimestre- '';D
ç110 PRINT PRINT
ç130 READ X,Y,Z,W
ç140 F= A * X + B * Y + C * Z + D * W
ç150 G= X + Y + Z + W
ç160 H = F / G
ç170 PRINT '' A média anual é: '';H
ç180 END
```

Confira o resultado deste programa:

```
As notas são:
Primeiro bimestre- 8
Segundo bimestre- 7
Terceiro bimestre- 5
Quarto bimestre- 9

A média anual é: 7.3
```

3. GET

A forma geral da instrução GET é:

GET nome-de-variável string

Esta instrução executa as seguintes operações:

a) interrompe o programa;

b) aguarda a digitação de uma e apenas uma tecla correspondente a um caractere;

c) associa o caractere digitado ao nome-de-variável contido na instrução;

d) prossegue o programa após a digitação do caractere.

O programa a seguir exemplifica o emprego da instrução GET:

```
çNEW
ç10 HOME
ç15 GET A$
ç20 PRINT ''Você pressionou a tecla '';A$
ç50 GOTO 15
```

Se, após o comando RUN, são pressionadas as teclas M,I,C,R, e O obtém-se uma tela semelhante à figura abaixo:

```
Você pressionou a tecla M
Você pressionou a tecla I
Você pressionou a tecla C
Você pressionou a tecla R
Você pressionou a tecla O
```

Esta instrução é muito útil quando usada com as instruções condicionais (ver capítulo XII).

Para interromper a execução deste programa, é necessário pressionar as teclas [control] e [reset] em conjunto. O programa entende [control] [C] como dados sendo introduzidos.

4. DEF FN

Esta instrução permite que se definam funções no TK //e, como demonstra o programa:

```
]NEW
]5 HOME
]10 PRINT ''X'',''FN A(X)'': PRINT
]20 DEF FN A(X) = X ^ 2 + 3 * X + 1
]30 FOR X = 0 TO 5
]40 PRINT X, FN A(X)
]50 NEXT X
```

O resultado do programa é do tipo:

```
X    FN A(X)
0    1
1    5
2    11
3    19
4    29
5    41
```

O programa aplica os valores de X = 0 até X = 5 à função FN A(X) = X ^ 2 + 3 * X + 1.

Vamos conferir os resultados:

```
X = 0 _____ 0 ^ 2 + 3 x 0 + 1 = 1
X = 1 _____ 1 ^ 2 + 3 x 1 + 1 = 5
X = 2 _____ 2 ^ 2 + 3 x 2 + 1 = 11
X = 3 _____ 3 ^ 2 + 3 x 3 + 1 = 19
X = 4 _____ 4 ^ 4 + 3 x 4 + 1 = 29
X = 5 _____ 5 ^ 2 + 3 x 5 + 1 = 41
```

O programa seguinte é uma extensão do programa anterior. O argumento da linha 60 FN A(I) é I, com o mesmo efeito de X em FN A(X):

```
]60 PRINT: PRINT
]70 PRINT "I", "FN A(I)": PRINT
]80 FOR I = 1 TO 3
]90 PRINT I, FN A(I) + 8
]100 NEXT I
```

Resultado:

```
X    FN A(X)

1    5
2    11
3    19
4    29
5    41

I    FN A(I)

1    13
2    19
3    27
```

IX - LIDANDO COM DESVIOS

Sempre que, num programa, a seqüência numérica de execução das linhas não é obedecida, diz-se que ocorreu um "desvio".

GOTO é uma instrução que executa desvios, como você já aprendeu no capítulo VII deste manual.

1. Os Desvios Condicionais

Desvios condicionais são desvios ou, de modo geral, operações do programa que somente ocorrem em situações predeterminadas.

Neste item, explicam-se as instruções que provocam este tipo de "desvio".

1.1. IF ... GOTO

Este conjunto de instruções faz com que o programa seja desviado para a linha indicada após GOTO, se a condição apresentada após IF (se) é verdadeira. Caso a condição após IF não seja verdadeira, o programa prossegue normalmente na próxima linha.

Exemplo:

```
]10 REM Usando GOTO e IF
]20 FOR x = 1 TO 5
]30 IF x = 3 GOTO 80: REM quando x = a 3 va para a linha 80
]40 PRINT "x = ";x
]50 NEXT x
]60 PRINT "Fim"
]70 STOP
]80 PRINT "Nao gosto do numero 3"
]90 GOTO 50: REM va para a linha 50
```

1.2. IF ... THEN

Este conjunto de instruções faz com que o programa execute determinada operação, se a condição apresentada após IF é verdadeira.

Exemplo:

```
]10 REM Usando IF e THEN
]20 FOR x = 1 TO 5
]30 IF x = 3 THEN PRINT "Nao gosto do numero 3"
]40 PRINT "x= ";x
]50 NEXT x
]60 PRINT "Fim."
]70 STOP
```

1.3. ON ... GOTO

A instrução ON...GOTO age de forma um pouco diferente das anteriores. Sua forma geral é:

ON número inteiro GOTO 1.º parâmetro, 2.º parâmetro, ...,n parâmetro

Cada parâmetro corresponde a um número de linha do programa. Assim, se o número inteiro após ON for 1, o programa será desviado para a linha de programa correspondente ao 1.º parâmetro; se for 2, o programa será desviado para o número de linha correspondente ao 2.º parâmetro e assim por diante.

Exemplo:

```
]10 REM Usando ON- GOTO
]20 FOR x = 1 TO 5
]30 ON x GOTO 100, 200, 300, 400, 500
]40 NEXT x
]50 PRINT "Fim."
]60 STOP
]100 PRINT "Gosto do numero";x: GOTO 40
]200 PRINT x;"E um numero par.": GOTO 40
]300 PRINT "Nao gosto do numero ";x: GOTO 40
]400 PRINT x;"Tambem e um numero par": GOTO 40
]500 PRINT x;"Este foi o ultimo desvio ": GOTO 50
```

1.4. IF ... THEN ... GOTO

Este conjunto de instruções combina a utilização de IF..THEN e IF...GOTO, como demonstra o programa a seguir:

```
]NEW
]10 REM divisao de numero inteiro por dois
]20 HOME
]30 REM obtencao do numero
]40 INPUT "Qual e o numero ? ";A
]50 REM verificacao e divisao
]60 IF A=INT (A) THEN C = A/2 : GOTO 100
]70 PRINT "O numero ";A;" nao e inteiro"
]80 GOTO 110
]100 PRINT "A metade de ";A;" é ";C
]110 PRINT:PRINT
]120 PRINT "Desejando dividir mais numeros, digite S"
]130 PRINT "Em caso contrario, digite outra tecla"
]140 GET D$
]150 PRINT:PRINT
]160 IF D$ = "S" THEN PRINT : GOTO 40
]170 PRINT: PRINT: PRINT "F","I","M"
]180 END
```

O programa utiliza duas instruções condicionais: uma para verificar se o número digitado é ou não inteiro e outra, para saber se é ou não desejado o prosseguimento do programa. Note que em cada caso, de acordo com o dado introduzido nas instruções condicionais, o programa toma uma atitude diferente.

A tela abaixo é apresentada se, após a ordem RUN, são digitados: o número 3.5, a letra S, o número 8 e a letra N:

```
Qual e o numero ? 3.5.
O Numero 3.5 nao e inteiro

Desejando dividir mais numeros digite S
Em caso contrario, digite outra tecla

Qual e o numero? 8
A metade de 8 e 4

Desejando dividir mais numeros, digite S
Em caso contrario, digite outra tecla

F                    I                    M
```

2. As Sub-rotinas

O uso de sub-rotinas é um artifício bastante aplicado em programação. Entende-se por sub-rotina um programa secundário que se pode usar diversas vezes dentro dum programa maior.

Por exemplo, um programa para resolução de problemas de física pode utilizar uma sub-rotina de conversão de unidades. Assim, sempre que for necessário converter uma determinada unidade (centímetros para metro, tonelada para quilos, hora para segundos etc.) basta que se chame a sub-rotina de conversão de unidades.

Além de facilitar o trabalho do programador - que não precisará escrever repetidas vezes um mesmo trecho do programa - as sub-rotinas economizam espaço na memória do computador.

2.1. GOSUB e RETURN

GOSUB é a instrução que desvia o programa para a sub-rotina desejada. Ao final da execução da sub-rotina, a instrução RETURN faz com que o programa volte à sua seqüência normal: uma instrução após GOSUB.

Observe o exemplo (não confunda a instrução RETURN com a tecla return):

O programa a seguir pode ser dividido em duas partes: o programa principal e a sub-rotina. As linhas de números 10 a 70 constituem o programa principal, e as linhas 100 a 150, a sub-rotina. A instrução GOSUB está no programa principal, enquanto o comando RETURN está na sub-rotina.

```
]NEW
]10 X=5 : Y=1
]20 GOSUB 100
]30 X=7 : Y=3
]40 GOSUB 100
]50 X=4 : Y=5
]60 GOSUB 100
]70 PRINT "FIM"
]80 STOP
]100 REM SUB-ROTINA
]110 FOR I = 1 TO X
]120 PRINT Y;
]130 NEXT I
]140 PRINT
]150 RETURN
```

programa principal

sub-rotina

Após a ordem RUN, obtém-se:

```
11111
3333333
5555
FIM
```

Examinemos agora a seqüência de execução do progama:

a) a linha 10 indica X = 5 e Y = 1;

b) na linha 20, GOSUB provoca o desvio para a sub-rotina da linha 100;

c) executa-se a sub-rotina e, na linha 150, RETURN faz com que o programa volte à linha 30, onde se atribui à variável X o valor 7, e à Y o valor 8;

d) as operações citadas são então repetidas para X = 7, Y = 3 e X = 4, Y = 5, e o programa é finalizado.

2.2. Sub-rotina dentro duma sub-rotina

Às vezes é necessário ter-se uma segunda sub-rotina dentro de uma sub-rotina.

Execute o programa:

```
]10 REM 'Pequena agenda telefonica' exemplo sub-rotinas
]15 HOME
]20 PRINT " --------- M E N U ---------"
]25 HTAB 5: VTAB 5: PRINT "1 - Inserir dados :"
]30 HTAB 5: PRINT "2 - Ler dados:"
]35 HTAB 5: PRINT "3 - Parar o programa"
]40 VTAB 21: INPUT "Escolha a opcao: ";O
]45 IF O > 3 OR O < 1 THEN VTAB 21: HTAB 17: PRINT "      ": GOTO 40
]50 ON O GOTO 70,300,400
]60 REM Primeira sub-rotina comeca aqui
]70 PRINT "Se houver, os dados serao apagados. ": PRINT "CONFIRMA <S
    OU N>": GET Q$: IF Q$ < > "S" THEN 15
]80 CLEAR : HOME
]100 VTAB 21: INPUT "Numero de pessoas a cadastrar ? ";Z
]110 DIM A$(Z,2)
]115 HOME
]120 FOR X = 1 TO Z
]125 PRINT "Entre o nome (maximo 20 caracteres)"
]130 INPUT " ";A$(X,1)
]131 IF LEN (A$(X,1)) > 20 THEN GOTO 350
]135 PRINT "Entre o telefone do ";A$(X,1)
]140 INPUT " ";A$(X,2)
]145 NEXT X
]150 REM Segunda sub-rotina
]300 HOME
]310 HTAB 1: VTAB 2: PRINT "Nomes:": HTAB 30: VTAB 2: PRINT "telefones:"
]315 PRINT : PRINT
]320 FOR X = 1 TO Z
]325 PRINT A$(X,1);: HTAB 30: PRINT A$(X,2)
]330 NEXT X
]340 PRINT : PRINT "Tecle algo para voltar ao menu ": GET G$
]345 GOTO 15
]350 PRINT "Nome muito grande": FOR tempo = 1 TO 1500: NEXT tempo:
    HOME: GOTO 125
]360 REM A TERCEIRA SUB-ROTINA COMECA AQUI
]400 HOME
]410 PRINT "Para voltar a rodar o programa, use: RUN "
]415 PRINT "Porem os dados serao apagados."
]420 PRINT "GOTO 15 nao apagara os dados"
]430 STOP
```

1.ª Sub-Rotina

2.ª Sub-Rotina

3.ª Sub-Rotina

2.3. ON ... GOSUB

O emprego de ON ... GOSUB é bastante semelhante ao de ON ... GOTO (item 1.3), mas, em vez de desviar o programa para determinadas linhas, ON...GOSUB desvia-o para determinada sub-rotina. Você não deve se esquecer de finalizar cada uma das sub-rotinas com a instrução RETURN.

Verifique os dois exemplos a seguir, o primeiro utilizando ON...GOTO e o segundo, ON...GOSUB.

Primeiro exemplo:

```
]NEW
]10 FOR I = 1 TO 100
]20 B% = RND(1) * 3 + 1
]30 ON B% GOTO 100,200,300
]100 A(1) = A(1) + 1
]110 GOTO 400
]200 A(2) = A(2) + 1
]210 GOTO 400
]300 A(3) = A(3) + 1
]400 NEXT I
]410 FOR I = 1 TO 3
]420 PRINT "A(";I;") = ";A(I)
]430 NEXT I
]440 END
```

Segundo exemplo:

```
]NEW
]10 FOR I = 1 TO 100
]20 B% = RND(1) * 3 + 1
]30 ON B% GOSUB 100,200,300
]40 NEXT I
]60 FOR I = 1 TO 3
]70 PRINT "A(";I;") = ";A(I)
]80 NEXT I
]90 END
]100 A(1) = A(1) + 1
]120 RETURN
]200 A(2) = A(2) + 1
]220 RETURN
]300 A(3) = A(3) + 1
]320 RETURN
```

Neste programa, determina-se randomicamente I = 1, 2 ou 3 por meio de RND. Se I = 1, então GOSUB é 100; para I = 2, GOSUB é 200; e para I = 3 GOSUB é 300. Isto é feito 100 vezes, e então se apresenta a estatística do número de vezes que cada valor foi apresentado.

2.4. ONERR GOTO

Se ocorre um erro durante a execução dum programa, o computador interrompe a execução e apresenta uma mensagem de erro.

Por exemplo:

```
]NEW
]10 HOME
]20 FOR I = 0 TO 10
]30 PRINT I, 5/I
]40 NEXT I
]RUN
0
?DIVISAO POR ZERO ERRO EM 30
```

Iniciada a execução do programa, o computador interrompe-a imediatamente, porque, sendo I = Ø, não é possível realizar 5/Ø.

ONERR GOTO é a abreviação de "ON ERROR GOTO" (em caso de erro, vá para). Esta instrução pode evitar que a execução do programa seja interrompida. ONERR GOTO provoca o desvio para uma rotina de tratamento adequado de erro. Tal rotina pode ser finalizada com o comando RESUME, que retorna o processamento ao ponto onde surgiu o erro.

Observe o exemplo:

```
]NEW
]10 HOME
]20 ONERR GOTO 100
]30 INPUT "Digite o numero " ;X
]40 Y = 1 / X
]50 PRINT "X = ";X,"Y = ";Y
]60 GOTO 30
]100 PRINT "Nao vale zero"
]105 IF PEEK (222) = 131 THEN 30
]110 RESUME
```

Se após dar a ordem RUN, você digita os números 56, 12 e Ø, obtém uma tela assim:

```
Digite o numero 56
X =56      Y = .0178571429
Digite o numero 12
X =12      Y = .0833333333
Digite o numero Ø
Nao vale zero
Digite o numero ■
```

Num programa extenso, há a probabilidade da ocorrência de vários erros diferentes. Através da leitura de PEEK(222), pode-se determinar o código do erro cometido no programa. A tabela que apresenta os códigos de todos os erros está no Apêndice B.

Se, por exemplo, na linha 107 tivéssemos digitado:

```
]107 A = PEEK (222) : PRINT A
```

o programa, além do resultado anterior, na ocasião apresentaria o número 131, que corresponde ao erro DIVISAO POR ZERO.

Para desativar o modo ONERR, deve-se executar o comando POKE 216,0.

2.5. POP

A instrução POP faz com que a próxima instrução RETURN desvie o programa para a ordem seguinte à penúltima instrução GOSUB. O efeito de POP é o de anular a última chamada de GOSUB, fazendo com que o programa volte para o GOSUB imediatamente anterior.

No exemplo seguinte, a instrução POP não é usada.

```
]NEW
]10 PRINT "A"
]20 GOSUB 100
]30 PRINT "E"
]40 END
]100 PRINT "B"
]110 GOSUB 200
]120 PRINT "D"
]130 RETURN
]200 PRINT "C"
]210 RETURN
```

Após RUN, dá-se a apresentação da seqüência A, B, C, D e E (uma letra por linha). A ordem de execução das linhas de comando pode ser melhor percebida se é acionando o comando TRACE. Obtêm-se os números das linhas, pela ordem de execução: #10 #20 #100 #110 #200 #210 #120 #130 #30 #40.

Se a linha "205 POP" é incluída, a letra D deixa de ser apresentada. A nova ordem de execução das linhas de comando passa a ser: #10 #20 #100 #110 #200 #205 #210 #30 #40.

Depois da inclusão de POP na linha 205, a instrução RETURN da linha 210 não desvia o programa para a linha 120, mas sim para a linha 30. POP anula a última situação de retorno, e a anterior passa a constar.

X - FUNÇÕES MATEMÁTICAS

Além das operações algébricas básicas tais como: +, -, *, /, ^, seu computador resolve funções matemáticas. Neste capítulo estudamos as funções existentes na linguagem do TK //e. Algumas dessas funções já lhe foram apresentadas: INT(X), RND(1) e SQR(X).

1. As Funções Trigonométricas

As funções trigonométricas que o TK //e calcula são as principais: seno, cosseno e tangente.

Na linguagem BASIC utilizada pelo TK //e, as operações com as funções trigonométricas são baseadas na unidade radiano. Isto é, o valor entre parênteses deve ser expresso em radianos, e não em graus. Lembre que 360 graus correspondem, aproximadamente, a 6.28318531 (2PI) radianos e que, portanto, cada radiano vale perto de 57.2957795 graus.

1.1. Função seno - SIN(X)

Digite o programa a seguir, para formar uma tabela do seno de X:

```
]NEW
]10 PRINT "GRAUS";TAB(8);"RADIANOS";TAB(24);"SEN(X)"
]20 PRINT
]30 R = 360 / 6.28318531
]40 FOR I = 0 TO 360 STEP 30
]50 Y = SIN(I / R)
]60 PRINT I;TAB(8);I / R;TAB(24);Y
]70 NEXT I
```

Observe que primeiramente tomamos a linha número 30 para obter o fator de conversão de graus em radiano. Na linha 50 os graus são convertidos em radianos (I/R). A conversão foi feita para cada 15 graus.

Depois da ordem RUN, obtém-se a seguinte tabela:

GRAUS	RADIANOS	SEN(X)
0	0	0
30	.523598776	.5
60	1.04719755	.866025404
90	1.57079633	1
120	2.0943951	.866025404
150	2.61799388	.5
180	3.14159266	0
210	3.66519143	-.500000001
240	4.1887902	-.866025403
270	4.71238898	-1
300	5.23598776	-.866025403
330	5.75958653	-.5
360	6.28318531	0

1.2. Função cosseno - COS(X)

Agora faça seu computador executar uma tabela do cosseno de X, de forma semelhante à realizada no subitem anterior:

```
]NEW
]10 PRINT "GRAUS";TAB(8);"RADIANOS";TAB(24);"COS(X)"
]20 PRINT
]30 R = 360 / 6.28318531
]40 FOR I = 0 TO 360 STEP 30
]50 Y = COS(I / R)
]60 PRINT I; TAB(8);I/R; TAB(24);Y
]70 NEXT I
```

Resultado:

GRAUS	RADIANOS	COS(X)
0	0	1
30	.523598776	.866025404
60	1.04719755	.5
90	1.57079633	0
120	2.0943951	-.5
150	2.61799388	-.866025403
180	3.14159266	-1
210	3.66519143	-.866025403
240	4.1887902	-.5
270	4.71238898	0
300	5.23598776	.500000001
330	5.75958653	.866025405
360	6.28318531	1

1.3. Função tangente - TAN(X)

Finalmente calcule os valores da tangente de X de maneira similar ao cálculo do COS(X) e do SIN(X):

```
]NEW
]10 PRINT "GRAUS";TAB(8);"RADIANOS";TAB(24);"TAN(X)"
]20 PRINT
]30 R = 360 / 6.28318531
]40 FOR I = 0 TO 360 STEP 30
]45 IF (I = 90) OR(I = 270)THEN PRINT I;TAB(8);I/ R;TAB(24);"INFINITO": NEXT I
]50 Y = TAN (I/R)
]60 PRINT I;TAB(8);I/R;TAB(24);Y
]70 NEXT I
```

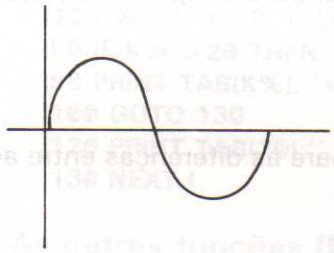
Resultado:

GRAUS	RADIANOS	TAN(X)
0	0	0
30	.523598776	.577350269
60	1.04719755	1.73205081
90	1.57079633	INFINITO
120	2.0943951	-1.73205081
150	2.61799388	-.577350268
180	3.14159266	0
210	3.66519143	.57735027
240	4.1887902	1.7320508
270	4.71238898	INFINITO
300	5.23598776	-1.7320508
330	5.75958653	-.577350268
360	6.28318531	0

2. Os Traçados Gráficos de Funções

2.1. Diagrama simples

O TK //e é capaz de calcular valores numéricos e de traçar gráficos de funções. Sem lançar mão dos recursos de alta-resolução, pode-se traçar gráficos de funções, usando a função TAB. Para isso, é preciso girar o eixo das coordenadas em 90 graus, no sentido horário, conforme mostra a figura abaixo:



a) sentido normal de execução do gráfico

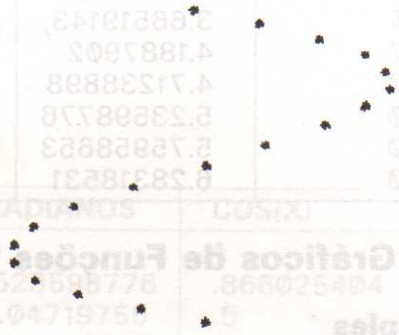
b) rotação do eixo do gráfico em 90 graus no sentido horário

O número de caracteres é limitado pela tela. Sabendo-se que o valor da função seno varia entre 1 e -1, e que a tela do TK //e tem somente 40 caracteres para cada linha, deve-se usar 20 como ponto médio.

Execute o programa a seguir. O ponto chave deste programa situa-se nas linhas 20 e 30.

```
]NEW
]10 FOR I = 0 TO 360 STEP 10
]20 Y = SIN (I/57.2957795)
]30 K% = Y * 19 + 20
]40 PRINT TAB (K%);"*"
]50 NEXT I
```

Resultado:



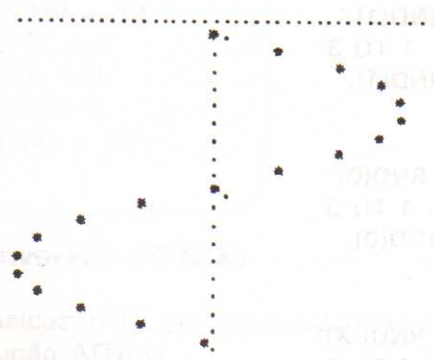
Na linha 20, Y é o valor de seno, sendo seus valores entre -1 e 1. Quando Y é multiplicado por 19, seus valores passam a variar entre -19 e 19. Então, soma-se 20 a ele, de modo que o valor de K% é obtido entre 1 e 39, passando a ocupar os limites da tela. A linha 40 apresenta o símbolo "*" em TAB(K%).

2.2. Gráfico da função seno

Depois de observar o programa do subitem anterior, compare as diferenças entre aquele e este:

```
]NEW
]10 FOR I = 0 TO 39
]20 PRINT ".";
]30 NEXT I
]40 PRINT
]50 FOR I = 0 TO 360 STEP 10
]60 Y = SIN(I/57.2957795)
]70 K%= Y * 19 + 20
]80 IF K% >20 THEN 120
]90 PRINT TAB(K%);"*";TAB(20);"."
]100 GOTO 130
]120 PRINT TAB(20);".";TAB(K%);"*"
]130 NEXT I
```

A figura a seguir ilustra o resultado deste programa:



2.3. Gráfico da função cosseno

Verifique o programa abaixo e compare seu resultado com o do programa apresentado no subitem anterior (obs.: se você tiver digitado o programa anterior, não digite NEW, basta editar a linha 60, alterada):

```
]10 FOR I = 0 TO 39
]20 PRINT ".";
]30 NEXT I
]40 PRINT
]50 FOR I = 0 TO 360 STEP 10
]60 Y = COS(I/57.2957795)
]70 K%= Y * 19 + 20
]80 IF K% >20 THEN 120
]90 PRINT TAB(K%);"*";TAB(20);"."
]100 GOTO 130
]120 PRINT TAB(20);".";TAB(K%);"*"
]130 NEXT I
```

2.4. As outras funções (RND ABS ATN SGN EXP LOG SQR)

2.4.1. Função randômica - RND(X)

Há três tipos de funções randômicas no TK //e: RND(1), RND(0) e RND(-X).

Siga o exemplo:

```
]NEW
]5 PRINT "RND(1)"
]10 FOR I = 1 TO 3
]20 PRINT RND(1),
]30 NEXT I
]40 PRINT
]45 PRINT "RND(0)"
]50 FOR I = 1 TO 3
]60 PRINT RND(0),
]70 NEXT I
]80 PRINT
]85 PRINT "RND(-X)"
]90 FOR I = 1 TO 3
]100 PRINT RND(-I),
]110 NEXT I
]120 END
```

Após a ordem RUN, a tela apresenta:

```
RND(1)
.235586735          .186784665
.37278107
RND(0)
.37278107           .37278107
.37278107
RND(-X)
2.199196472E-08     2.99205567E-08
4.48217179E-08
```

Obs.: Se você não obteve valores iguais aos deste exemplo, não se preocupe: os valores RND (1) são aleatórios.

Partindo dos resultados obtidos, observe que:

a) RND(1) fornece um número entre 0 e 0.999999999;

b) RND(0) repete o número fornecido pela instrução RND anterior;

c) cada RND(-X) seleciona um número fixo e de valor bastante pequeno.

2.4.2. Função valor absoluto - ABS(X)

ABS(X) faz com que sejam obtidos valores absolutos.

Verifique o programa a seguir:

```
]NEW
]10 FOR I = -10 TO 10 STEP 5
]20 PRINT "ABS(";I;") = ";ABS(I)
]30 NEXT I
```

Resultado:

```
ABS(-10) = 10
ABS(-5) = 5
ABS(0) = 0
ABS(5) = 5
ABS(10) = 10
```

2.4.3. Função trigonométrica inversa - ATN(X)

Além das funções trigonométricas básicas, o TK //e efetua a função inversa da tangente de X. Para tanto, emprega-se a instrução ATN(X).

Observe o exemplo e seu resultado:

```
]NEW
]10 FOR I = -10 TO 10 STEP 5
]20 PRINT "ATN(";I;") = ";ATN(I)
]30 NEXT I
]40 END
```

Resultado:

```
ATN(-10) =
-1.47112768
ATN(-5) = -1.37340077
ATN(0) = 0
ATN(5) = 1.37340077
ATN(10) = 1.47112768
```

Outras funções trigonométricas podem ser calculadas através das fórmulas incluídas no Apêndice A.

2.4.4. Reconhecimento do sinal - SGN(X)

O programa a seguir é um exemplo de SGN(X). Se o valor de X é negativo, o computador apresenta na tela -1; se o valor é 0, apresenta 0; e se o valor é positivo, apresenta 1:

```
]NEW
]10 FOR I = -10 TO 10 STEP 5
]20 PRINT "SGN(";I;") = "; SGN(I)
]30 NEXT I
```

Resultado:

```
SGN(-10) = -1
SGN(-5) = -1
SGN(0) = 0
SGN(5) = 1
SGN(10) = 1
```

2.4.5. Função exponencial - EXP(X)

A aplicação da matemática em áreas como as da Engenharia e da Física requer a utilização de expoentes naturais, que têm **e** (aprox. 2.718) como base . Dado que **e** elevado a X é igual a Y, Y = e ^ X. Na linguagem BASIC do TK //e, pode-se expressar isto por: Y = EXP(X).

Execute o programa a seguir e confira o resultado:

```
]NEW
]10 FOR I = 1 TO 5
]20 PRINT "EXP(";I;") = ";EXP(I)
]30 NEXT I
```

Resultado:

```
EXP(1) = 2.71828183
EXP(2) = 7.3890561
EXP(3) = 20.0855369
EXP(4) = 54.5981501
EXP(5) = 148.413159
```

2.4.6. Função logarítmica - LOG(X)

A função logarítmica pode ser obtida no TK //e através da instrução LOG(X).

Veja o exemplo:

```
]NEW
]10 FOR I = 1 TO 5
]20 PRINT "LOG(";I;") = ";LOG(I)
]30 NEXT I
```

Após o comando RUN, tem-se:

```
LOG(1) = 0
LOG(2) = .693147181
LOG(3) = 1.09861229
LOG(4) = 1.38629436
LOG(5) = 1.60943791
```

2.4.7. Raiz quadrada - SQR(X)

Esta função já foi apresentada no item 2 do capítulo II, e tem por finalidade extrair a raiz quadrada de um número.

Execute o programa a seguir e observe os resultados:

```
]NEW
]10 HOME
]20 INPUT "QUAL O NUMERO ";A
]30 PRINT "A RAIZ QUADRADA DE ";A;" E ";SQR(A)
]40 FOR I = 1 TO 500 : NEXT I
]50 GOTO 20
```

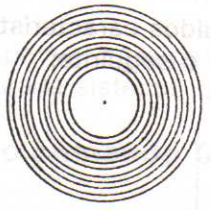
XI - ARMAZENAMENTO EM DISQUETE

À transferência de dados da memória do computador para a unidade de disco são atribuídos diversos nomes. Como a memória RAM é volátil, costuma-se dizer que os dados nela contidos podem ser "perdidos", se não forem "salvos" num dispositivo auxiliar de memória mais "consistente", como, por exemplo, o disquete. O recurso de "salvar" (em inglês, SAVE) as informações em disquetes permite que elas sejam preservadas, mesmo quando se desliga o computador.

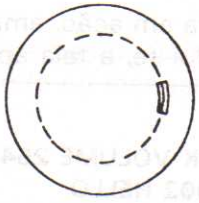
1. Salvando Programas em Disquete

1.1. Inicializando o disquete (INIT)

Antes que um disquete seja utilizado, deve ser inicializado, para ser controlado por um sistema operacional. Durante o processo de formatação, o sistema cria trilhas magnéticas concêntricas, sobre a face do disquete, e divide-as em setores. Se o sistema utilizado é o TKDOS, são criadas 35 trilhas, que se subdividem em 16 setores. Em cada setor cabem 256 bytes.



35 TRILHAS
POR DISCO



16 SETORES
POR TRILHA

256 BYTES
POR SETOR

Levando-se em conta que cada sistema operacional formata o disquete de um modo bastante particular, é de esperar que um disquete formatado por um determinado sistema não possa ser utilizado por um sistema diferente.

Alguns setores do disquete contêm informações acerca de seu conteúdo. Estas formam uma espécie de índice, geralmente denominado catálogo ou diretório. Além do número do volume do disquete, o diretório contém nomes de arquivos, espaço que ocupam, etc.

Para formatar um disquete, proceda da seguinte maneira:

Notas: 1) As operações indicadas devem ser efetuadas com o modo CAPS LOCK ativado.
2) Certifique-se de que o disquete a ser inicializado não contém dados importantes, pois o processo de formatação destrói o conteúdo dos disquetes.

a) Introduza na unidade de disco um disquete contendo o sistema operacional TKDOS.

b) Dê uma "partida quente" no computador (consulte o capítulo IV, subitem 6.4.1). Isto fará com que o TK //e carregue o sistema contido no disquete.

c) Quando o sinal de prompt (]) aparecer, substitua o disquete do TKDOS pelo disquete a ser formatado.

d) Digite o comando INIT HELLO e tecle return.

O "led" da unidade de disco acende-se, sinalizando que o periférico foi acionado. Normalmente a unidade de disco emite ruídos característicos enquanto formata o disquete.

O reaparecimento do sinal de prompt na tela indica que a operação está concluída. O disquete pode agora ser utilizado para gravar quaisquer dados, no formato do sistema TKDOS.

1.2. Examinando o conteúdo do diretório (CATALOG)

O bloco de dados gravados num disquete é chamado de arquivo. Para obter a apresentação visual de um catálogo, proceda da seguinte forma:

a) Certifique-se de que o disquete recém-inicializado está dentro da unidade de disco, e que esta está bem fechada.

b) Digite CATALOG e tecle return.

Novamente a unidade de disco entra em ação, emitindo ruídos característicos de leitura. Quando a leitura do diretório conclui-se, a tela apresenta o seguinte:

```
DISK VOLUME 254
A 002 HELLO
```

O sistema TKDOS encarrega-se de determinar o número de volume 254. Este procedimento é rotineiro quando o valor não é previamente especificado.

A segunda linha do catálogo informa que existe no disquete um arquivo de formato A, ocupando dois setores (002), denominado HELLO.

1.3. Introduzindo um programa de abertura

Embora o diretório acuse sua presença, o arquivo HELLO está vazio, visto que não havia nenhum programa na memória, no momento da inicialização do disquete.

Tente inicializar novamente o disquete, digitando antes o seguinte programa:

```
]10 HOME
]20 HTAB 7: VTAB 10
]30 PRINT "Programa de Apresentação"
]40 PRINT: PRINT TAB(5);"Criado por (seu nome) em (data atual)"
```

Liste-o para ver se não existe algum erro de digitação. Se tudo estiver correto, digite:

INIT HELLO

Tecle [return] e aguarde que o disquete seja novamente formatado.

Quando o "led" da unidade de disco apagar-se, digite CATALOG e certifique-se de que o programa HELLO está relacionado no diretório do disquete recém-formatado.

Depois de confirmar que o disquete foi formatado, dê uma "partida fria", mantendo na unidade de disco o disquete inicializado. Desta vez, surge no vídeo a seguinte mensagem:

Programa de Apresentação

Criado por (nome) em (data)

Ao consultar o diretório, por meio do comando CATALOG, observe que o programa HELLO continua constando na listagem, da mesma forma que antes.

Em ambos os exemplos apresentados, o sistema TKDOS foi gravado no disquete. Assim sendo, todos os comandos pertinentes a este sistema estão disponíveis quando o disquete é acessado pelo computador.

O TK //e é compatível com diversos sistemas operacionais. Procure obter informações precisas sobre cada sistema, consultando os manuais específicos.

1.4. Usando o comando SAVE

Para gravar programas num disquete, usa-se o comando SAVE, no seguinte formato:

SAVE nome do programa

Digite o seguinte programa a ser gravado:

```
] NEW
]5 REM PROGRAMA TESTE
]10 HOME : PRINT "PROGRAMA TESTE"
]20 PRINT
]30 PRINT "FIM"
```

Insira o disquete-arquivo (formatado) na unidade de disco. A seguir, digite SAVE TESTE e tecle [return].

Imediatamente se aciona a unidade de disco, operando-se a gravação do programa. Ao término deste processo, o led do periférico apaga-se.

Nota: Aconselha-se anotar na etiqueta do disquete o nome com o qual cada um dos programas foi gravado. Tal nome deverá ser informado sempre que você quiser carregar o programa no computador.

2. Carregando um programa gravado em disquete (LOAD)

Para carregar programas gravados em disquete, usa-se o comando LOAD, no seguinte formato:

LOAD nome do programa

Experimente carregar o programa TESTE (subitem 1.4). Digite NEW, para limpar a memória do TK //e, e em seguida: LOAD TESTE return.

A unidade de disco entra em ação. Terminada a leitura, o led deste periférico apaga-se.

Advertência: O nome do programa deve ser digitado exatamente como foi gravado. De outra forma, não será reconhecido, e não se operará a carga.

XII - ARMAZENAMENTO EM FITA CASSETE

Utilizando um gravador cassete comum, pode-se armazenar dados em fita magnética.

Pode-se recorrer ao gravador quando o sistema não possui uma unidade de disco. Contudo, o gravador não substitui a unidade de disco, dado que o acesso à fita magnética não é direto, mas seqüencial.

Nota: A utilização de um gravador implica em certas restrições inerentes às características do processo de gravação:

a) O TKDOS não pode ser carregado (é imprescindível o uso de uma unidade de disco).

b) Só podem ser armazenados programas, blocos de memória, tabelas de figuras e matrizes numéricas.

1. Gravando Programas em Fita Magnética (SAVE)

Para salvar programas ou dados da memória do TK //e, você deve proceder da seguinte maneira:

a) Conecte um dos plugues do cabo de leitura/gravação na tomada CASSETE do computador, introduzindo o outro plugue na tomada MIC do gravador.

b) Coloque no gravador uma fita magnética nova, de boa qualidade, bobinando-a de maneira que seu campo magnético (e não a parte plástica) fique exposto ao cabeçote de gravação do cassete.

c) Digite o comando SAVE e acione o modo de gravação do gravador (geralmente pressionando as teclas RECORD e PLAY em conjunto). Tecle return.

Este procedimento resulta na emissão de um sinal sonoro (bip), indicando que a operação teve início. Durante a gravação, o cursor desaparece da tela e retorna após um outro "bip", que sinaliza o final da operação.

Ao término da gravação, desligue o gravador e rebobine a fita até o começo, para deixá-la preparada para uso.

O processo de gravação não interfere nos dados retidos na memória do computador. Agora eles encontram-se na memória e na fita cassete.

2. Carregando Programas de uma Fita (LOAD)

A transferência de programas ou dados de uma fita magnética para um computador recebe várias denominações. Costuma-se dizer que o computador "lê" os dados da fita ou "carrega-os" na memória.

Para que o computador leia os dados de uma fita cassete pré-gravada, carregando suas informações na memória, proceda conforme as indicações a seguir:

a) Coloque a fita no gravador e bobine-a até a posição onde estão os dados a serem lidos pelo computador.

b) Instale o cabo de leitura/gravação de maneira que um de seus plugues se encaixe na tomada CASSETE do computador e o outro, na tomada EAR do gravador. Ajuste o volume do gravador em aproximadamente 3/4 de sua potência.

c) Digite o comando LOAD e pressione return.

d) Aperte a tecla PLAY do gravador.

O cursor desaparece, e um "bip" é emitido. Quando o cursor retornar à tela, e um outro sinal sonoro for emitido, pare o gravador e verifique se o programa foi carregado integralmente.

Se, após a leitura, você não obtiver o resultado esperado, leia a fita diversas vezes, regulando o volume do gravador até encontrar a potência adequada ao TK //e.

É provável que, durante as tentativas de leitura, sejam exibidas algumas mensagens do sistema. Elas são comuns neste tipo de procedimento.

Procure anotar o volume em que se completa com êxito a leitura da fita.

3. Gravando e Recuperando Dados Numéricos

O TK //e possui dois comandos específicos para gravação e leitura de dados numéricos em fita magnética: os comandos STORE e RECALL.

3.1. STORE

O comando STORE permite que uma matriz numérica seja gravada em fita cassete para posterior leitura. O programa em BASIC que fizer uso deste comando deve conter uma matriz dimensionada exclusivamente para este fim.

Por exemplo:

```
]5 HOME
]10 DIM A(3,3,3)
]20 FOR K=0 TO 3: FOR J=0 TO 3: FOR I=0 TO 3
]30 A=INT(10*RND(1))
]40 A(I,J,K)=A
]45 PRINT "A(";I;",";J;",";K;")= ";A;
]50 NEXT: NEXT: NEXT
]55 PRINT
]60 PRINT "Digite qualquer tecla quando o gravador estiver pronto"
]70 GET A$
]80 STORE A
]90 PRINT "Operacao Terminada"
]100 END
```

O gravador deve estar em RECORD quando o programa executar a linha que contém a instrução STORE. Ao término da operação, a próxima linha é executada.

3.2. RECALL

Para que uma matriz seja recuperada, deve-se usar o comando RECALL. Este comando coloca os dados numéricos gravados em fita numa matriz dimensionada num programa em BASIC.

Para recuperar os dados gravados do exemplo anterior, digite o seguinte programa:

```
]10 DIM B(3,3,3)
]20 PRINT "Digite qualquer tecla quando o gravador estiver pronto"
]25 GET A$
]30 RECALL B
]35 PRINT "Operacao Terminada"
]40 FOR K=0 TO 3: FOR J=0 TO 3: FOR I=0 TO 3
]45 PRINT "B(";I;",";J;",";K;")= ";B(I,J,K);
]50 NEXT: NEXT: NEXT
]55 END
```

Embora tenhamos dado um nome diferente à matriz que vai receber os dados gravados, as dimensões são as mesmas. No comando RECALL, o TK //e admite apenas matrizes dimensionadas da mesma forma que a usada em STORE ou, no máximo, com a última dimensão maior.

Exemplos:

a) para gravar:

```
10 DIM A(3,3,3)
20 .....
.....
100 STORE A
110 END
```

b) para recuperar:

```
10 DIM B(3,3,5)
20 RECALL A
.....
.....
110 END
```

Neste exemplo, a última dimensão da matriz B é maior que a da matriz A. Os elementos adicionais são preenchidos com zeros.

XIII - GRÁFICOS E FIGURAS

1. Traçados de Baixa-Resolução

1.1. COLOR

A instrução COLOR permite a definição de cores na tela da TV (se o TK //e está ligado a um televisor em branco e preto, esta instrução gera tonalidades de cinza).

O TK //e permite que se usem até dezesseis diferentes cores num mesmo gráfico.

A tabela a seguir indica os números que se relacionam às diferentes cores:

Ø - preto	6 - azul médio	11 - rosa
1 - magenta	7 - azul claro	12 - verde claro
2 - azul escuro	8 - marrom	13 - amarelo
3 - púrpura	9 - laranja	14 - ciano (azul piscina)
4 - verde escuro	10 - cinza 2	15 - branco
5 - cinza 1		

Nota: Dependendo da TV ou monitor de vídeo utilizado, a tonalidade das cores pode variar.

A instrução COLOR é, necessariamente, usada com a instrução PLOT, como veremos no próximo subitem.

1.2. GR - PLOT - TEXT

O modo gráfico é iniciado pela instrução GR e é desativado pelo comando TEXT.

Usar o TK //e para "plotar" (traçar gráficos) significa definir as coordenadas horizontal e vertical e, então, colocar um ponto nesta posição. O TK //e divide a tela em 48 colunas (coordenada vertical) e 40 linhas (coordenada horizontal). As coordenadas têm valores de Ø a 39, na horizontal, e de Ø a 47, na vertical.

A forma básica da instrução PLOT é:

```
PLOT X,Y
```

Onde X corresponde à coordenada horizontal, e Y, à vertical.

Embora possam ser definidos pontos isolados na tela, tal procedimento não é usual, pois definir ponto a ponto uma figura é extremamente cansativo.

Os programas a seguir ilustram o uso de GR, COLOR e de PLOT, fornecendo, respectivamente, o traçado de uma reta vertical azul e o de uma horizontal verde.

1.º)
```
]NEW
]10 HOME
]20 GR
]30 COLOR = 6
]40 FOR I = 0 TO 39
]50 PLOT 20,I
]60 NEXT I
```

Nota: GR limpa a tela de baixa-resolução gráfica (como o comando HOME na tela de texto).

Caso não surjam as cores, procure ajustar a sintonia fina do seletor de canais.

2.º)
```
]NEW
]10 HOME
]20 GR
]30 COLOR = 4
]40 FOR I = 0 TO 39
]50 PLOT I,20
]60 NEXT I
```

1.3. HLIN - VLIN

HLIN e VLIN indicam, respectivamente, linha horizontal e linha vertical.

Através de HVIN, obtêm-se, com menor número de instruções, os mesmos resultados do primeiro programa do subitem anterior:

```
]TEXT
]NEW
]10 HOME
]20 GR
]30 COLOR = 6
]40 VLIN 0,39 AT 20
```

De forma semelhante, o segundo programa do subitem 1.2 pode ser refeito através das instruções:

```
]TEXT
]NEW
]10 HOME
]20 GR
]30 COLOR = 4
]40 HLIN 0,39 AT 20
```

1.4. SCRN

A instrução SCRN verifica, num determinado ponto da tela, o número da cor. Vamos utilizar esta instrução no último exemplo do subitem anterior e verificar o resultado:

```
]TEXT
]NEW
]10 HOME
]20 GR
]30 COLOR = 4
]40 HLIN 0,39 AT 20
]50 PRINT SCRN (20,20)
]60 PRINT SCRN (8,7)
```

No programa acima, após a ordem RUN, além do traçado da linha, a tela apresenta os números 4 e 0, que são correspondentes à cor verde e preta, respectivamente. O ponto (20,20) da tela está definido como verde, e o ponto (8,7), não definido, assume a cor do fundo, no caso, a preta.

2. Os Traçados de Alta-Resolução

A alta-resolução refere-se a traçados com maior número de detalhes e melhor definição. A capacidade de alta-resolução no TK //e possibilita a definição de 53.760 pontos.

Em baixa-resolução, usa-se o método de blocos, isto é, para cada coordenada definida é apresentado um bloco. Em alta-resolução, emprega-se o método de pontos.

2.1. HCOLOR

HCOLOR é usada no modo de alta-resolução da mesma forma que COLOR, no modo de baixa resolução. Neste modo, o número de cores disponíveis restringe-se a oito.

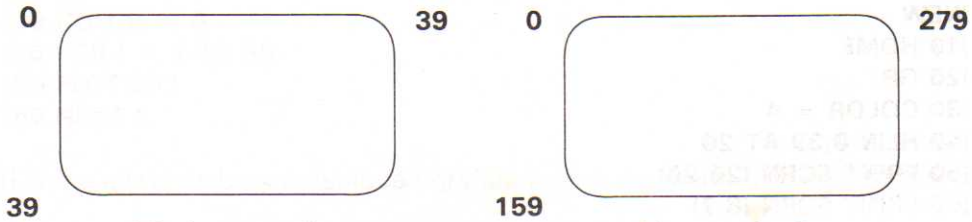
As cores e seus códigos são:

0 - preto 1	4 - preto 2
1 - verde	5 - laranja
2 - magenta	6 - azul
3 - branco 1	7 - branco 2

2.2. HPLOT, HGR e TEXT

A instrução HGR faz com que o TK //e entre no modo gráfico de alta-resolução.

A figura a seguir indica as coordenadas da tela em modo normal e em alta resolução:



Modo normal, com 4 linhas de texto

Alta-resolução, com 4 linhas de texto

Nota: Para voltar ao modo texto, você deve digitar o comando TEXT.

De maneira similar a PLOT, HPLOT executa o traçado, mas em alta-resolução.

Seguem exemplos do uso das instruções HCOLOR, HPLOT e HGR:

1.º) Exemplo (definição de quatro pontos, um em cada canto da tela):

```
]TEXT
]NEW
]10 HGR
]20 HOME
]30 HCOLOR = 7
]40 HPLOT 277,1
]50 HPLOT 1,1
]60 HPLOT 1,158
]70 HPLOT 277,158
```

Note que os pontos apresentados são pouco visíveis.

Digite agora o comando TEXT e siga ao próximo exemplo:

2.º) Exemplo (traçado de uma reta horizontal na parte superior da tela):

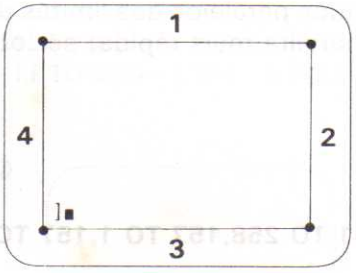
```
]TEXT
]10 HGR
]20 HOME
]30 HCOLOR = 7
]40 FOR I = 1 TO 279
]50 HPLOT I,1
]60 NEXT I
```

3.º) Exemplo (é executada uma linha vertical próxima ao lado esquerdo da tela):

```
]TEXT
]NEW
]10 HGR
]20 HOME
]30 HCOLOR = 7
]40 FOR I = 1 TO 191
]50 HPLOT 1,I
]60 NEXT I
```

4.º) Exemplo (o programa a seguir desenha um retângulo de lados paralelos aos limites da tela - leia as observações):

```
]TEXT
]NEW
]10 HGR: HCOLOR = 3
]20 FOR I = 1 TO 278
]30 HPLOT I,1: REM Lado superior
]40 NEXT I
]50 FOR I = 1 TO 190
]60 HPLOT 278,I: REM Lado direito
]70 NEXT I
]80 FOR I = 278 TO 1 STEP-1
]90 HPLOT I,190: REM Lado inferior
]100 NEXT I
]110 FOR I = 190 TO 1 STEP -1
]120 HPLOT 1,I: REM Lado esquerdo
]130 NEXT I
```



5.º) Exemplo (os valores definidos excedem os limites da tela; o TK //e emite uma mensagem de erro):

```
]TEXT
]10 HGR: HCOLOR = 3
]20 HOME
]30 FOR I = 0 TO 300
]40 HPLOT I,0
]50 NEXT I
```

Após o comando RUN, surge na tela uma reta, e soa um "beep". Digite control reset. O computador da a mensagem:

```
VALOR ILEGAL ERRO EM 40
```

O valor de I atingiu 280, no entanto, o máximo valor aceito pela instrução HPLOT (máxima coordenada horizontal) é 279.

6.º) Exemplo (tente prever o resultado do programa):

```
]TEXT
]NEW
]10 HGR
]20 HOME
]30 HCOLOR = 5
]40 FOR I = 80 TO 110
]45 FOR J = 125 TO 155
]50 HPLOT J,I
]55 NEXT J
]60 NEXT I
```

2.3. HPLOT X1,Y1 TO X2,Y2

Esta é a instrução mais prática para o traçado de retas. Ela faz com que seja construída uma reta entre os pontos de coordenadas X1,Y1 e X2,Y2. Por exemplo, pode-se traçar uma linha oblíqua na tela, conforme mostra o programa:

```
]TEXT
]NEW
]10 HGR
]20 HCOLOR = 9
]30 HPLOT 1,1 TO 279,150
```

Através de uma única instrução, podem-se traçar várias retas. Desta forma, o exemplo do traçado de um retângulo com lados paralelos aos limites da tela, apresentado no subitem anterior, pode também (e de maneira mais rápida) ser obtido por meio do programa:

```
]TEXT
]NEW
]10 HGR
]20 HCOLOR = 3
]30 HPLOT 1,1 TO 258,1 TO 258,157 TO 1,157 TO 1,1
```

Observe agora as figuras formadas através dos seguintes programas:

a)
```
]TEXT
]NEW
]10 HGR
]20 HCOLOR = 3
]30 FOR I = 0 TO 60 STEP 10
]40 HPLOT I,I TO 259-I,I
]50 NEXT I
```

b)
```
]TEXT
]NEW
]10 HGR
]20 HCOLOR = 2
]30 FOR I = 0 TO 60 STEP 10
]40 HPLOT I,I TO 258-I,I TO 258-I, 159-I TO I,159-I TO I,I
]50 NEXT I
```

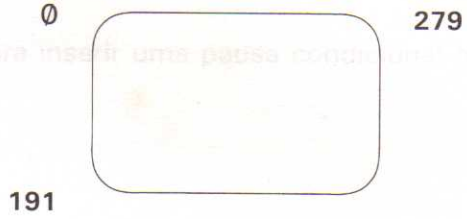
c)
```
]TEXT
]NEW
]10 HGR
]20 HCOLOR = 5
]30 FOR I = 0 TO 150 STEP 5
]40 HPLOT I,I TO 258-I,I TO 258-I, 159-I TO I,159-I TO I,I
]50 NEXT I
```

2.4. HGR2

No modo de alta-resolução gráfica, acionado por HGR, as quatro linhas finais da tela permanecem no modo texto. O campo de alta-resolução permite a definição de 280 por 160 pontos. HGR2 faz com que o modo de alta-resolução gráfica passe a preencher a tela inteira, definindo até 280 por 192 pontos, e omitindo o modo texto das últimas quatro linhas.

Veja como fica o exemplo do subitem anterior, quando ampliado até os limites da tela de alta-resolução.

```
]TEXT
]NEW
]10 HGR2
]20 HCOLOR= 3
]30 FOR I = 0 TO 60 STEP 10
]40 HPLOT I,I TO 279 - I,I TO 279 - I,191 - I TO I,191 - I TO I,I
]50 NEXT I
```



XIV - LINGUAGEM DE MÁQUINA

1. Uso de Rotinas em Linguagem de Máquina

O uso da linguagem de máquina permite uma maior flexibilidade, melhor aproveitamento da memória e facilidade na programação do TK //e. Este assunto, porém, é bastante extenso e apresenta uma certa complexidade, de forma que deverá ser pesquisado em manuais de programação avançada.

Neste capítulo descrevemos, de forma sintética, as instruções BASIC relacionadas com o uso da linguagem de máquina, possibilitando a programadores mais experientes utilizá-las.

Obs.: As instruções PEEK e POKE, que se incluiriam neste capítulo, foram comentadas no item 4 do capítulo VI.

1.1. CALL

Esta instrução provoca o desvio do programa para uma rotina em linguagem de máquina. Sua forma geral é:

```
CALL xyz
```

Onde xyz é o endereço inicial da rotina desejada. Para que, depois da execução da rotina, o programa volte automaticamente à próxima ordem após CALL, a última instrução da rotina deve ser RETURN.

Um exemplo da utilização desta instrução é CALL -936, cujo efeito é exatamente o mesmo do comando HOME.

1.2. WAIT

Esta instrução é usada para inserir uma pausa condicional no programa.

Sua forma geral é:

```
WAIT x,y,z
```

x é um endereço de memória, que deve estar entre os limites Ø e 65535. Estando fora desses limites, provoca uma mensagem de erro do tipo:

```
? VALOR ILEGAL ERRO
```

x e z devem estar entre os limites Ø e 255 decimal. Quando WAIT é executado, estes valores são convertidos para binário entre Ø e 11111111.

Sendo especificado apenas y, é feita uma operação AND com cada um dos bits deste e do conteúdo que se encontra no endereço x. Se o resultado deste processo são oito zeros, o teste é repetido. E se algum dos resultados é diferente de zero, isto é, se houver pelo menos um bit igual a 1 nos dois bytes, o WAIT completa-se e o programa prossegue na próxima instrução.

Sendo especificados os três parâmetros, WAIT executa o seguinte: primeiramente, uma operação XOR (ou exclusivo) com cada bit do byte contido no endereço x, com o correspondente bit do byte z. Em seguida, executa um AND entre cada bit resultante da operação citada e do byte y. Se o resultado deste conjunto de operações são oito zeros, o teste é repetido. Em caso diverso, o programa prossegue na próxima ordem.

1.3. USR(x)

Esta instrução permite a transferência do valor x para o acumulador de ponto flutuante (endereços $9D a $A3) e permite também que seja efetuado um desvio para o endereço $ØA. Aí deve existir uma rotina ou JMP adequados. O valor de retorno para a função é o conteúdo do acumulador de ponto flutuante.

1.4. HIMEM:

Esta instrução permite determinar a máxima posição de memória disponível para o programa BASIC, incluindo as variáveis. É usada para proteger a área de memória acima do valor determinado. A área protegida pode conter dados, tabelas de figuras ou rotinas em linguagem de máquina.

A forma geral desta instrução é:

```
HIMEM: x
```

Onde x é o máximo endereço desejado, podendo variar de Ø a 65535. Estando fora desses limites ocasiona uma mensagem de erro:

```
? VALOR ILEGAL ERRO
```

1.5. LOMEM:

Semelhante à instrução anterior, esta define o menor endereço disponível para variáveis. Normalmente, o TK //e define automaticamente LOMEM: como o endereço após a área de programa.

Forma geral da instrução:

```
LOMEM: x
```

Onde x é o endereço desejado.

2. Sistema Monitor

Dentro da ROM do TK //e existe um poderoso programa chamado Monitor Disassembler (MD). Ele atua como supervisor do sistema de operação. Usando-o, pode-se descobrir os segredos das 65.536 posições de memória.

Através dele, tornam-se possíveis as seguintes manipulações: verificar um ou vários endereços; mudar o conteúdo de qualquer endereço da RAM; escrever diretamente programas em linguagem de máquina; comparar e transferir blocos de dados.

2.1. Entrando no monitor

O programa Monitor inicia no endereço $FF69 (ou -151 em decimal). Para entrar no Monitor, deve ser usado inicialmente um comando CALL -151. O sistema responde com um asterisco na margem esquerda do vídeo.

O Monitor aceita linhas de entrada de dados com até 255 caracteres e alguns comandos (aceitos assim que se pressiona a tecla return).

Para retornar ao BASIC, pressione control C e return ou simplesmente control reset.

O Monitor opera diretamente com números hexadecimais, usando quatro dígitos para expressar endereços (os mais significativos, se houver, não precisam ser digitados) e dois dígitos para expressar conteúdos. O programa possui cinco registradores, dois dos quais são especiais: um fornece o endereço da última posição de memória utilizada e o outro, o endereço da próxima posição cujo conteúdo pode ser mudado. Eles são chamados de: "última posição aberta" (UPA) e "próxima posição alterável" (PPA). A utilidade destes dois registradores será mostrada adiante.

2.2. Examinando a memória

Quando se digita o endereço de apenas uma posição da memória, o Monitor responde com o endereço digitado, um traço, um espaço e o valor da posição de memória endereçada.

Por exemplo:

```
*F800     return
*F800- 4A
*-
```

Cada vez que o Monitor mostra o valor contido numa posição, lembra esta como UPA. Neste caso, também é considerada como PPA.

Se é digitado um ponto (.) seguido de um endereço, o Monitor mostra os valores contidos nas posições desde a UPA até o endereço digitado (desde que este último seja maior que o correspondente à UPA).

Observe os exemplos a seguir:

```
*FA00    return

FA00- D8 return
* 62 5A 48 26 62 94 88 return

FA08- 54 44 C8 54 68 44 EB 94
*F800    return

F800- 4A
*.F812   return
F801- 08 20 47 F8 28 A9 0F
F808- 90 02 69 E0 85 2E B1 26
F810- 45 30 25
*.F82A   return

F813- 2E 51 26 91 26
F818- 60 20 00 F8 C4 2C B0 11
F820- C8 20 0E F8 90 F6 69 01
F828- 48 20 00
*
```

Outra forma de usar o ponto é digitar o endereço inicial, ponto e endereço final da área de memória a ser examinada:

```
*F200.F22F  return

F200- 05 86 2D 86 2C 60 4C 99
F208- E1 20 EC F1 E4 F0 B0 08
F210- A5 F0 86 F0 85 2C 85 2D
F218- A9 C5 20 C0 DE 20 F8 E6
F220- E0 30 B0 E2 60 20 EC F1
F228- 8A A4 F0 C0 28 B0 D7 4C
*F900.F910  return

F900- F8 69 BF 20 ED FD CA D0
F908- EC 20 48 F9 A2 06 A4 2F
F910- E0
*
```

Cuidado ao examinar as posições de E/S entre $C000 e $C0FF: algumas delas, não sendo tratadas corretamente, bloqueiam o funcionamento do computador.

2.3. Alterando o conteúdo da memória

Agora veremos como usar o endereço PPA. Os valores contidos nestes endereços são aleatórios.

Digite a seguinte seqüência:

```
*4200    return

4200- A0
*:12     return
```

O conteúdo de PPA é alterado com o valor que você digitou. Confira examinando a posição novamente:

```
*4200    return

4200- 12
*
```

Pode-se alterar o conteúdo de várias posições consecutivas, separando os novos valores com um espaço:

```
*4000:12 23 34 45 56 67 78 89                    return

*4000                                            return

4000- 12
*                                                return
23 34 45 56 67 78 89
*4200:0 1 2 3                                    return

*:4 5 6 7                                        return

*4200.4207                                       return

4200- 00 01 02 03 04 05 06 07                    return
*
```

O TK //e permite também que se introduzam caracteres ASCII. Os caracteres devem ser precedidos por um apóstrofo (') e separados entre si por um espaço. O Monitor calcula automaticamente o código de cada caracteres.

Por exemplo:

```
*4000: 'M 'I 'C 'R 'O 'D 'I 'G 'I 'T 'A 'L

*4000.400B

4000-    CD C9 C3 D2 CF C4 C9 C7
4008-    C9 D4 C1 CC
```

2.4. Transferindo um bloco de memória

Pode-se considerar um bloco de memória (definido por dois endereços separados por um ponto) como uma entidade autônoma. É possível, então, movimentá-lo ou transferi-lo de um lugar para outro, usando o comando M (MOVE, de movimentar). Deve ser indicado o endereço inicial e o final do bloco de memória, e ainda o endereço inicial do destino (posição de memória a partir da qual é deslocado o bloco de memória). O comando deve ter a seguinte forma:

{destino} < {início} . {final} M

As palavras entre as chaves devem ser substituídas por endereços hexadecimais.

Por exemplo:

```
*4600.460F    return

4600- A0 A0 A0 A0 A0 A0 A0 A0
4608- A0 A0 A0 A0 A0 A0 A0 A0
*4000:00 11 22 33 44 55 66 77 88 99 AA BB CC DD EE FF    return

*4000.400F    return

4000- 00 11 22 33 44 55 66 77
4008- 88 99 AA BB CC DD EE FF
*4600<4000.400FM    return

*4600.460F    return

4600- 00 11 22 33 44 55 66 77
4608- 88 99 AA BB CC DD EE FF
*4700.4707    return

4700- A0 A0 A0 A0 A0 A0 A0 A0
*4700<4000.4007M    return

*4700.4707    return

4700- 00 11 22 33 44 55 66 77
*
```

Se o endereço de destino fica dentro do bloco podem ocorrer problemas. Assim, o bloco entre o endereço inicial e o destino é tratado como sub-blocos, e os valores nele contidos são duplicados.

Veja:

```
*4200.4307    return

4200- A0 A0 A0 A0 A0 A0 A0 A0
4208- A0 A0 A0 A0 A0 A0 A0 A0
4300- A0 A0 A0 A0 A0 A0 A0 A0
*4200:00 11 22 33 44 55 66 77 88 99 AA BB CC DD EE FF    return

*4206<4200.4301M    return

*4200.4307    return

4200- 00 11 22 33 44 55 00 11
4208- 22 33 44 55 00 11 22 33
4300- 44 55 00 11 22 33 44 55
*
```

2.5. Comparando dois blocos de memória

Pode-se verificar dois blocos de memória através do comando VERIFY. Tal procedimento é recomendável, em especial, depois do comando M. Caso haja uma discrepância, são indicados o endereço onde a discrepância dá-se e os dois valores discrepantes. O comando VERIFY é do tipo:

{destino} < {início} . {final} V

Exemplo:

```
*4000:00 11 22 33 44 55 66 77    return

*4100<400.407M    return

*4100<400.407V    return

*4006:AA    return

*4100<4000.4007V    return

4006-AA (66)
*
```

2.6. Soma e subtração de valores haxadecimais

Através do modo Monitor, que emprega a linguagem de máquina, pode-se fazer com que o TK //e efetue a soma e a subtração de valores hexadecimais com até dois dígitos.

Exemplos:

a) `* 5B + 2F    return`
`=8A`

b) `* CF + 68    return`
`=37`

No exemplo "b", o resultado verdadeiro seria 137. Mas, como tal resultado supera dois dígitos, o TK //e despreza o "vai um".

```
c) * FF - 42      return
   = BD
```

2.7. Salvando e carregando blocos de memória em fita cassete

O comando W (Write) permite armazenar em fita cassete e dela carregar no computador blocos de memória. Este comando permite transferir de 1 até 65.536 bytes.

Para armazenar, há duas versões do comando WRITE (W):

{início} . {final} W

A operação com o gravador é similar àquela utilizada com os comandos SAVE e LOAD em linguagem BASIC. Vejamos um exemplo:

```
*4000:0 1 2 3 4 5 6 7 8 9 A B C D E F    return

*4000.40F  return

4000- 00 01 02 03 04 05 06 07
4008- 08 09 0A 0B 0C 0D 0E 0F
*4000.400FW
OK
*
```

Para carregar o comando READ (R):

{início} . {final} R

```
*4000.400F    return
4000- 00 01 02 03 04 05 06 07
4008- 08 09 0A 0B 0C 0D 0E 0F
*
```

Se o programa é carregado corretamente, aparece o símbolo * , em caso contrário, aparece ERR.

Observe que é possível carregar um bloco numa posição diferente daquela em que foi armazenado.

2.8. Salvando e carregando blocos de memória em disquete

O TKDOS tem três comandos destinados a manipular blocos de memória em binário: BSAVE, BLOAD e BRUN.

2.8.1. BSAVE

BSAVE permite que se grave um programa em linguagem de máquina ou um arquivo com dados binários. Sua sintaxe correta é:

BSAVE NOMEARQ,Ann, Lmm

NOMEARQ corresponde ao nome do arquivo binário e segue as mesmas regras de um nome de arquivo em BASIC ou texto. Estes arquivos são identificados no CATALOG por uma letra B (de binário) na posição mais à esquerda da tela.

Ann indica o endereço do início do bloco (nn é um número entre Ø e 65535). Lmm é o tamanho do bloco (mm é número entre Ø e 32767). O maior bloco que o TKDOS pode guardar num único campo é o de tamanho 32767.

Por exemplo, para gravar o bloco de 4ØØ a 4ØF, deve-se, em primeiro lugar, converter os dois endereços para decimal:

$$4ØØ_{16} = 4 \times 16^2 + Ø \times 16^1 + Ø \times 16^0 = 524_{1Ø}$$

$$4ØF_{16} = 4 \times 16^2 + Ø \times 16^1 + 15 \times 16^0 = 539_{1Ø}$$

O comprimento do bloco é igual a:

(endereço final - endereço inicial) + 1 = 16

Para salvá-lo em disco, deve-se digitar:

BSAVE BLOCO1,A524,L16

2.8.2. BLOAD

A sintaxe de BLOAD é:

BLOAD NOMEARQ, Ann

NOMEARQ é o nome de um arquivo gravado em binário (identificado com a letra B no CATALOG).

Ann é o endereço inicial a partir de onde o arquivo é lido. Este endereço não precisa ser idêntico ao especificado em BSAVE e é opcional. Se é omitido, o TK //e considera-o igual ao endereço com que o bloco foi gravado.

2.8.3. BRUN

BRUN é equivalente a RUN no BASIC e executa um programa em linguagem de máquina. Sua sintaxe é:

BRUN NOMEARQ, Ann

NOMEARQ corresponde ao nome de um dos arquivos binários listados no CATALOG (marcados com a letra B) e nn é o endereço onde deve ser armazenado o programa (o programa é executado a partir deste endereço). Se Ann é omitido, o TK //e considera o endereço igual àquele com o qual o arquivo foi gravado.

2.9. Criando e executando programas em linguagem de máquina

Depois de introduzir um programa em linguagem de máquina, digite o endereço inicial seguido da letra G e return para que ele seja executado.

Exemplos:

{endereço} G

A letra G é um comando do modo Monitor para transferir o controle do TK //e ao programa, no endereço especificado.

O programa em linguagem de máquina é considerado uma sub-rotina. Para retornar ao Monitor, deve-se colocar, no final do programa, a instrução de retorno (RTS) do 65C02. Digite o seguinte programa e execute-o:

```
*4000:A9 C1 20 ED FD 18 69 1 C9 DB D0 F6 60 FF FF FF    return

*4000.400F    return

4000- A9 C1 20 ED FD 18 69 01
4008- C9 DB D0 F6 60 FF FF FF
*4000G    return
ABCDEFGHIJKLMNOPQRSTUVWXYZ
*
```

2.10. Outros comandos

Pode-se usar a saída de vídeo no modo normal (N) ou no modo inverso (I). Veja como são usados os comandos:

```
*F800.F80F    return

F800- 4A 08 20 47 F8 28 A9 0F
F808- 90 02 69 E0 85 2E B1 26
*I    return

*F800.F80F    return

F800- 4A 08 20 47 F8 28 A9 0F
F808- 90 02 69 E0 85 2E B1 26
*N    return

*F800.F80F    return

F800- 4A 08 20 47 F8 28 A9 0F
F808- 90 02 69 E0 85 2E B1 26
*
```

2.11. Examinando registradores

O Monitor reserva cinco posições de memória para os cinco registradores internos do 65CØ2. Estes registros são representados pelas seguinte letras:

A = acumulador
X = registrador de índice X
Y = registrador de índice Y
P = registrador de status do processador (status register)
S = apontador de pilha (stack pointer)

Os registradores acima podem ser consultados a qualquer momento pelo Monitor, através do comando EXAMINE, obtido por um comando control E. O processo é o seguinte:

Estando no modo Monitor, digite o comando control E (EXAMINE) e, em seguida, pressione a tecla return. O resultado é do tipo indicado abaixo:

A = FF X = ØØ Y = ØØ P = BØ S = F4

Os valores à direita do sinal de igualdade correspondem aos últimos valores dos registradores.

Obs.: Os valores apresentados no exemplo podem variar, conforme a situação.

Observe que à posição apresentada pelo apontador de pilha S deve-se somar $ØlØØ, pois o endereço está na segunda página de memória.

Por exemplo:

A = ØØ X = EA Y = BF P = CØ S = FØ

O endereço real da pilha é $ØlØØ + $ØØFØ = $ØlFØ

Nota: Para aprender a programar em linguagem de máquina, consulte bibliografia sobre o microprocessador 65Ø2 ou 65CØ2.

3. O Mini Assembler

O Mini Assembler é um programa que simplifica o trabalho de digitação de programas em linguagem de máquina. Ele permite o uso de mnemônicos em vez de códigos em linguagem de máquina.

O uso do mini assembler pressupõe um conhecimento da linguagem Assembly do 65Ø2. Desejando utilizá-lo, procure bibliografia sobre o microprocessador 65Ø2, pois, neste item, não entramos em detalhes sobre programação.

3.1. Acessando o Mini Assembler

Para obter o Mini Assembler, proceda do seguinte modo:

a) Entre no modo Monitor (CALL-151)

b) tecle [!] é [return]

O sinal de prompt muda para ! , indicando a presença do Mini Assembler.

Nota: O Mini Assembler também pode ser obtido a partir do Integer BASIC.

3.2. Usando o Mini Assembler

Para fazer uso do Mini Assembler, primeiramente se deve indicar o endereço inicial do programa. A primeira instrução deve ser precedida do endereço de sua localização (todos os valores devem ser hexadecimais). Antes de digitar o mnemônico da instrução, digite dois pontos.

Por exemplo:

```
!0D83:LDA $C030
```

O computador responde, decodificando a instrução e mostrando o resultado na tela:

```
0D83- AD 30 C0     LDA $C030
```

A instrução a seguir pode ser digitada sem o endereço, bastando que se digite um espaço logo após o sinal de prompt.

Por exemplo:

```
! DEY
```

Havendo algum erro, o Mini Assembler sinaliza o local da ocorrência com ^, quando se tecla [return].

Por exemplo:

```
! BEN $0D8E
    ^
```

Para corrigir, basta que se digite a instrução certa:

```
! BNE $0D8E
```

3.3. Saindo do Mini Assembler

Para sair do Mini Assembler e voltar para ao modo Monitor, basta digitar uma vez a tecla [return], antes de ter digitado qualquer coisa após o último sinal de prompt:

3.4. Digitando um programa em Mini Assembler

Mostramos agora um programa em linguagem de máquina que gera um sinal de dois tons, como uma sirene.

Vamos armazenar este programa a partir do endereço $ØD83. Os endereços em $ØD81 entre $ØD82 vão ser usados para armazenamento intermediário, e devem conter incialmente os valores FF e FF.

Entre no modo Monitor e coloque os seguintes valores:

```
!    return
*ØD81:FF FF    return
*!    return
```

Digite inicialmente o primeiro endereço, seguido por dois pontos, e a primeira instrução:

```
!ØD83:LDA $CØ3Ø
```

Digite agora o restante do programa. Não se esqueça de incluir um espaço após o sinal de prompt:

```
! DEY
! BNE $ØD8E
! DEC $ØD82
! BEQ $ØD97
! DEX
! BNE $ØD86
! LDX $ØD81
! JMP $ØD83
! RTS
! LDY #$ØØ
! LDX #$ØØ
! LDA #$47
! STA $ØD81
! LDA #$AØ
! STA $ØD82
! JSR $ØD83
! BIT $CØØØ
! BPL $ØDB5
! LDA $CØØØ
! BIT $CØ1Ø
! RTS
! LDY #$ØØ
! LDX #$ØØ
! LDA #$6Ø
! STA $ØD81
! LDA $#AØ
! STA $ØD82
! JSR $ØD83
! JMP $ØD98
! JSR $ØD98
! CMP $#83
! BNE $ØDC9
! RTS
```

Para testar este programa, tecle return, obtendo sinal de prompt * e, a seguir:

```
*DC9G
```

Para usá-lo em BASIC, coloque numa linha do programa:

```
]CALL 3529
```

XV - MANIPULAÇÃO DE FIGURAS EM ALTA-RESOLUÇÃO

1. Introdução

O TK //e possui cinco comandos especiais que permitem manipular figuras no modo gráfico de alta-resolução:

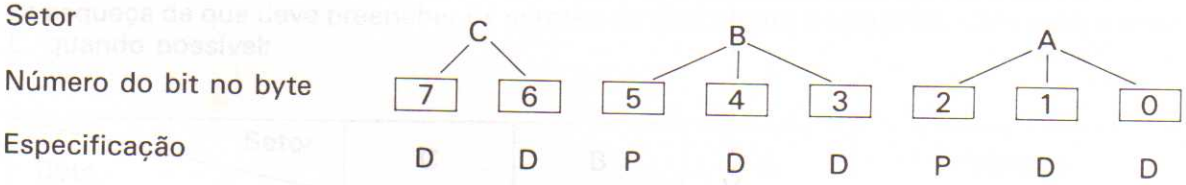
DRAW XDRAW SCALE ROT SHLOAD

Para que estes comandos possam ser usados, deve-se construir um traçado através duma "definição de figuras". Tal definição de figuras é constituída de uma seqüência de vetores armazenados numa série de bytes da memória do TK //e. Uma ou mais destas definições (cujo conjunto forma uma "tabela de figuras") podem ser criadas através do teclado e armazenadas em disquete ou fita cassete para uso futuro.

Cada byte da definição de figura é dividido em três setores, e cada setor pode especificar um vetor, podendo ou não demarcar um traço, e também uma direção de movimento (cima, baixo, direita ou esquerda).

DRAW e XDRAW utilizam cada byte, na definição da figura, setor por setor, desde a primeira até a última definição do byte. Quando um byte que contenha somente zeros é alcançado, a definição da figura é finalizada.

Distribuição de três setores A, B, e C num byte que forma uma definição de figura:



Cada par de bits DD especifica uma direção de movimento, e cada bit P especifica se o traço deve ou não ser demarcado antes do movimento, conforme indica a tabela a seguir:

Bits	Valor	Ação
DD	ØØ	move para cima
DD	Ø1	move para a direita
DD	1Ø	move para baixo
DD	11	move para a esquerda
P	Ø	não demarca
P	1	demarca

O último setor C (os dois bits mais significativos) não tem bit do tipo P.

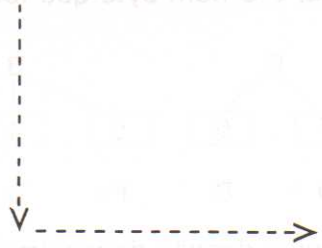
Para este setor, P é considerado valor igual a zero, de modo que o setor C pode especificar somente movimento, sem traços demarcados.

DRAW e XDRAW processam os setores da direita para a esquerda (A, B e, então, C). Todo setor do byte que contém somente zeros é ignorado. Desta forma, o byte não pode acabar com um movimento no setor C que corresponda a Ø0 (para cima), dado que este setor contém apenas zeros. Neste caso, B não pode fazer o movimento Ø0Ø, que é ignorado, nem é possível o movimento Ø0Ø no setor A (que finaliza a definição da figura), a menos que haja um bit 1 em B ou C.

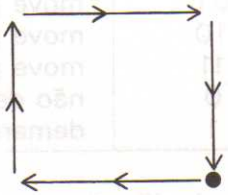
Suponha que você queira desenhar a forma que aparece na figura abaixo:

Primeiramente desenhe esta forma em papel quadriculado, usando dois traços dos quadrados para cada lado da figura. Em seguida opte por onde começar a definição da figura. Para efeito de rotação (tema abordado posteriormente), é aconselhável que a forma seja iniciada por um centro (ponto eqüidistante).

Vamos começar o desenho pelo centro. Descreva um movimento para baixo sem desenhar. Usando apenas ângulos de 9Ø graus para as mudanças de direção, execute um movimento (ainda sem demarcar) para a direita, conforme indica a figura:

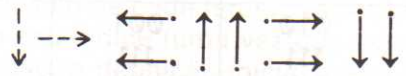


Recomponha o quadrado com uma série de vetores, cada um com um movimento para a esquerda, para cima, para a direita ou para baixo. Lembre-se de que, ao se moverem, os vetores devem marcar um traço:



ponto inicial
definido no passo anterior

Agora junte os vetores numa seqüência:



A tabela 1, a seguir, apresenta os vetores e seus respectivos códigos:

Vetor	Código	
↑	000	
-->	001 ou 01	somente movimentam
↓	010 ou 10	
<--	011 ou 11	
↑	100	
→	101	plotam e movimentam
↓	110	
←	111	

tabela 1

Baseando-se na tabela 1, execute a codificação dos vetores, como mostra a tabela 2. Não se esqueça de que deve preencher os setores da direita para a esquerda, utilizando o setor C, quando possível:

Byte \ Setor	C	B	A	Vetores
0		010	001	↓ -->
1		111	111	← ←
2		100	100	↑ ↑
3		101	101	→ →
4		110	110	↓ ↓
5	00	000	000	finaliza

tabela 2

Na tabela a seguir, você obtém a conversão dos vetores em código hexadecimal, pois é neste formato que eles entram na memória. Para construir a tabela 3, utilize as informações contidas na tabela 4.

Byte \ Setor	C	B	A	Hexadecimal
0	00	010	001	11
1	00	111	111	3F
2	00	100	100	24
3	00	101	101	2D
4	00	110	110	36
5	00	000	000	00

1.º dígito 2.º dígito

tabela 3

Os valores hexadecimais são obtidos aplicando-se cada grupo de 4 bits de um byte na tabela 4:

Cod. Binário	Cod. Hex.
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7

Cod. Binário	Cod. Hex
1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

tabela 4

Por exemplo, 3F é obtido inicialmente, através da tabela 2. Juntando dois vetores que movimentam e desenham para a esquerda, obtemos 111 e 111. Colocando estes dígitos lado a lado, e completando as casas à esquerda com zeros até preencher 8 posições (1 byte), temos: 00111111

Em seguida, separando, da esquerda para a direita, de quatro em quatro, para formar os dois dígitos:

0011 1111
1.º 2.º

Confirme, na tabela 4, que:

0011 = 3 e 1111 = F

Portanto, convertendo-se 0011 e 1111 em hexadecimal, obtêm-se 3F.

O conjunto de bytes a que chegamos, apresentado na tabela 3, é a definição da figura. Ainda é necessário que se forneçam alguns parâmetros a mais para que a tabela de figuras fique completa.

A localização da tabela, cujo endereço genérico denominamos S, deve conter o número de definições de figuras (de Ø a 255), em hexadecimal. No nosso caso, o número é 1. Em seguida devemos incluir os índices (em hexadecimal) que exprimem quantos bytes após o inicial começa a definição de cada figura. Vamos colocar nossa definição de figura imediatamente abaixo de seu índice (uma vez que apenas há uma definição de figura). Isto significa que, neste caso, a definição inicia-se em S+4.

Endereço inicial = S

bytes		
	S+Ø	n (Ø a FF) ← número total de definições de figuras
	+1	não usado
	+2	2 dígitos de menor valor
	+3	2 dígitos de maior valor — D1: índice do 1.º byte de definição da figura 1, relativamente a S
	+4	2 dígitos de menor valor
	+5	2 dígitos de maior valor — Dn: índice do 1.º byte de definição da figura n, relativamente a S
	+2n+1	2 dígitos de menor valor

Formato geral da tabela de definições de figuras:

S+D1	1 byte	definição da figura 1
	:	
	último byte = ØØ	
	:	
S+D2	1 byte	definição da figura 2
	:	
	último byte = ØØ	
S+Dn	1 byte	definição da figura n
	:	
	último byte = ØØ	

tabela 5

A figura a seguir mostra como a definição da forma do quadrado de nosso exemplo é carregada na memória do computador. Ela retrata uma aplicação da tabela 5 aos dados até aqui compilados:

Byte	0	01	→ número de figuras
	1	00	→ não usado
	2	04	índice da definição da
	3	00	figura 1
	4	11	definição da figura 1
	5	3F	
	6	24	
	7	2D	
	8	36	
	9	00	→ byte de finalização da definição de figuras

tabela 6

1.1. Modo Monitor

Para que os dados apresentados na tabela 6 sejam registrados, deve-se entrar no modo Monitor do TK //e (veja o subitem 2.1 do cap. XIV).

1.2. Carga da tabela de figuras

Agora está tudo pronto para que a tabela de figuras seja carregada na memória do TK //e.

É necessária a escolha de um endereço inicial. Para nosso exemplo, vamos selecionar 1DFC.

Nota: O endereço escolhido deve ser menor que o maior endereço existente na memória do sistema e não deve estar numa área utilizada por HGR. O endereço 1DFC é justamente abaixo da primeira página da memória utilizada para a definição de gráficos de alta-resolução.

Digite o endereço inicial da tabela de figuras:

```
* 1DFC
```

Se a tecla return é pressionada, o TK //e mostra o conteúdo do endereço especificado. Este conteúdo não nos interessa, visto que vai ser alterado pela introdução de nossa tabela, com o passo seguinte. Em lugar de return, você deve digitar (logo após o *):

```
* 1DFC: 01    return
* 1DFC    return
```

A tela apresenta:

```
1DFC- 01
*
```

O TK //e diz que o valor Ø1 (hexadecimal) está armazenado no endereço 1DFC. Para armazenar mais do que dois dígitos de números hexadecimais em bytes sucessivos de memória, basta entrar o primeiro endereço (1DFC) e, então, os valores desejados, separados por espaço.

Vamos entrar agora nossa tabela:

```
* 1DFC: Ø1 ØØ Ø4 ØØ 11 3F 24 2D 36 ØØ   return
```

Para conferir as informações da tabela de figuras, pode-se examinar cada byte separadamente ou simplesmente pressionar repetidas vezes a tecla Return até que todos os bytes sejam apresentados:

```
* 1DFC   return
1DFC- Ø1
*   return
ØØ Ø4 ØØ
*   return
1EØØ- 11 3F 24 2D 36 ØØ BD FE
*
```

Não se preocupe com os bytes que aparecem após o ØØ, pois não fazem parte de sua tabela.

1.3. Indicando ao BASIC o endereço da tabela de figuras

Se a sua tabela de figuras está correta, tudo o que falta é indicar (para poder ser utilizado pelo BASIC) onde se encontra o endereço inicial da tabela - isto é feito automaticamente quando essa tabela é carregada de fita cassete pelo comando SHLOAD, conforme é visto adiante. O BASIC "procura" esta indicação no endereço E8 (dois dígitos de menor ordem) e E9 (dois dígitos de maior ordem).

Para a nossa tabela, iniciada no endereço 1DFC, devemos digitar:

```
* E8: FC 1D   return
```

1.4. Proteção da tabela de figuras

É conveniente a proteção da sua tabela de figuras contra uma alteração acidental pelo programa BASIC. Com este propósito, vamos alterar HIMEM: nos endereços 73 e 74 hexadecimais, digitando:

```
* 73: FC 1D
```

Isto também é feito, automaticamente, pelo uso de SHLOAD.

1.5. Armazenando uma tabela de figuras e SHLOAD

Para armazenar sua tabela de figuras, você precisa conhecer três dados:

1.º) o endereço da tabela (1DFC, no nosso exemplo);
2.º) o último endereço da tabela (1EØ4, no nosso exemplo);
3.º) a diferença entre os endereços final e inicial (Ø8, no nosso exemplo).

Estes dados devem ser fornecidos em hexadecimal.

O terceiro dado (diferença entre os endereços final e inicial) deve ser armazenado nos endereços Ø (os dois dígitos menos significativos) e 1 (os dois dígitos mais significativos):

```
* Ø: Ø8 ØØ    return
```

Agora você pode "escrever" (armazenar na fita cassete) o tamanho e o endereço inicial e final da tabela:

```
* Ø.1W 1DFC.1EØ4W
```

Somente pressione return depois de colocar a fita cassete no gravador, bobiná-la e pressionar os controles de gravação (provavelmente as teclas PLAY e RECORD do gravador).

1.5.1. SHLOAD

Este comando executa a leitura da tabela gravada em fita cassete. Para tanto, volte ao BASIC (control C e return) e digite:

```
] SHLOAD    return
```

Logo a seguir, aperte a tecla PLAY do gravador. Você deve ouvir um sinal sonoro (bip) quando o número de bytes da tabela for lido com sucesso e um segundo sinal quando for concluída a leitura da tabela.

2. Usando a Tabela de Figuras

Podemos agora escrever um programa em BASIC usando a tabela de figuras e os comandos apropriados vistos a seguir.

O programa seguinte é bastante simples e define uma figura na tela, executa uma rotação de 16 graus e repete a figura apresentada. Cada nova figura é maior que a anterior. Execute-o com RUN:

```
] 10 HGR
] 20 HCOLOR=3
] 30 FOR R = 1 TO 40
] 40 ROT = R
] 50 SCALE = R
] 60 DRAW 1 AT 139,90
] 70 NEXT R
```

Para ver inicialmente um pequeno quadrado, acrescente a linha:

```
] 65 END
```

Você pode inserir uma pausa e apagar cada quadrado após sua apresentação, acrescentando:

```
] 63 FOR I = Ø TO 1ØØØ: NEXT I
] 65 XDRAW 1 AT 139,9Ø
```

Execute o comando RUN e observe agora o desenvolvimento do programa.

2.1. DRAW

O formato geral deste comando é:

```
DRAW n AT x,y
```

Quando DRAW é executado, desenha-se uma figura no modo gráfico de alta-resolução, começando no ponto cujas coordenadas são x e y. A figura é a enésima da tabela de figuras, carregada previamente pelo comando SHLOAD ou digitada no modo Monitor do TK //e. O valor "n" deve estar entre Ø e 255, e representa o número da figura indexada no byte zero da tabela de figuras previamente definida.

A coordenada x deve ter valores entre Ø e 278, e y, entre Ø e 191. Se o valor excede esses limites, ocorre a mensagem de erro:

```
? VALOR ILEGAL ERRO
```

A cor, a rotação e a escala da figura a ser apresentada devem ser indicadas antes que DRAW seja executado.

Se o comando DRAW é emitido sem que haja uma tabela de figuras, podem ser criadas figuras aleatórias em toda área de memória para gráficos de alta resolução, com provável destruição de seu programa. Nesta hipótese, deve-se pressionar reset ou control reset, ou control C e return.

2.2. XDRAW

Este comando é similar a DRAW, exceto quanto à cor usada para desenhar a figura, que é a complementar daquela já existente nos pontos determinados. Através de XDRAW pode-se suprimir da tela um traço, sem alterar a cor de fundo. Seu formato é semelhante ao da instrução DRAW:

```
XDRAW n AT x,y
```

2.3. ROT e SCALE

O formato da instrução ROT é:

```
ROT = x
```

Esta instrução tem por função executar uma rotação na figura a ser traçada por DRAW ou XDRAW. O ângulo de rotação é especificado por x, cujo valor pode ser de Ø a 255.

ROT = Ø especifica uma rotação nula; ROT = 16 faz com que a figura definida por DRAW execute uma rotação de 9Ø graus no sentido horário; ROT = 32 provoca uma rotação de 18Ø graus, e assim por diante. A repetição do processo ocorre quando ROT = 64.

A forma geral de SCALE é:

```
SCALE = x
```

Onde x determina a escala da figura.

A instrução SCALE determina o tamanho da figura a ser traçada por DRAW ou XDRAW, a partir do valor 1 (reproduzindo, ponto por ponto, a definição da figura) até 255 (quando cada vetor é ampliado 255 vezes).

SCALE = Ø é o máximo tamanho de escala permitido (e não o menor), pois é equivalente a SCALE = 255.

Para SCALE = 1, apenas quatro valores de rotação são recomendados: Ø, 16, 32 e 48; para SCALE = 2, são recomendadas oito rotações, e assim por diante.

Geralmente, os valores de rotação não reconhecidos não ocasionam a definição da figura orientada de acordo com o primeiro valor de rotação definido.

No programa-exemplo a seguir, é usada a mesma tabela de figuras do exemplo anterior:

```
] NEW
] 10 HGR
] 20 J = INT(RND(1)*6+1)
] 30 HCOLOR = J
] 40 FOR I = 1 TO 25
] 50 ROT = I
] 60 SCALE = I
] 70 DRAW 1 AT 70,90
] 80 DRAW 1 AT 210,90
] 90 NEXT I
] 100 GOTO 20
```

Surgem duas figuras iguais, e a cor varia randomicamente de uma execução para outra.

2.4. Definição de três figuras

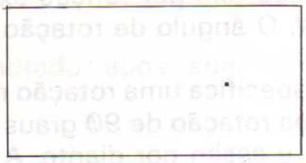
A tabela seguinte presta-se à definição de três figuras. Ela pode ser carregada na memória do computador no mesmo endereço (1DFC) e pelo mesmo método que você usou para carregar a primeira tabela.

As formas contidas na tabela são as seguintes:

fig. 1

fig. 2

fig.3



Byte				
	0	03	→	número de figuras
	1	00	→	não usado
	2	08	→	índice da definição da figura 1
	3	00	→	byte de separação de índices
	4	0E	→	índice da definição da figura 2
	5	00	→	byte de separação de índices
	6	16	→	índice da definição da figura 3
	7	00	→	byte de separação
	8	11		
	9	3F		
	10	24		definição da figura 1
	11	2D		
	12	36		
	13	00	→	byte de finalização da definição da fig. 1
	14	11		
	15	3F		
	16	24		
	17	24		definição da figura 2
	18	2D		
	19	36		
	20	36		
	21	00	→	byte de finalização da definição da fig. 2
	22	11		
	23	3F		
	24	3F		
	25	24		definição da figura 3
	26	2D		
	27	2D		
	28	36		
	29	00	→	byte de finalização da definição de figuras

Após ter introduzido a tabela na memória do computador, utilize o programa em BASIC para apresentar as três figuras na tela de Alta-Resolução:

```
] 10 HGR
] 20 HCOLOR=3
] 30 ROT=0: SCALE= 10
] 40 DRAW 1 AT 60,100 :REM FIGURA 1
] 50 DRAW 2 AT 120,100 : REM FIGURA 2
] 60 DRAW 3 AT 200,100 : REM FIGURA 3
```

2.5. SHLOAD

Como você já viu, o comando SHLOAD carrega uma tabela de figuras da fita cassete. Esta tabela é carregada exatamente abaixo de HIMEM:, e HIMEM: é fixado depois, abaixo da tabela, para protegê-la. No momento da leitura, o endereço inicial da tabela é fixado automaticamente. Se uma segunda tabela é carregada, substituindo a primeira, HIMEM: deve ser "resetado" antes do SHLOAD para evitar perda de memória.

Somente o comando control reset pode deter SHLOAD.

XVI - EFEITOS SONOROS

Uma das características do TK //e que chama a atenção é sua capacidade de gerar variada gama de sons. Clics, bips, tiros, explosões e até música podem ser produzidos mediante algumas operações.

1. O Som no BASIC

No TK //e, o som é produzido por um alto-falante interno, acionado por software.

Através da digitação de control G, produz-se um som agudo ("bell"). Num programa em BASIC, este som pode ser obtido da seguinte forma:

a) digita-se PRINT;

b) abrem-se aspas;

c) digita-se control G (este caractere não fica visível na tela);

d) fecham-se as aspas.

Pode-se, ainda, usar a função CHR$ para obter este som. O código do caractere utilizado é 7:

```
PRINT CHR$ (7)
```

2. Usando o PEEK (-16336)

Ao referenciar-se o endereço 49200, através de um PEEK (49200) ou PEEK (-16336), também se aciona o alto-falante.

Experimente usar este artifício, colocando-o numa linha de programa, como no exemplo:

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

Se esta é executada, emite-se apenas um clic através do alto-falante.

Acrescente agora a linha:

```
]160 GOTO 150
```

Isso faz com que o computador produza infinitos "clics". control C pára o programa.

Outra forma de alterar o som produzido é usar PEEK (-16336) várias vezes numa expressão.

Digite e execute o seguinte programa:

```
]10 REM Programa de efeitos sonoros
]20 S = -16336
]30 FOR I= 1 TO 100
]40 SOM = PEEK(S)+PEEK(S)-PEEK(S)-PEEK(S)-PEEK(S)-PEEK(S)-PEEK(S)
]50 NEXT I
```

Tente variar o número de PEEKs na expressão da linha 40. Varie também o sinal das parcelas e veja o que acontece.

3. Som em Linguagem de Máquina

Embora seja possível criar sons em BASIC da maneira que foi descrita, a faixa de freqüência de sons é limitada. No BASIC, as instruções têm de ser interpretadas antes de serem executadas. Tal fato determina uma demora que implica na menor freqüência do sons emitidos via alto-falante. A freqüência máxima obtida com o BASIC é de 72 Hz.

Os limites do BASIC podem ser bastante ampliados se são usadas rotinas em linguagem de máquina, para gerar determinados sons durante a execução de um programa. Em linguagem de máquina, a freqüência máxima possível está em torno de 50 kHz (o alto-falante limita a frequência na faixa audível). Assim, podem-se gerar toda a sorte de ruídos e também música.

Toda vez em que se referencia o endereço $C030 (49200 em decimal), gera-se um clic através do alto falante. Cada vez que este endereço é acessado, o sinal presente no alto-falante muda de zero para um ou vice-versa.

A seguir é apresentada uma série de efeitos sonoros em linguagem de máquina. Antes de digitar as instruções, entre no modo Monitor (CALL-151). Depois, digite o endereço seguido de dois pontos e os códigos hexadecimais.

METRALHADORA LASER

```
08B6: 20 BF
08B8: 08 20 BF 08 20 BF 08 A9
08C0: 00 85 FF A9 FF 85 FE A9
08C8: 00 8D 30 C0 EE 30 C0 CE
08D0: 30 C0 A6 FF CA D0 FD C6
08D8: FE F0 05 E6 FF 4C C7 08
08E0: 60
```

Nota: Para executar este programa no modo Monitor, basta digitar 08B6G ou, no modo BASIC, CALL 2230.

EXPLOSÃO

```
*1560: A9 07 85 06 A0 00 99 09
*1568: 85 FE A9 00 8D 30 C0 EE
*1570: 30 C0 CE 30 C0 A2 FF CA
*1578: D0 FD A2 FF CA D0 FD A2
*1580: FF CA D0 FD B6 21 CB CA
*1588: D0 FD B6 21 CA D0 FD B6
*1590: 21 CA D0 FD B6 21 CA D0
*1598: FD C6 FE F0 03 4C 6A 15
*15A0: A9 45 20 AB FC C6 06 D0
*15A8: BD 60
```

Nota: Para executar este efeito, basta digitar 1560G no modo Monitor ou, em modo BASIC, CALL 5472.

HELICÓPTERO

```
*15AC: A0 23 20 B4
*15B0: 15 88 D0 FA A9 01 85 FF
*15B8: A9 45 85 FE A9 00 8D 30
*15C0: C0 EE 30 C0 CE 30 C0 A6
*15C8: FF CA D0 FD C6 FE F0 05
*15D0: E6 FF 4C BC 15 A9 03 85
*15D8: 08 A9 20 85 06 A9 03 85
*15E0: 07 8D 30 C0 EE 30 C0 CE
*15E8: 30 C0 C6 06 D0 02 C6 07
*15F0: D0 F8 C6 08 D0 E3 60
```

Nota: Para executar este efeito, basta digitar 15ACG no modo Monitor ou, no modo BASIC, CALL 5548.

BATALHA

```
*14D6: A9 FF
*14D8: 85 07 A9 FF 85 09 A9 80
*14E0: 85 FE A9 00 8D 30 C0 EE
*14E8: 30 C0 CE 30 C0 A0 03 A6
*14F0: 09 CA D0 FD A2 64 CA D0
*14F8: FD 88 F0 03 4C EF 14 56
*1500: 09 A9 00 8D 30 C0 EE 30
*1508: C0 CE 30 C0 A0 03 A6 07
*1510: CA D0 FD A2 64 CA D0 FD
*1518: 88 F0 03 4C 0E 15 C6 FE
*1520: F0 05 E6 09 4C E2 14 20
*1528: 3C 15 20 3C 15 20 3C 15
*1530: 20 3C 15 20 3C 15 20 3C
*1538: 15 20 3C 15 A9 FF 85 FF
*1540: A9 9B 85 FE A9 00 8D 30
*1548: C0 EE 30 C0 CE 30 C0 A6
*1550: FF CA D0 FD C6 FE F0 05
*1558: E6 FF 4C 44 15 60
```

Para executar este programa, basta digitar 14D6G no modo Monitor e, em modo BASIC, CALL 5334.

Todos estes efeitos estão em endereços que não se sobrepõem, podendo ser utilizados num mesmo programa.

Por exemplo:

```
]5 FOR I=1 TO 10
]10 CALL 5548
]20 CALL 2230
]30 CALL 5334
]40 CALL 5472
]50 NEXT I
```

APÊNDICE A

FUNÇÕES TRIGONOMÉTRICAS

As funções trigonométricas apresentadas aqui não fazem parte do BASIC do TK //e, mas podem ser obtidas através das seguintes fórmulas:

SECANTE
```
SEC(X) = 1/COS(X)
```

COSSECANTE
```
COSC(X) = 1/TAN(X)
```

CO-TANGENTE
```
COT(X) = 1/TAN(X)
```

ARCO-SENO
```
ARCSEN(X) = ATN( X / SQR (-X * X * 1))
```

ARCO-COSSENO
```
ARCOS(X) = -ATN( X / SQR (-X * X + 1)) + 1.57Ø8
```

ARCO-SECANTE
```
ARCSEC(X) = ATN (SQR( X * X - 1)) + SGN(X) - 1) * 1.57Ø8
```

ARCO-CO-SECANTE
```
ARCCOSC(X) = ATN (1/SQR( X * X - 1)) + (SGN(X) - 1) * 1.57Ø8
```

ARCO-CO-TANGENTE
```
ARCOT(X) = -ATN(X) + 1.57Ø8
```

SENO HIPERBÓLICO
```
SINH(X) = (EXP(X) - EXP(-X)) / 2
```

COSSENO HIPERBÓLICO
```
COSH(X) = (EXP(X) + EXP(-X)) / 2
```

TANGENTE HIPERBÓLICA
```
TANH(X) = -EXP(-X) / (EXP(X) + EXP(-X)) * 2 + 1
```

SECANTE HIPERBÓLICA
```
SECH(X) = 2 / (EXP(X) + EXP(-X))
```

COSSECANTE HIPERBÓLICA
```
COSCH(X) = 2 / (EXP(X) - EXP(-X))
```

CO-TANGENTE HIPERBÓLICA
```
COTH(X) = EXP(-X) / (EXP(X) - EXP(-X)) * 2 + 1
```

SENO HIPERBÓLICO INVERSO
ARGSINH(X) = LOG(X SQR(X * X + 1))

COSSENO HIPERBÓLICO INVERSO
ARGCOSH(X) = LOG(X + SQR (X * X - 1)

TANGENTE HIPERBÓLICA INVERSA
ARGTANH(X) = LOG ((1 + X) / (1 - X)) / 2

SECANTE HIPERBÓLICA
ARGSECH(X) = LOG((SQR(-X * X + 1) + 1) / X

COSSECANTE HIPERBÓLICA INVERSA
ARGCOSCH(X) = LOG (SGN(X) * SQR(X * X + 1) + 1) / X

CO-TANGENTE HIPERBÓLICA INVERSA
ARGCOTH(X) = LOG((X + 1) / (X - 1)) / 2

A MOD B
MOD(A) = INT((A / B - INT(A / B) * B + .Ø5) * SGN(A / B)

APÊNDICE B

MENSAGENS DE ERRO

Após a ocorrência de um erro, o BASIC retorna ao nível de comando indicado pelo caractere de pronto (]) e o cursor. O valor das variáveis bem como os textos de programas permanecem intactos, mas o programa não pode prosseguir por meio do comando CONT. Todos os contadores dos "loops" GOSUB e FOR são zerados. Para precatar-se contra isto, você pode usar uma sub-rotina apropriada, através de um ONERR GOTO.

As mensagens de erro são exibidas em dois formatos:

1 - ?XX ERRO. Este formato é apresentado quando ocorre erro em comando direto (modo imediato).

2 - ?XX ERRO EM YY. Mensagens deste tipo aparecem quando ocorre erro em modo programado, ou seja, em comandos cuja execução não é imediata.

Em ambos os casos XX representa o nome do erro específico. Em modo programado, YY é o número da linha na qual o erro ocorreu. Neste modo o erro somente é detectado quando a linha em que ele se encontra é executada.

Todas as mensagens de erro do sistema são precedidas de um ponto de interrogação (?). Se uma mensagem aparece sem este sinal, trata-se de uma mensagem do tipo "não-erro" (como "PAROU EM 2Ø6", por exemplo) ou, então, de mensagem proveniente do sistema DOS (por ex., "ERRO E/S").

Segue uma breve explicação dos erros e de suas causas:

NEXT SEM FOR

A variável utilizada no comando NEXT é diferente da variável usada no último comando FOR, ou então existe algum comando NEXT sem que houvesse previamente um comando FOR.

SINTAX

Falta de parênteses, pontuação incorreta, comando digitado errado ou inexistente, caractere ilegal, etc.

RETURN SEM GOSUB

Existe algum comando RETURN sem que previamente tivesse sido definido um comando GOSUB.

FALTA DADOS

Um comando READ foi encontrado, porém não há mais DATA disponível (todas as variáveis já foram lidas ou inexiste o comando DATA).

VALOR ILEGAL

Pode ocorrer numa das seguintes hipóteses:

1. Valor negativo na definição de um array, exemplo: LET A(-3) = Ø
2. LOG com argumento nulo ou negativo.
3. SQR com argumento negativo.
4. Uso de argumento impróprio nos comandos: MID$, LEFT$, RIGHT$, WAIT, PEEK, POKE, TAB, SPC, ON ... GOTO ou nos comandos gráficos.
5. A ^ B, sendo A negativo e B não inteiro.

ESTOURO

O resultado de uma operação matemática excede a capacidade do BASIC para representá-lo.

SEM MEMORIA

Pode ocorrer numa das seguintes situações:

1. Programa muito extenso.
2. Excesso de variáveis.
3. Mais do que 1Ø niveis de loops de FOR ... NEXT.
4. Mais do que 24 niveis de loops de GOSUB ... RETURN.
5. Mais do que 36 níveis de cálculo (parênteses).
6. LOMEM maior que HIMEM, ou um destes fora dos limites.

COMANDO INEXISTE

Ocorre quando um GOTO ou GOSUB referencia uma linha inexistente de programa.

INDICE ILEGAL

Ocorre quando o índice de um array é maior que o DIM especificado. Ex.: LET A(1,1,1)=5 ou LET A(1,3)=5, sendo dimensionado DIM A(2,2).

REDIM ARRAY

Ocorre quando um array é dimensionado mais de uma vez dentro de um programa.

DIVISAO POR ZERO

Qualquer tentativa de divisão por zero acusa erro.

COMANDO ILEGAL

Os comandos INPUT, DEF FN, GET, DATA não podem ser usados no modo imediato.

INCOMPATIVEL

Ocorre quando o comando é alfanumérico e a variável associada é numérica ou vice-versa. Ex.: LET A$ = 5.

STRING LONGO

Ocorre quando é criado um string com mais de 255 caracteres.

FORMULA COMPLEXA

Ocorre quando se usam muitas funções como argumento de funções numa mesma operação.

NAO CONTINUA

Ocorre quando se deseja continuar um programa inexistente ou que foi interrompido por outro erro.

FUNCAO INEXISTE

Ocorre quando se referencia uma função não definida anteriormente.

PAROU EM

A interrupção do programa através de control-C é entendida pelo computador como uma ocorrência de erro.

Dissemos que o comando BASIC ONERR GOTO permite a criação de rotinas de tratamento de erro, sem que o programa seja interrompido. Se o comando ONERR GOTO está presente, cada erro envia ao endereço 222 (decimal) da memória um código específico. O conteúdo dessa posição da memória pode ser lido, através de um PEEK(222), e associado a qualquer variável numérica. Ex.: ERRO = PEEK(222). Através do código fornecido pela leitura, obtém-se o tipo de erro ocorrido.

A seguir, você tem uma lista dos códigos e suas interpretações:

CÓDIGO	MENSAGEM
0	NEXT SEM FOR
16	SINTAX
22	RETURN SEM GOSUB
42	FALTA DADOS
53	VALOR ILEGAL
69	ESTOURO
77	SEM MEMORIA
90	COMANDO INEXISTE
107	INDICE ILEGAL
120	REDIM ARRAY
133	DIVISAO POR ZERO
149	COMANDO ILEGAL
163	INCOMPATIVEL
176	STRING LONGO
191	FORMULA COMPLEXA
210	NAO CONTINUA
224	FUNCAO INEXISTE
255	PAROU EM (control-C)

APÊNDICE C

ECONOMIA DE ESPAÇO E DE TEMPO

Aqui são apresentadas algumas sugestões referentes a procedimentos que o usuário pode adotar para compactar o uso da memória pelo seu programa:

1) Use linhas de múltiplos comandos, de no máximo 239 caracteres. Recomenda-se praticar este expediente após o programa ter sido testado, já que a edição pode resultar difícil, se for encontrado algum erro.

   Ex.: FOR I = 1 TO 10Ø: PRINT "TK //e" :NEXT

2) Apague todos os comandos REM do programa.
   Recomenda-se guardar uma versão do programa com os REMs para facilitar a documentação.

3) Use, se possível, matrizes de números inteiros no lugar de números reais.

4) Use variáveis no lugar de constantes.

5) Use o comando END somente para separar o programa de sub-rotinas. Não o use no fim dum programa. A utilização deste comando nos exemplos dados neste manual tem caráter puramente didático.

6) Reutilize, se possível, as mesmas variáveis dentro de um programa, desde que isto não implique em conflitos de lógica (ao reutilizar uma variável, certifique-se de que o seu conteúdo atual não é mais necessário ou se ela vai ser modificada em outra parte do programa mais à frente).

7) Procure criar sub-rotinas para áreas de programa que executarem tarefas similares.

8) Utilize os elementos zero das matrizes como, por exemplo, A(Ø), B(Ø,Y).

9) Quando A$ = "COL" é redefinido por A$ = "AP", o antigo string "COL" não é apagado na memória, embora não seja mais acessível. Se periodicamente se usa uma instrução do tipo X= FRE(Ø) dentro do programa, o BASIC "limpa" antigos conteúdos do topo da memória.

Agora algumas recomendações para aumentar a velocidade de execução do BASIC:

1) Use variáveis no lugar de constantes.

2) Defina no início do programa as variáveis mais usadas. Assim, diminui-se o tempo de procura.

3) Use NEXT sem a variável índice. Com isso, o processador não efetua a procura.

4) Coloque no início do programa as linhas mais referenciadas por GOTOs ou GOSUB.

APÊNDICE D

CÓDIGOS DOS COMANDOS

128	END	129	FOR	130	NEXT
131	DATA	132	INPUT	133	DEL
134	DIM	135	READ	136	GR
137	TEXT	138	PR#	139	IN#
140	CALL	141	PLOT	142	HLIN
143	VLIN	144	HGR2	145	HGR
146	HCOLOR	147	HPLOT	148	DRAW
149	XDRAW	150	HTAB	151	HOME
152	ROT=	153	SCALE=	154	SHLOAD
155	TRACE	156	NOTRACE	157	NORMAL
158	INVERSE	159	SOUND	160	COLOR
161	POP	162	VTAB	163	HIMEM:
164	LOMEM:	165	ONERR	166	RESUME
167	RECALL	168	STORE	169	SPEED=
170	LET	171	GOTO	172	RUN
173	IF	174	RESTORE	175	&
176	GOSUB	177	RETURN	178	REM
179	STOP	180	ON	181	WAIT
182	LOAD	183	SAVE	184	DEF
185	POKE	186	PRINT	187	CONT
188	LIST	189	CLEAR	190	GET
191	NEW	192	TAB	193	TO
194	FN	195	SPC	196	THEN
197	AT	198	NOT	199	STEP
200	+	201	-	202	*
203	/	204	^	205	AND
206	OR	207	>	208	=
209	<	210	SGN	211	INT
212	ABS	213	USR	214	FRE
215	SCRN	216	PDL	217	POS
218	SQR	219	RND	220	LOG
221	EXP	222	COS	223	SIN
224	TAN	225	ATN	226	PEEK
227	LEN	228	STR$	229	VAL
230	ASC	231	CHR$	232	LEFT$
233	RIGHT$	234	MID$		

APÊNDICE E

PALAVRAS RESERVADAS

Estas palavras reservadas são palavras que o BASIC reconhece para uso exclusivo:

ABS	AND	ASC	AT	ATN	
CALL	CHR$	CLEAR	COLOR	CONT	COS
DATA	DEF	DEL	DIM	DRAW	
&	END	EXP			
FN	FOR	FRE	FLASH		
GET	GOTO	GOSUB	GR		
HCOLOR	HGR	HGR2	HIMEM:	HLIN	HOME
HPLOT	HTAB				
IF	INPUT	INT	INVERSE	IN #	
LEFT$	LEN	LET	LIST	LN	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					
XDRAW	XPLOT				

APÊNDICE F

CÓDIGO ASCII

DEC	HEX	CHR$	DEC	HEX	CHR$	DEC	HEX	CHR$	DEC	HEX	CHR$
Ø	ØØ	NUL	32	2Ø	SP	64	4Ø	@	96	6Ø	*
1	Ø1	SOH	33	21	!	65	41	A	97	61	a
2	Ø2	STX	34	22	"	66	42	B	98	62	b
3	Ø3	ETX	35	23	#	67	43	C	99	63	c
4	Ø4	EOT	36	24	$	68	44	D	1ØØ	64	d
5	Ø5	ENQ	37	25	%	69	45	E	1Ø1	65	e
6	Ø6	ACK	38	26	&	7Ø	46	F	1Ø2	66	f
7	Ø7	BEL	39	27	'	71	47	G	1Ø3	67	g
8	Ø8	BS	4Ø	28	(	72	48	H	1Ø4	68	h
9	Ø9	HT	41	29	)	73	49	I	1Ø5	69	i
1Ø	ØA	LF	42	2A	*	74	4A	J	1Ø6	6A	j
11	ØB	VT	43	2B	+	75	4B	K	1Ø7	6B	k
12	ØC	FF	44	2C	,	76	4C	L	1Ø8	6C	l
13	ØD	CR	45	2D	-	77	4D	M	1Ø9	6D	m
14	ØE	SO	46	2E	.	78	4E	N	11Ø	6E	n
15	ØF	SI	47	2F	/	79	4F	O	111	6F	o
16	1Ø	DLE	48	3Ø	O	8Ø	5Ø	P	112	7Ø	p
17	11	DC1	49	31	1	81	51	Q	113	71	q
18	12	DC2	5Ø	32	2	82	52	R	114	72	r
19	13	DC3	51	33	3	83	53	S	115	73	s
2Ø	14	DC4	52	34	4	84	54	T	116	74	t
21	15	NAK	53	35	5	85	55	U	117	75	u
22	16	SYN	54	36	6	86	56	V	118	76	v
23	17	ETB	55	37	7	87	57	W	119	77	w
24	18	CAN	56	38	8	88	58	X	12Ø	78	x
25	19	EM	57	39	9	89	59	Y	121	79	y
26	1A	SUB	58	3A	:	9Ø	5A	Z	122	7A	z
27	1B	ESC	59	3B	;	91	5B	[	123	7B	{
28	1C	FS	6Ø	3C	<	92	5C	'	124	7C	~
29	1D	GS	61	3D	=	93	5D	]	125	7D	|
3Ø	1E	RS	62	3E	>	94	5E	^	126	7E	~
31	1F	US	63	3F	?	95	5F	_	127	7F	DEL

CARACTERES ACENTUADOS EM MODO PORTUGUÊS

CÓDIGO DECIMAL	CÓDIGO HEXA	CARACTERES	CÓDIGO DECIMAL	CÓDIGO HEXA	CARACTERES
35	23	õ	95	5F	à
38	26	ê	96	6Ø	é
64	4Ø	á	123	7B	í
91	5B	ã	124	7C	ô
92	5C	â	125	7D	ú
93	5D	ç	126	7E	ó

CÓDIGO ASCII GERADOS PELAS TECLAS E/OU COMBINAÇÃO DE TECLAS

TECLA	MINÚSCULAS	TECLA + CONTROL	TECLA + SHIFT	TECLA + CTRL + SHIFT
]}	5D	1D	7D	1D
~	60	60	7E	7E
A	61	01	41	01
B	62	02	42	02
C	63	03	43	03
D	64	04	44	04
E	65	05	45	05
F	66	06	46	06
G	67	07	47	07
H	68	08	48	08
I	69	09	49	09
J	6A	0A	4A	0A
K	6B	0B	4B	0B
L	6C	0C	4C	0C
M	6D	0D	4D	0D
N	6E	0E	4E	0E
O	6F	0F	4F	0F
P	70	10	50	10
Q	71	11	51	11
R	72	12	52	12
S	73	13	53	13
T	74	14	54	14
U	75	15	55	15
V	76	16	56	16
W	77	17	57	17
X	78	18	58	18
Y	79	19	59	19
Z	7A	1A	5A	1A

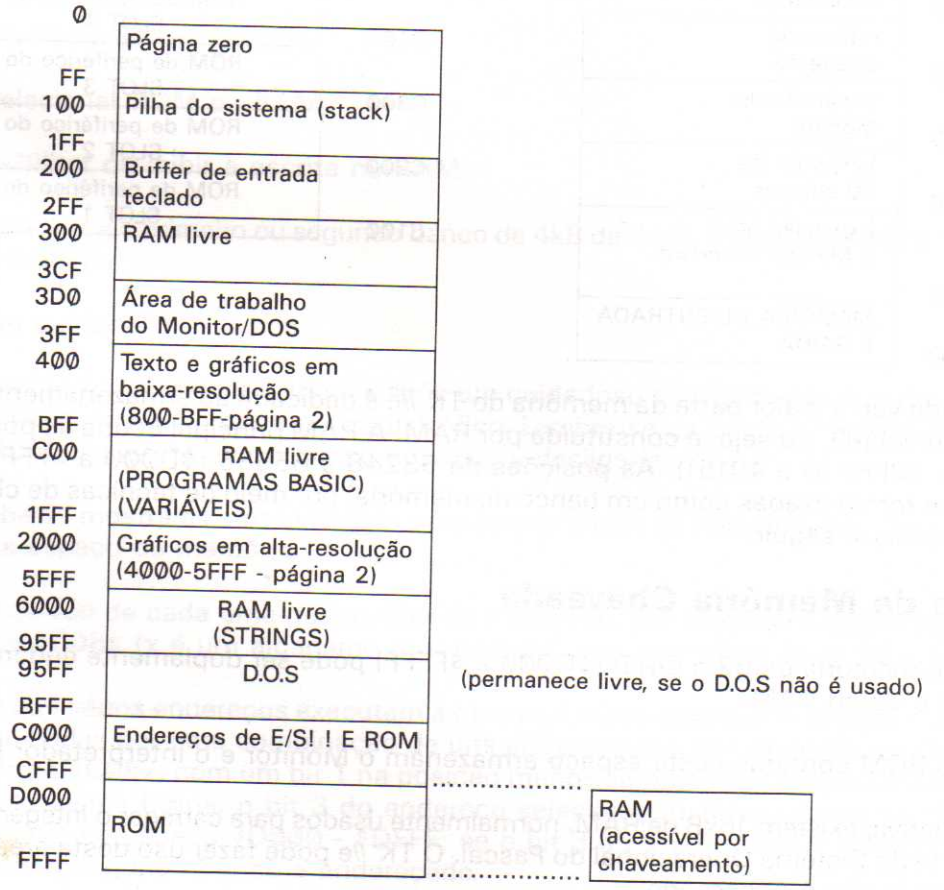
TECLA	NORMAL	TECLA + CONTROL	TECLA + SHIFT	TECLA + CTRL + SHIFT
DELETE	7F	7F	7F	7F
SETA-ESQUERDA	08	08	08	08
TAB	09	09	09	09
SETA-ABAIXO	0A	0A	0A	0A
SETA-ACIMA	0B	0B	0B	0B
RETURN	0D	0D	0D	0D
SETA-DIREITA	15	15	15	15
ESC	1B	1B	1B	1B
space	20	20	20	20
' "	27	27	22	22
, <	2C	2C	3C	3C
- _	2D	2D	5F	1F
. >	2E	2E	3E	3E
/ ?	2F	2F	3F	3F
0)	30	30	29	29
1 !	31	31	21	21
2 @	32	00	40	00
3 #	33	33	23	23
4 $	34	34	24	24
5 %	35	35	25	25
6 ^	36	1E	5E	1E
7 &	37	37	26	26
8 *	38	38	2A	2A
9 (	39	39	28	28
; :	3B	3B	3A	3A
= +	3D	3D	2B	2B
[{	5B	1B	7B	1B
' |	5C	1C	7C	1C

APÊNDICE G

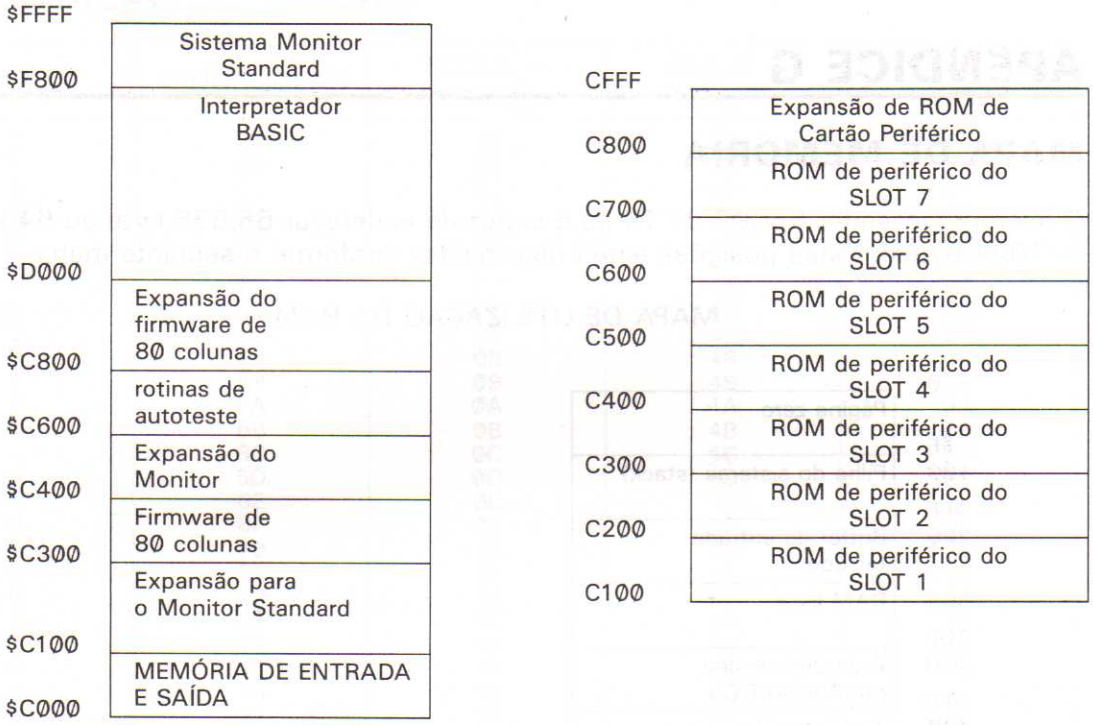
MAPA DE MEMÓRIA

O microprocessador 65CØ2 do TK //e é capaz de endereçar 65.536 byte ou 64 kB (1kB = 1Ø24 bytes). Estas posições estão distribuídas conforme o seguinte mapa:

MAPA DE UTILIZAÇÃO DA RAM



MAPA DA ROM



Como se pode ver, a maior parte da memória do TK //e é dedicada ao armazenamento programável (ou volátil), ou seja, é constituída por RAM. A RAM principal ocupa as posições de $0000 a $BFFF (0 a 49151). As posições de 53248 a 65535 ($D000 a $FFFF) são acessíveis se forem usadas como um banco de memória, por meio de técnicas de chaveamento, descritas a seguir.

1. Banco de Memória Chaveada

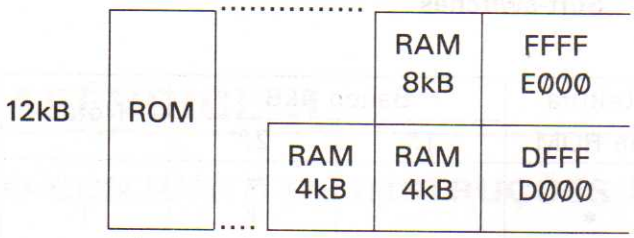
O espaço de memória de 52 a 64kB ($D000 a $FFFF) pode ser duplamente endereçado: como ROM e como RAM.

Os 12kB da ROM contidos neste espaço armazenam o Monitor e o interpretador BASIC.

Como alternativa, existem 16kB de RAM, normalmente usados para carregar o Integer BASIC ou uma parte do Sistema Operacional do Pascal. O TK //e pode fazer uso desta área quando o Applesoft não é necessário.

Diferentes áreas de memória, lidas pelos mesmos endereços, formam o que se chama de "banco de memória chaveada".

Pode parecer estranho que 16k de RAM possam ocupar o espaço de 12k. Isso acontece porque os últimos 4k são usados duplamente, como é mostrado a seguir:



(8kB + 4kB + 4kB)

2. Acessando o Banco de Memória

Pode-se dar acesso a um banco de memória por meio de "soft-switches" (determinados endereços que, referenciados, permitem a execução de certas rotinas).

As "soft-switches" podem:

a) selecionar ROM ou RAM;

b) habilitar ou inibir a escrita na RAM;

c) selecionar o primeiro ou segundo banco de 4kB da RAM, no espaço endereçado de $0000 a $DFFF.

Muito cuidado!

Use o banco de memória apenas após um cuidadoso planejamento. Um descuido no chaveamento do banco pode ser catastrófico. Lembre-se de que o interpretador BASIC está na ROM e que chavear o banco equivale a desligá-la.

A tabela I mostra as soft-switches para formar todas as combinações de escrita e leitura neste espaço de memória.

O endereço de cada uma destas chaves é formado por valores hexadecimais escritos na forma $C08x (x é um algarismo hexadecimal, compreendido entre $0 e $F).

Note que vários endereços executam a mesma função. Isto ocorre porque as soft-switches são ativadas por uma combinação de bits endereçáveis. Por exemplo, qualquer endereço da forma $C08x, com um bit 1 na posição menos significativa, habilita a RAM para escrita. Da mesma forma, o bit 3 do endereço seleciona qual dos blocos de 4kB da RAM é usado para o espaço $D000 - $DFFF; se o bit 3 é zero, o primeiro banco é endereçado, se não, o segundo banco é endereçado.

TABELA I - Soft-switches

Endereço	Escrita na RAM	Leitura na RAM	Leitura na ROM	Banco 4kB		Notas
				1.º	2.º	
$C080		*			*	
$C081	*		*		*	1
$C082			*		*	
$C083	*	*			*	2
$C084		*			*	
$C085	*		*		*	1
$C086			*		*	
$C087	*	*			*	2
$C088		*		*		
$C089	*		*	*		1
$C08A			*	*		
$C08B	*	*		*		2
$C08C		*		*		
$C08D	*		*	*		1
$C08E			*	*		
$C08F	*	*		*		2

1) Uma leitura desta posição habilita a escrita na RAM e a leitura na ROM.

2) Duas acessos de leitura sucessivas destas posições habilitam a RAM para leitura e escrita.

Quando a RAM não está habilitada para leitura, a ROM está. Mesmo que a RAM esteja com a leitura bloqueada, a escrita é possível, desde que esteja habilitada.

Ao ser ligado (ou após um reset), o TK //e é inicializado de forma a ler a ROM e escrever na RAM, usando o segundo banco de RAM. Assim, se as teclas control reset são acionadas, perde-se o controle sobre o programa em execução.

Se está sendo usado o Integer BASIC, o reset não causa transtornos, pois o controle passa para o DOS e este seleciona a versão do BASIC que está sendo utilizada.

Se o banco de memória está sendo usado para escrita e leitura, não se pode usar a ROM. Todavia, muitas vezes é necessário usar o Monitor Disassembler. A solução é transferir o conteúdo da parte da ROM dedicada ao Monitor para qualquer área de RAM disponível; habilitar o banco para escrita; e transferir o Monitor da área de RAM para o banco, no mesmo local onde estava armazenado antes (o monitor está contido entre os endereços $F800 até $FFCB). Assim, também é possível habilitar este banco para leitura, perdendo-se o controle da ROM, mas com o programa monitor disponível em RAM.

APÊNDICE H

O CONJUNTO DE INSTRUÇÕES DO 65C02

Usamos as seguintes notações:

A	Acumulador
X,Y	Registradores de índice
M	Memória
C	Carry
B	Borrow
P	Registrador de estados do processador
S	Ponteiro da pilha
@	Conforme resultado da operação
+	Soma
^	E lógico
-	Subtrai
%	Ou exclusivo
*	Retira da pilha
$	Transfere para a pilha
←	Transfere para
→	Transfere para
V	Ou lógico
PC	Contador de programa
PCH	Byte mais significativo do contador de programa
PCL	Byte menos significativo do contador de programa
OPER	Operando
#	Modo de endereçamento imediato

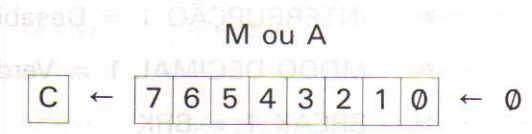


Figura 1 — deslocamento de um bit à esquerda

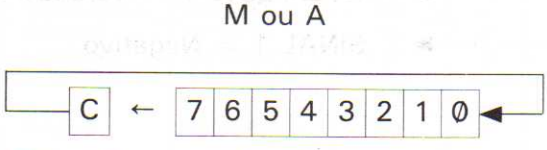


Figura 2 — rotação de um bit à direita

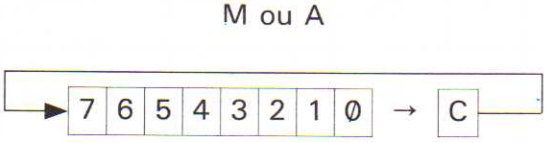
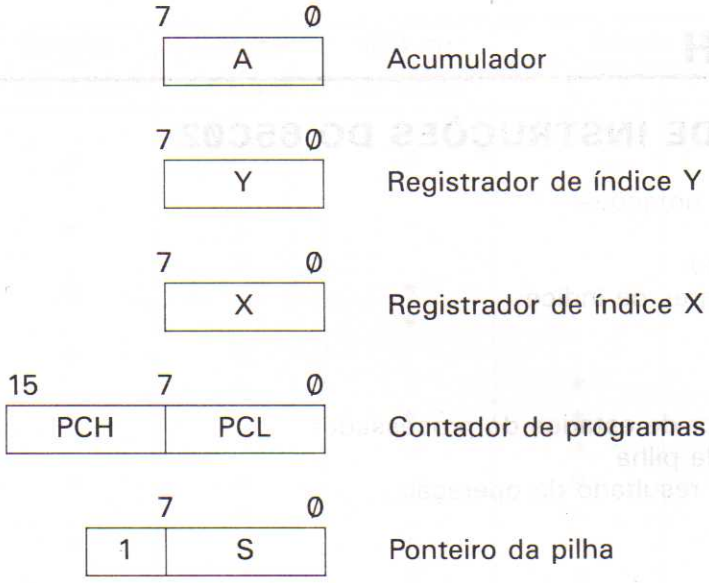
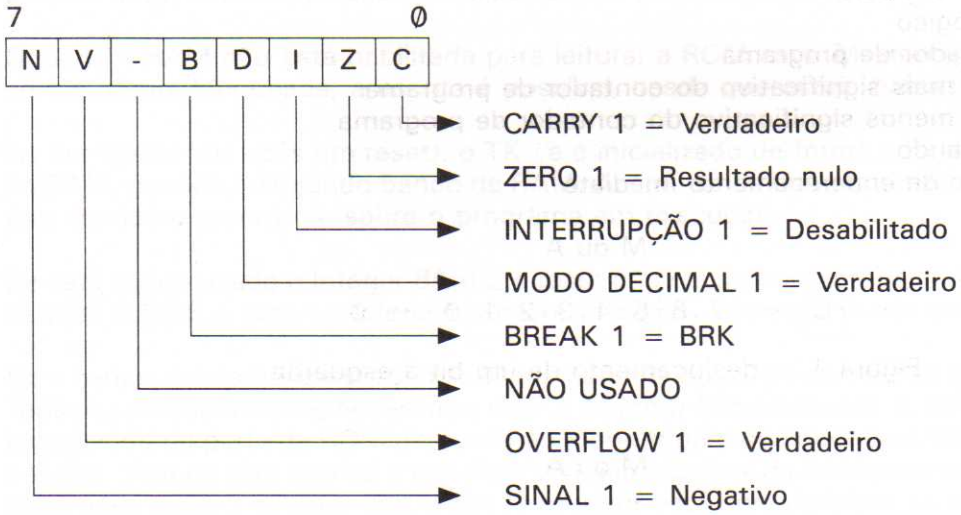


Figura 3 — rotação de um bit à esquerda

MODELO DO 65C02 PARA PROGRAMAÇÃO



Registrador de Estados do Processador



O CÓDIGOS DE OPERAÇÃO

NOME DESCRIÇÃO	OPERAÇÃO	MODOS DE ENDEREÇAMENTO	MNEMÔNICOS (ASSEMBLY)	CÓDIGO DE OPERAÇÃO	BYTES	REGISTRADORES DE ESTADOS (P) N V B D I Z C	CICLOS	NOTA
ADC Adiciona memória com acumulador com carry	A+M+C→A					@ @ - - - @ @		
		Imediato	ADC #OPER	69	2		2	
		Página Zero	ADC OPER	65	2		3	
		Página Zero,X	ADC OPER,X	75	2		4	
		Indireto da Página zero	ADC (OPER)	72	2		5	(3)
		Absoluto	ADC OPER	6D	3		4	
		Absoluto,X	ADC OPER,X	7D	3		4	
		Absoluto,Y	ADC OPER Y	79	3		4	
		(Indireto,X)	ADC (OPER,X)	61	2		6	
		(Indireto),Y	ADC (OPER),Y	71	2		5	
AND Faz operação "E" da memória com o acumulador	A^M→A					@ - - - - @ -		
		Imediato	AND #OPER	29	2		2	
		Página Zero	AND OPER	25	2		3	
		Página Zero,X	AND OPER,X	35	2		4	
		Indireto da Página zero	AND (OPER)	32	2		5	(3)
		Absoluto	AND OPER	2D	3		4	
		Absoluto.X	AND OPER,X	3D	3		4	
		Absoluto.Y	AND OPER.Y	39	3		4	
		(Indireto,X)	AND (OPER.X)	21	2		6	
		(Indireto),Y	AND (OPER),Y	31	2		5	
ASL Desloca um bit à esquerda (Memória ou acumulador)	ver fig.1					@ - - - - @ @		
		Acumulador	ASL A	ØA	1		2	
		Página Zero	ASL OPER	Ø6	2		5	
		Página Zero,X	ASL OPER,X	16	2		6	
		Absoluto	ASL OPER	ØE	3		6	
		Absoluto,X	ASL OPER,X	1E	3		6	(4)
BCC Salta se CARRY = Ø	Salta se C=Ø	Relativo	BCC OPER	9Ø	2	- - - - - - -	2	
BCS Salta se carry=1	Salta se C=1	Relativo	BCS OPER	BØ	2	- - - - - - -	2	
BEQ Salta se o resultado for Ø	Salta se Z=1	Relativo	BEQ OPER	FØ	2	- - - - - - -	2	
BIT Testa bits da memória com o acumulador	A^M N←M7 V←M6					M 7 M 6 - - - - -		
		Imediato	BIT # OPER	89	2		2	(1)
		Página Zero	BIT OPER	24	2		3	(1)
		Página Zero,X	BIT OPER,X	34	2		4	(3)
		Absoluto	BIT OPER	2C	3		4	(1)
		Absoluto,X	BIT OPER,X	3C	3		4	(3)
BMI Salta se o resultado for negativo	Salta se N=1	Relativo	BMI OPER	3Ø	2	- - - - - - -	2	
BNE Salta se não for igual a Ø	Salta se Z=Ø	Relativo	BNE OPER	DØ	2	- - - - - - -	2	
BPL Salta se o resultado for negativo	Salta se N=Ø	Relativo	BPL OPER	1Ø	2	- - - - - - -	2	

NOME DESCRIÇÃO	OPERAÇÃO	MODOS DE ENDEREÇAMENTO	MNEMÔNICOS (ASSEMBLY)	CÓDIGO DE OPERAÇÃO	BYTES	REGISTRADORES DE ESTADOS (P) N V B D I Z C	CICLOS	NOTA
BRA Salta sempre	Salto Obrigatório	Relativo	BRA OPER	80	2	- - - - - - -	2	(3)(4)
BRK Parada	Força uma interrupção PC+2→P	Implícito	BRK	00	1	- - 1 - 1 - -	7	(2)
BVC Salta se não ocorreu overflow	Salta se V=0	Relativo	BVC OPER	50	2	- - - - - - -	2	
BVS Salta se ocorreu overflow	Salta se V=1	Relativo	BVS OPER	70	2	- - - - - - -	2	
CLC Limpa flag de carry	0→C	Implícito	CLC	18	1	- - - - - - 0	2	
CLD Desabilita modo decimal	0→D	Implícito	CLD	D8	1	- - - 0 - - -	2	
CLI Desabilita interrupção	0→I	Implícito	CLI	58	1	- - - - 0 - -	2	
CLV Limpa flag de overflow	0→V	Implicito	CLV	B8	1	- 0 - - - - -	2	
CMP Compara memória com acumulador	A-M					@ - - - - @ @		
		Imediato	CMP #OPER	C9	2		2	
		Página Zero	CMP OPER	C5	2		3	
		Página Zero,X	CMP OPER,X	D5	2		4	
		Indireto da Página zero	CMP (OPER)	D2	2		5	(3)
		Absoluto	CMP OPER	CD	3		4	
		Absoluto,X	CMP OPER,X	DD	3		4	
		Absoluto,Y	CMP OPER,Y	D9	3		4	
		(Indireto,X)	CMP (OPER,X)	C1	2		6	
		(Indireto),Y	CMP (OPER),Y	D1	2		5	
CPX Compara memória com registrador X	X-M					@ - - - - @ @		
		Imediato	CPX #OPER	E0	2		2	
		Página Zero	CPX OPER	E4	2		3	
		Absoluto	CPX OPER	EC	3		4	
CPY compara memória com registrador Y	Y-M					@ - - - - @ @		
		Imediato	CPY #OPER	C0	2		2	
		Página Zero	CPY OPER	C4	2		3	
		Absoluto	CPY OPER	CC	3		4	
DEA Decrementa acumulador	A←A - 1	Acumulador	DEA	3A	2	@ - - - - @ -	2	(3)
DEC Decrementa memória de uma unidade	M←M-1					@ - - - - @ -		
		Página Zero	DEC OPER	C6	2		5	
		Página Zero X	DEC OPER,X	D6	2		6	
		Absoluto	DEC OPER	CE	3		6	
		Absoluto,X	DEC OPER,X	DE	3		6	(4)

NOME DESCRIÇÃO	OPERAÇÃO	MODOS DE ENDEREÇA-MENTO	MNEMÔNICOS (ASSEMBLY)	CÓDIGO DE OPERAÇÃO	BYTES	REGISTRADORES DE ESTADOS (P) N V B D I Z C	CICLOS	NOTA
DEX Decrementa registrador X de uma unidade	X←X-1	Implícito	DEX	CA	1	@ - - - - @ -	2	
DEY Decrementa registrador Y de uma unidade	Y←Y-1	Implícito	DEY	88	1	@ - - - - @ -	2	
EOR "Ou-exclusivo" da memória com o acumulador	A←A % M	Imediato	EOR #OPER	49	2	@ - - - - @ -	2	
		Página Zero	EOR OPER	45	2		3	
		Página Zero,X	EOR OPER,X	55	2		4	
		Indireto da Página zero	EOR (OPER)	52	2		5	(3)
		Absoluto	EOR OPER	4D	3		4	
		Absoluto,X	EOR OPER,X	5D	3		4	
		Absoluto,Y	EOR OPER,Y	59	3		4	
		(Absoluto,X)	EOR (OPER,X)	41	2		6	
		(Absoluto),Y	EOR (OPER),Y	51	2		5	
INA Incrementa Acumulador	A←A + 1	Acumulador	INA	1A	1	@ - - - - @ -	2	(3)
INC Incrementa memória de uma unidade	M←M+1	Página Zero	INC OPER	E6	2	@ - - - - @ -	5	
		Página Zero,X	INC OPER,X	F6	2		6	
		Absoluto	INC OPER	EE	3		6	
		Absoluto,X	INC OPER,X	FE	3		6	(4)
INX Incrementa X de uma unidade	X←X-1	Implícito	INX	E8	1	@ - - - - @ -	2	
INY Incrementa Y de uma unidade	Y←Y+1	Implícito	INY	C8	1	@ - - - - @ -	2	
JMP Salta para outro endereço	PCL←M PCH←M+1	Absoluto	JMP OPER	4C	3	- - - - - - -	3	
		Indireto	JMP (OPER)	6C	3		6	
		Absoluto (Indireto,X)	JMP OPER	7C	3		6	(3)
JSR Salta para uma sub-rotina	PC+2$ PCL←OPERL PCH←OPERH	Absoluto	JSR OPER	2Ø		- - - - - - -	6	
LDA Carrega A com a memória	A←M	Imediato	LDA #OPER	A9	2	@ - - - - @ -	2	
		Página Zero	LDA OPER	A5	2		3	
		Página Zero,X	LDA OPER,X	B5	2		4	
		Indireto da Página zero	LDA (OPER)	B2	2		5	(3)
		Absoluto	LDA OPER	AD	3		4	
		Absoluto,X	LDA OPER,X	BD	3		4	
		Absoluto,Y	LDA OPER,Y	B9	3		4	
		(Indireto,X)	LDA (OPER,X)	A1	2		6	
		(Indireto),Y	LDA (OPER),Y	B1	2		5	

NOME DESCRIÇÃO	OPERAÇÃO	MODOS DE ENDEREÇAMENTO	MNEMÔNICOS (ASSEMBLY)	CÓDIGO DE OPERAÇÃO	BYTES	REGISTRADORES DE ESTADOS (P) N V B D I Z C	CICLOS	NOTA
LDX Carrega X com a memória	X←M	Imediato Página Zero Página Zero,Y Absoluto Absoluto,Y	LDX #OPER LDX OPER LDX OPER,Y LDX OPER LDX OPER,Y	A2 A6 B6 AE BE	2 2 2 3 3	@ - - - - @ -	2 3 4 4 4	
LDY Carrega Y com a memória	Y←M	Imediato Página Zero Página Zero,X Absoluto Absoluto,X	LDY #OPER LDY OPER LDY OPER,X LDY OPER LDY OPER,X	A0 A4 B4 AC BC	2 2 2 3 3	@ - - - - @ -	2 3 4 4 4	
LSR Desloca um bit à direita (memória e acumulador)	Veja fig.1	Acumulador Página Zero Página Zero,X Absoluto Absoluto,X	LSR A LSR OPER LSR OPER,x LSR OPER LSR OPER,X	4A 46 56 4E 5E	1 2 2 3 3	0 - - - - @ @	2 5 6 6 6	 (4)
NOP Nenhuma operação é executada	Nada é feito	Implícito	NOP	EA	1	- - - - - - -	2	
ORA Faz a operação lógica "OU" entre o acumulador e memória	A←A V M	Imediato Página Zero Página Zero,X Página Zero, Indireto Absoluto Absoluto,X Absoluto,Y (Indireto,X) (Indireto),Y	ORA #OPER ORA OPER ORA OPER,X ORA OPER ORA OPER ORA OPER,X ORA OPER,Y ORA (OPER,X) ORA (OPER),Y	09 05 15 12 0D 1D 19 01 11	2 2 2 2 3 3 3 2 2	@ - - - - @ -	2 3 4 5 4 4 4 6 5	
PHA Coloque o acumulador na pilha	$←A S-1→A	Implícito	PHA	48	1	- - - - - - -	3	
PHP Coloque P na pilha	$←P S-1→S	Implícito	PHP	08	1	- - - - - - -	3	
PLA Retire o acumulador da pilha	A←* S+1→S	Implícito	PLA	68	1	@ - - - - @ -	4	
PLP Retire P da pilha	P←* S+1→S	Implícito	PLP	28	1	RESTAURADO Porém B=1	4	
PLX Retira X da pilha	X←* S+1→S	Implícito	PLX	FA	1	- - - - - - -	4	(3)
PLY Retira Y da Pilha	Y←* S+1→S	Implícito	PLY	7A	1	- - - - - - -	4	(3)

NOME DESCRIÇÃO	OPERAÇÃO	MODOS DE ENDEREÇA-MENTO	MNEMÔNICOS (ASSEMBLY)	CÓDIGO DE OPERAÇÃO	BYTES	REGISTRADORES DE ESTADOS (P) N V B D I Z C	CICLOS	NOTA
ROL Rotaciona um bit à esquerda (memória e acumulador)	Veja fig.2					@ - - - - @ @		
		Acumulador	ROL A	2A	1		2	
		Página Zero	ROL OPER	26	2		5	
		Página Zero,X	ROL OPER X	36	2		6	
		Absoluto	ROL OPER	2E	3		6	
		Absoluto,X	ROL OPER,X	3E	3		6	(4)
ROR Rotaciona um bit à direita (memória e acumulador)	Veja fig.3					@ - - - - @ @		
		Acumulador	ROR A	6A	1		2	
		Página Zero	ROR OPER	66	2		5	
		Página Zero,X	ROR OPER,X	76	2		6	
		Absoluto	ROR OPER	6E	3		6	
		Absoluto,X	ROR OPER,X	7E	3		6	(4)
RTI Retorno de uma interrupção	P← * PC← *	Implícito	RTI	4Ø	1	RESTAURADO Porém B=1	6	
RTS Retorno de uma sub-rotina	PC$ PC←PC+1	Implícito	RTS	6Ø	1	- - - - - - -	6	
SBC Subtrai memória do acumulador com borrow	A←A-M-B					@ @ - - - @ @		
		Imediato	SBC #OPER	E9	2		2	
		Página Zero	SBC OPER	E5	2		3	
		Página Zero,X	SBC OPER,X'	F5	2		4	
		Indireto da Página zero	SBC OPER	F2	2		5	(3)
		Absoluto	SBC OPER	ED	3		4	
		Absoluto,X	SBC OPER,X	FD	3		4	
		Absoluto,Y	SBC OPER,Y	F9	3		4	
		(Indireto,X)	SBC (OPER,X)	E1	2		6	
		(Indireto),Y	SBC (OPER),Y	F1	2		5	
SEC Coloca um no carry	C←1	Implícito	SEC	38	1	- - - - - - 1	2	
SED Habilita modo decimal	D←1	Implícito	SED	F8	1	- - - 1 - - -	2	
SEI Habilita interrupção	1←I	Implícito	SEI	78	1	- - - - 1 - -	2	
STA Armazena acumulador na memória	A→M					- - - - - - -		
		Página Zero	STA OPER	85	2		3	
		Página Zero,X	STA OPER,X	95	2		4	
		Indireto da Página zero	STA OPER	92	2		5	(3)
		Absoluto	STA OPER	8D	3		4	
		Absoluto,X	STA OPER,X	9D	3		5	
		Absoluto,Y	STA OPER,Y	99	3		5	
		(Indireto,X)	STA (OPER,X)	81	2		6	
		(Indireto),Y	STA (OPER,,Y	91	2		6	
STX Armazena X na memória	X→M					- - - - - - -		
		Página Zero	STX OPER	86	2		3	
		Página Zero,Y	STX OPER,Y	96	2		4	
		Absoluto	STX OPER	8E	3		4	

NOME DESCRIÇÃO	OPERAÇÃO	MODOS DE ENDEREÇA-MENTO	MNEMÔNICOS (ASSEMBLY)	CÓDIGO DE OPERAÇÃO	BYTES	REGISTRADORES DE ESTADOS (P) N V B D I Z C	CICLOS	NOTA
STY Armazena Y na memória	Y→M	 Página Zero Página Zero,X Absoluto	 STY OPER STY OPER,X STY OPER	 84 94 8C	 2 2 3	- - - - - - -	 3 4 4	
STZ Armazena zero na memória	ØØ→OR	 Página Zero Página Zero,X Absoluto Absoluto,X	 STZ OPER STZ OPER,X STZ OPER STZ OPER,X	 64 74 9C 9E	 2 2 3 3	- - - - - - -	 3 4 4 5	 (3) (3) (3) (3)
TAX Transfere A para X	A→X	Implícito	TAX	AA	1	@ - - - - @ -	2	
TAY Transfere A para Y	A→Y	Implícito	TAY	A8	1	@ - - - - @ -	2	
TRB Testa e reseta bits memória com acumulador	$\overline{A}$^M→M	 Página Zero Absoluto	 TRB OPER TRB OPER	 14 1C	 2 3	M 7 M 6 - - - @ -	 5 6	(1) (3) (3)
TSB Testa e seta bits memória e acumulador	AVM→M	 Página Zero Absoluto	 TSB OPER TSB OPER	 Ø4 ØC	 2 3	M 7 M 6 - - - @ -	 5 6	(1) (3) (3)
TSX Transfere o ponteiro da pilha X	S→X	Implícito	TSX	BA	1	@ - - - - @ -	2	
TXA Transfere X para o acumulador	X→A	Implícito	TXA	8A	1	@ - - - - @ -	2	
TXS Transfere X para o ponteiro da pilha	X→S	Implícito	TXS	9A	1	- - - - - - -	2	
TYA Transfere Y para o acumulador	Y→A	Implícito	TYA	98	1	@ - - - - @ -	2	

1 - Os bits 6 e 7 são transferidos para o registrador de estados. Se o resultado for zero, Z = 1.

2 - O comando BRK não pode ser mascarado colocando-se 1 no flag I.

3 - Exclusivo do 65CØ2.

4 - O número de ciclos de máquina indicado são para CPU NCR65CØ2; para CPU ROCKWELL 65CØ2, acrescente 1 ao número de ciclos indicado.

MODOS DE ENDEREÇAMENTO DO 65CØ2

O 65CØ2 possui quinze modos de endereçamento, que permitem o acesso a uma determinada posição de memória. São eles:

Implícito: O endereço que contém o operando está implícito no código da instrução.

Acumulador: Indica que a operação ocorre no acumulador.

Imediato: O operando está no byte seguinte ao do código da instrução.

Endereçamento Absoluto: O segundo e terceiro bytes da instrução contêm respectivamente, o byte menos e byte mais significativo do endereço.

Página Zero : O segundo byte da instrução contém um endereço da página zero ($ØØ a $FF).

Indexado Absoluto : O endereço (endereço efetivo) é dado pela soma de endereço de dois bytes (endereço de base) com o conteúdo de um registrador de índices (X ou Y).

Indexado de Página Zero : O endereço (endereço efetivo) é dado pela soma de um endereço de página zero (endereço de base) com o conteúdo de um registrador de índices (X ou Y) - o carry não é considerado.

Relativo: O segundo byte da instrução é um número com sinal (-128 a +127) a ser somado ao byte menos significativo do contador de programas, quando este estiver apontando para a próxima instrução.

Indexado Indiretamente: O segundo byte da instrução é somado ao conteúdo do registrador de índices X. O CARRY não é considerado. O resultado desta operação aponta para uma nova localização da página zero que contém o byte menos significativo do endereço efetivo. A próxima localização da página zero contém o byte mais significativo.

(*) Absoluto Indexado indiretamente: O número formado pelo segundo byte (menos significativo) e o terceiro byte (mais significativo) da instrução é somado ao conteúdo do registrador de índices X - o carry é considerado. O resultado desta operação aponta para uma nova localização da memória que contém o byte menos significativo do endereço efetivo. A próxima localização da memória contém o byte mais significativo.

Indireto Indexado: O segundo byte da intrução aponta para um endereço da página zero. O seu conteúdo é somado ao conteúdo do registrador Y. O resultado desta operação vai ser o byte menos significativo do endereço efetivo. O carry desta adição é somado ao conteúdo da próxima localizaçao de memória da página zero. O resultado torna-se o byte mais significativo do endereço.

(*) Indireto da Página Zero: O segundo byte da instrução aponta para uma localização da página zero que contém o endereço do byte menos significativo do endereço efetivo. A localização seguinte contém o byte mais significativo.

Absoluto indireto (apenas em instruções JMP): O segundo byte da instrução contém o byte menos significativo de uma localização de memória. O terceiro byte contém o byte mais significativo. Este endereço contém o byte menos significativo do endereço efetivo. A localização seguinte contém o byte mais significativo. O endereço efetivo é carregado no contador de programa (registrador PC (16 bits)).

Nota: (*) Apenas para o 65CØ2.

GLOSSÁRIO

acesso - Processo de obtenção ou de armazenamento de dados, informações, funções, comandos, operações, etc.

acumulador - Um registrador do microprocessador, onde os dados são armazenados e processados.

alta-resolução gráfica - Modo do display em que o traçado das figuras compõe-se de pontos. No TK //e a alta-resolução pode ocupar até 28Ø pontos em cada linha e 192 linhas na tela, num total de 53.86Ø pontos. Mantendo-se uma janela de 4 linhas de texto, uma composição em alta-resolução pode formar-se de até 28Ø pontos X 16Ø linhas, num total de 44.8ØØ pontos. Apenas oito cores ficam disponíveis em alta-resolução.

apagar - Veja delete.

apontador - Ponteiro. Registro que contém o endereço de determinada localização da memória.

Applesoft - BASIC padrão da linha Appplu. * Este termo é uma marca registrada da Apple Computer Inc.

argumento - Valor ou valores que se associam a uma função.

armazenamento - Guarda dos dados e outras informações, em determinados endereços de memória.

array - Arranjo, conjunto ou matriz. Disposição ordenada de elementos em linhas e colunas.

assembler - Programa que traduz os códigos do Assembly para a linguagem de máquina.

baixa-resolução gráfica - Modo do display em que o traçado das figuras compõe-se de blocos de pontos. No TK //e a baixa-resolução pode ocupar até 4Ø colunas por 4Ø linhas, permanecendo as 4 últimas linhas da tela no modo texto. Dezesseis cores ficam disponíveis na baixa resolução.

banco de dados - Conjunto de dados armazenados sistematicamente, para facilitar a pesquisa e a atualização.

bit - Binary digit. Dígito binário. Unidade de informação, assumindo o valor Ø ou 1.

bloco - Conjunto de elementos considerados como uma unidade.

boot ou bootstrap - Autocarregador. Emprego de instruções preliminares para carregar um programa no computador, executadas automaticamente assim que o equipamento é ligado.

buffer - Área de memória de armazenamento temporário de informações e de dados, enquanto estes estão sendo transferidos de um local para outro do sistema.

bus - Canal ou linha através do qual se podem tranferir dados e outras informações.

byte - Unidade formada pelo conjunto de oito bits. Cada byte pode assumir um valor entre Ø e 255 (decimal).

call - Chamar. Requerer a execução de uma sub-rotina ou função.

caractere alfanumérico - Letra, número, símbolo, sinal de pontuação.

carregar - Transferir programas ou dados de uma fita cassete ou disquete para a memória do computador. O mesmo que ler ou fazer a leitura.

cartão - Veja interface.

chip - Circuito integrado, constituído de centenas ou milhares de transistores e diodos, montados numa plaqueta de silício de pequeno tamanho.

CATALOG - Catálogo. Lista dos arquivos de um disquete. O mesmo que diretório.

código binário - Sistema que se baseia nas correlações de dois fatores: 1 (positivo ou ligado) e Ø (negativo ou desligado).

coluna - Disposição na tela ou na impressora na direção vertical.

comando - (1) Controle de operações, em geral de entrada e saída.
(2) Qualquer instrução em linguagem de máquina.
(3) Ordens do BASIC que o computador executa de pronto (RUN, LIST, CONT).

compactar - Agregar elementos ou reuni-los em determinada disposição, em geral para economizar memória.

compatibilidade - (1) Capacidade do equipamento para manipular dados ou programas processados por outra máquina.
(2) Possibilidade de interconexão de equipamentos, quando eles possuem o mesmo tipo de portas de E/S.

compilar - Converter um programa escrito em linguagem de alto nível num programa em linguagem de máquina.

concatenação - União de dois ou mais strings formando um único string mais longo.

configuração - Conjunto de equipamentos interligados para operarem como um sistema.

control - Controle. No TK //e a tecla control modifica a atuação de outras teclas.

CP/M - Control Program for Microprocessors. Um tipo de sistema operacional.

CPU - Central Processing Unit. Unidade Central de Processamento. É o "cérebro" do computador, encarregando-se do controle da interpretação e execução das instruções.

cursor - Indicador móvel que mostra onde o próximo caractere pode ser impresso na tela.

dado - Valor, fórmula, conjunto de caracteres, etc. que tenha uma significação.

decimal - Sistema que tem como base os dez números de Ø a 9.

decodificar - (1) Interpretar uma informação codificada, traduzindo-a numa linguagem compreensível.
(2) Determinar o significado do código de operação de uma instrução ou de um conjunto de sinais.

delete - Apagar, limpar, suprimir, eliminar da tela, da memória, do disquete ou da fita cassete.

digitar - Pressionar as teclas com os dedos. Teclar.

dígito - (1) Símbolo de um sistema de numeração de base N. Por ex., na notação decimal, um dígito é um dos caracteres dentre Ø e 9. (2) Ação e efeito de digitar.

diretório - Veja CATALOG.

display - Visualização, apresentação visual dos sinais de imagem emitidos pelo computador.

disquete - Disco flexível. Elemento físico de uma unidade de armazenamento em discos.

dividendo - Valor a ser dividido.

divisor - Valor pelo qual se divide o dividendo.

DOS - Disk Operating System. Sistema operacional de disco. Software controlador das operações da(s) unidade(s) de disco.

editar - Rearranjar os dados através da sua alteração, seleção, supressão ou inserção.

endereçar - (1) Atribuir endereço.
(2) Enviar a um endereço de memória.

endereço - Número que identifica uma posição de armazenamento da memória.

entrada - (1) A tranferência de dados ou informações ao computador através de uma fonte externa como o teclado, unidade de disco, modem, gravador.
(2) O acesso a uma operação ou endereço de memória.

entrada/saída - Transferência de dados ou informações. Veja entrada e saída.

E/S - entrada/saída.

esc - Abreviatura de escape character.

escape - Caractere de controle que sinaliza uma mudança no significado do caractere ou dos caracteres que o seguem.

escrever - (1) Gravar, registrar dados em qualquer meio de armazenamento, como fita magnética, disquete.
(2) Digitar um programa.

expansão - Memória RAM adicional, obtida através de interface.

expoente - Valor ao qual a base é elevada.

formatar - (1) Preparar o disquete, dividindo-o em trilhas e setores, para que o sistema operacional localize os dados.
(2) Especificar ou controlar o arranjo e a apresentação dos dados.

gráfico - (1) Representação gráfica ou desenho de qualquer figura..
(2) Modo do display que habilita a saída de imagem gráfica - veja texto.

gravar - Registrar os programas e dados, passando-os do computador para a fita magnética ou disquete.

hardware - Suporte físico (eletrônico ou mecânico).

hexadecimal - Sistema numérico de base 16, que utiliza os dígitos de Ø a 9 e A, B, C, D, E e F. Cada dígito hexadecimal corresponde a quatro bits.

imprimir - Fazer aparecer na tela ou no papel da impressora.

incremento - Valor acrescentado a uma variável.

índice - (1) Número que especifica a localização de um dado numa matriz ou tabela.
(2) Incremento ou decremento da base de um endereço.

informações - Dados, sinais, comandos, caracteres.

inicializar - (1) Atribuir valores às variáveis antes de começar o processamento.
(2) Veja formatar (1).

INIT - Abreviatura de initialize (inicializar).

instrução - Ordem para que o computador realize alguma tarefa específica.

interface - Placa ou cartão contendo um circuito que compatibiliza sinais entre itens de hardware, conectando periféricos ao computador, ou que fornece expansão de memória, ou que permite obter 8Ø colunas no vídeo, etc.

interromper - (1) Parar a execução de um programa, definitiva ou temporariamente.
(2) Suspender qualquer operação, comando, modo de operação.

jumper - Conector, ponte.

kB - Abreviatura de kilobyte.

kilobyte - Unidade de medida de capacidade da memória, representando 1.Ø24 bytes ou 8.192 bits.

led - Indicador luminoso que sinaliza o funcionamento ou acionamento do aparelho.

ler - Veja carregar.

limpar - Veja delete.

linguagem de máquina - Linguagem de programação, de baixo-nível, cujos comandos, sendo escritos em códigos binários, são diretamente executados pelo microprocessador. Cada microprocessador entende uma linguagem de máquina própria.

linha - Disposição, na tela ou na impressora, na direção horizontal.

linha de comando - A seqüência da instrução (instruções) até que se tecle return. Pode ocupar uma ou várias linhas da tela.

linha de programa - Linha de comando dentro de um programa.

listar - Apresentar na tela uma, algumas ou todas as linhas do programa.

listagem - Ação e efeito de listar.

loop - Ciclo. Seqüência do programa que é executada repetitivamente até que determinada condição seja satisfeita.

mapa de memória - Tabela que representa a alocação das áreas de memória.

matriz - Organização, arranjo dos dados em forma de tabela.

MByte - Abreviatura de megabyte.

megabyte - Unidade de medida de armazenamento, representando 1.Ø48.576 bytes.

memória - O componente de hardware destinado à armazenagem dos dados.

minuendo - Valor a ser subtraído.

microprocessador - Veja CPU.

mnemônico - Codificação que lembra a função ou o significado de instruções, registros etc.

mode - Modo.

modem - Contração de modulator-demodulator. Dispositivo que modula e demodula os sinais transmitidos por equipamentos e redes de comunicação.

modulador RF - Modulador de rádio-freqüência. Dispositivo que converte os sinais de vídeo gerados pelo computador para que esses sejam aceitos pela TV.

multiplicador - Valor a ser multiplicado.

multiplicando - Valor pelo qual o multiplicador é multiplicado.

nome de arquivo - Nome (conjunto de caracteres) que identifica um arquivo.

número binário - Número representado em notação binária.

operador lógico - Símbolo que representa uma operação lógica, combinando valores para produzir um resultado lógico.

operador matemático - Símbolo que representa uma operação matemática, inclusive as operações comparativas.

paddle - Controlador manual, com um botão de disparos, usado principalmente em jogos eletrônicos.

página - (1) Área da memória contendo o texto ou gráfico em apresentação no vídeo. (2) Uma tela inteira.

palavra reservada - Palavra do repertório da linguagem do computador e que não pode ser usada como nome-de-variável.

parcela - Valor a ser somado.

partida fria - Processo de inicialização do TK //e quando a tensão é ligada (ou em situações em que o efeito é o mesmo que desligar e ligar a fonte) e o sistema operacional é carregado na memória principal e, a seguir, carregando e executando um programa (compare com partida quente.

partida quente - Processo de reinicialização do TK //e, sem que haja a necessidade de se recarregar o sistema operacional na memória, e sem que se perca o programa ou informações essenciais.

pausa - Parada temporária do processamento ou da impressão.

periférico - Qualquer aparelho ou dispositivo externo ao computador, como unidade de disco, modem, paddle, impressora, etc.

pilha - Área da memória onde os dados podem ser armazenados continuamente e onde o dado disponível para recuperação é sempre o que foi armazenado em último lugar.

placa - Veja interface.

ponto flutuante - Relativo a um sistema de numeração em que a posição da vírgula não permanece fixa em relação a um extremo dos dígitos.

posição de memória - Endereço de armazenamento.

prancha digitalizadora - Digitalizador (tablet).

processamento - Tratamento ou manipulação da entrada dos dados, de acordo com as instruções, e execução de operações destinadas à consecução de certa tarefa.

PRODOS - Abreviatura de Profissional Disk Operating System.

programar - (1) Escrever um programa.
(2) Atribuir determinadas funções às teclas programáveis.

radicando - Valor a ser radiciado. Valor do qual se extrai a raiz.

randômico - Aleatório. Relativo a um conjunto cujos elementos não mantêm entre si nenhuma relação de ordem, de organização.

registrador - Dispositivo interno do microprocessador destinado a armazenar dados que serão usados no processamento.

reset - Restaurar. Voltar ao estágio inicial.

residente - Programa que sempre está na memória.

return - Retorno.

rotina - Conjunto de instruções necessárias para a execução de determinado passo dentro do programa.

saída - (1) A transferência de dados ou informações do computador para uma fonte externa, como o vídeo, unidade de disco, gravador.
(2) O abandono de determinada operação ou processo.

salvar - Veja gravar.

shift - (1) Movimento de um caractere ou grupo de caracteres para a esquerda ou direita de uma posição da memória.
(2) Deslocamento dos dados para a direita ou esquerda.

sintaxe - Conjunto de regras que governam a estruturação de comandos e de instruções numa linguagem de programação.

sistema operacional - Software que controla a execução das operações do computador, ordenando-as, comandando entrada, saída, compilação, alocação de memória etc.

slot - Fenda onde se encaixa uma placa de interface.

software - (1) Suporte lógico. Suporte de programação. Programas relacionados com a operação do computador.
(2) Programa aplicativo.

soft-switches - Determinados endereços capazes de provocar alterações co computador, quando acessados.

stack - Pilha.

step - Etapa, passo. sub-rotina - Rotina subordinada a uma rotina principal. Quando é chamada, a sub-rotina recebe o controle da execução de determinada operação. Ao término da operação, o controle volta à rotina principal.

subtraendo - Valor a ser subtraído do minuendo.

tabular - Alinhar uma tabela ou as margens direita e esquerda de um texto.

teclar - Veja digitar.

tela - Meio físico onde se faz o display das imagens geradas pelo computador, em geral a tela da TV ou do monitor de vídeo.

tempo de execução - Tempo necessário para decodificar e realizar instruções de um programa.

texto - Modo do display que habilita a representação visual dos caracteres - veja gráfico (2).

unidade de disco - Disk drive. Dispositivo que grava e recupera informações na superfície magnética de disquetes.

valor - (1) Quantidade.
(2) Item de informação que pode ser atribuído a uma variável (número ou string).

vídeo - Tela.

Índice Alfabético-Remissivo

A

ABS X-6
Acumulador XIV-2, H-1 a H-3
Adição
    ver soma
Alta-resolução gráfica
    ver gráficos
AND VIII-2 a VIII-4
Apontador
Apóstrofo XIV-5
Armazenamento XI-3, XII-1
Array
    ver matrizes
Arredondamento II-6
ASC VII-7
ASCII VII-6 a VII-9
    códigos VII-9, F-1,
Assembly XIV-11
Asterisco XIV-3
ATN X-7

B

Baixa-resolução gráfica
    ver gráficos
Banco de memória G-2 a G-4
Barra de espaço I-3
BLOAD XIV-9
Blocos de memória XIV-6 a XIV-9
BRUN XIV-9
BSAVE XIV-9
Buffer I-1

C

Cadeia (string) IV-1, VI-1, VII-4 a VII-8
Calculadora II-1
CALL XIV-1
caps lock I-3
Caracteres
    acentuados I-7
    de controle I-4
    em branco I-3
    maiúsculos I-3
    minúsculos I-3

Carregar
do disquete XI-4
da fita cassete XII-1 a XII-3
CATALOG XI-4
CHR$ VII-8, XVI-1
CLEAR VI-10
COLOR XIII-1
Comandos
códigos D-1
Condicionais IX-1 a IX-3
CONT VI-7
control I-4
coordenada
horizontal XIII-1
vertical XIII-1
Cores
códigos da alta-resolução XIII-1
códigos da baixa-resolução XIII-3
COS X-2
Cursor I-4

D

Dados IV-1
DATA VI-1
DEF FN VIII-6
DEL V-1
delete I-4, I-10
Depuração de programas
DIM VII-11 a VII-13
Dois pontos III-8
DOS XIV-8, XIV-1
Digitalizador
Diretório XI-1
Disc drive
ver unidade de disco
Divisão II-4
DRAW XV-1, XV-8, XV-9
Dupla alta-resolução
ver gráficos

E

Economia de tempo e de espaço C-1
Edição V-1 a V-7
END III-3, VI-7
Erros
códigos dos B-3
mensagens de B-1 a B-3
esc I-9
EXAMINE XIV-11
EXP XV-8

Expoente X-8
Exponenciação X-8

F

Falso VII-1, IX-1
FLASH VI-11
FOR IV-8
Formatação V-7 a V-9, XI-1
FRE(Ø) VI-1Ø
Funções trigonométricas X-1 a X-5, X-7, A-1 e A-2

G

GET VIII-6
GOSUB IX-3 a IX-6, IX-8
GOTO IV-4
GR XIII-1
Gráficos
    alta-resolução XIII-3 a XIII-7
    baixa-resolução XIII-1, XIII-2
    das funções trigonométricas X-3 a X-5
    dupla alta-resolução I-4, XIII-7
    manipulação de figuras em alta-resolução XV-1 a XV-1Ø
Gravar
    no disquete XI-1
    na fita cassete XII-1 a XII-3

H

HCOLOR XIII-3
HGR XIII-4
HGR2 XIII-7
HIMEM: XIV-2, XV-1Ø
HLIN XIII-2
HOME III-3
HPLOT XIII-4
HTAB V-8

I

IF IX-1, IX-2
Inicializar XI-1
INIT XI-1
INPUT IV-5
INVERSE VI-11
INT VI-6
Interfaces

L

LEFT$ VI-2
LEN VI-1

Leitura XI-4, XII-1 a XII-3
LET IV-2
Linguagem de máquina XVI-1 a XVI-11, XVI-2
Linha
de programa III-4
enumeração III-4
LIST III-3
LOAD XII-1
LOG X-8
LOMEM: XIV-2
Loop IV-8

M

Mapa I-1
de utilização da RAM G-1
da ROM G-2
Matrizes VII-9 a VII-13
bidimensionais VII-10, VII-12
tridimensionais VII-10, VII-11, VII-13
unidimensionais VII-10, VII-11, VII-12
Memória I-1
endereçamento H-9 e H-10
mapas G-1 e G-2
VI-12, G-2
Mensagens de erro B-1 a B-3
Microprocessador
ver 65C02
MID$ VI-4
Mini Assembler XIV-11 a XIV-14
acessando XIV-12
digitando um programa XIV-13
saindo XIV-12
usando XIV-12
Mode I-7
Modem
Modo
imediato II-1
acentuado I-7
CAPS LOCK I-3
ESCAPE I-9, IV-7
gráfico XIV-1 a XIV-7
programado III-4
pré-programado I-9
PROG I-5
Monitor XIV-2
texto I-3, V-7
Monitor XIV-2 a XIV-11
acessando XIV-3
alternando o conteúdo da memória XIV-5
examinando a memória XIV-3
comparando dois blocos de memória XIV-3
criando e executando programas em linguagem de máquina XIV-10

examinando registradores XIV-11
salvando e carregando blocos de memória em disquete XIV-8
salvando e carregando blocos de memória em fita cassete XIV-8
transferindo um bloco de memória XIV-6
Música XVI-3
Multiplicação II-3

N

NEXT IV-8
NEW III-1
Nome-de-variável IV-1
NOT VII-2 a VII-4
NOTRACE VI-10
NORMAL VI-11
Números II-6
aleatórios VI-5
fracionários II-6
hexadecimais XIV-3, XIV-7, XIV-12
inteiros VI-7, VI-6
notação científica II-7, II-8
randômicos ver aleatórios
reais II-9

O

ON-GOSUB XIV-5
ON-GOTO XIV-2
ONERR GOTO XIV-6
Operações
algébricas II-6
comparativas VI-1
lógicas VI-2
matemáticas X-1 a X-9

P

Palavras reservadas
Parênteses II-6
Partida fria I-8
Pausa XIV-1
PEEK VI-12, XVI-2
Pilha XIV-11
PLOT XIII-1
POKE VI-12
Ponto decimal II-6
Ponto de exclamação XIV-12
Ponto e vírgula III-5
Ponto flutuante XII-2 POP IX-8
POS V-9
Potenciação II-4
PPA XIV-3, XIV-5
PRINT III-1 a III-2

"conteúdo" III-1
conteúdo III-2
conteúdo, conteúdo, conteúdo, ... III-2
conteúdo ; conteúdo III-2
Prioridade II-1
Prog I-5
Prompt
ver sinal de pronto

R

Radiciação II-5
Raiz quadrada II-5, X-9
RAM
ver memória
Randômico
READ VIII-1
RECALL XII-3
Registrador XIV-3, XIV-11
REM XIV-8
reset I-8
return I-4
RETURN IX-3
RESTORE VIII-4
RESUME IX-7
RIGHT$ VI-3
RND VI-5, X-5 a X-6
ROM
Rotina X-9
RTS XIV-10
RUN III-3

S

SAVE XII-1
SCALE XV-10
SCRN XIII-3 65C02 I-2, XIV-1, G-1, H-1 a H-10 shift I-3
SHLOAD XV-7, XV-8, XV-10
Símbolos das operações algébricas V-1
SIN X-1
Sistema operacional XI-3
Soft-switches G-3 e G-4
Som XVI-1 a XVI-4
Soma II-2, XIV-7
SPC V-8
SPEED VI-11
SQR II-5, X-5
Stack XIV-11
STEP IV-10
STOP VI-7
STORE XII-2
String
ver cadeia

STR$ VIII-4, VIII-5
Sub-rotinas XIV-3 a XIV-8, XIV-10
Subtração II-2, XIV-7
Super Parallel I-7

T

tab I-5
TAB V-7, X-3 a XV-8
Tabela de figuras XV-1 a XV-8
Tabulação V-7
Teclado I-4, I-7
TEXT XIII-1, XIII-4
TRACE VI-9

U

Unidade Central de Processamento
    ver CPU
UPA XIV-3
USR XIV-2

V

VAL VII-6
Variáveis IV-1, X-9
    inteiras IV-2
    reais IV-2
    string IV-2
Verdadeiro VII-1, IV-1
VERIFY XIV-7
Vírgula II-6
VLIN XIII-2
VTAB V-8

W

WAIT XIV-1

X

XDRAW XV-1, XV-2, XV-9

Z

Z80 I-1

Fotocomposição direta em filme positivo, a partir do disquete do cliente, impressão e acabamento.

BANDEIRANTE S.A. GRÁFICA E EDITORA
452-3444 / 572-0033