

TK 2000/II

manual de operação

MICRODIGITAL

INTRODUÇÃO

INTRODUÇÃO

A disseminação do computador é hoje uma tendência irreversível. Não está longe o dia em que o computador será tão comum e ainda mais imprescindível do que atualmente é o telefone.

Este manual tem a finalidade de introduzi-lo na operação do TK-2000 COLOR. No primeiro capítulo são apresentados alguns conceitos gerais de computação necessários no decorrer do manual. O segundo capítulo descreve especificamente o computador TK-2000 COLOR. Os demais capítulos se referem ao uso da linguagem BASIC em nosso equipamento.

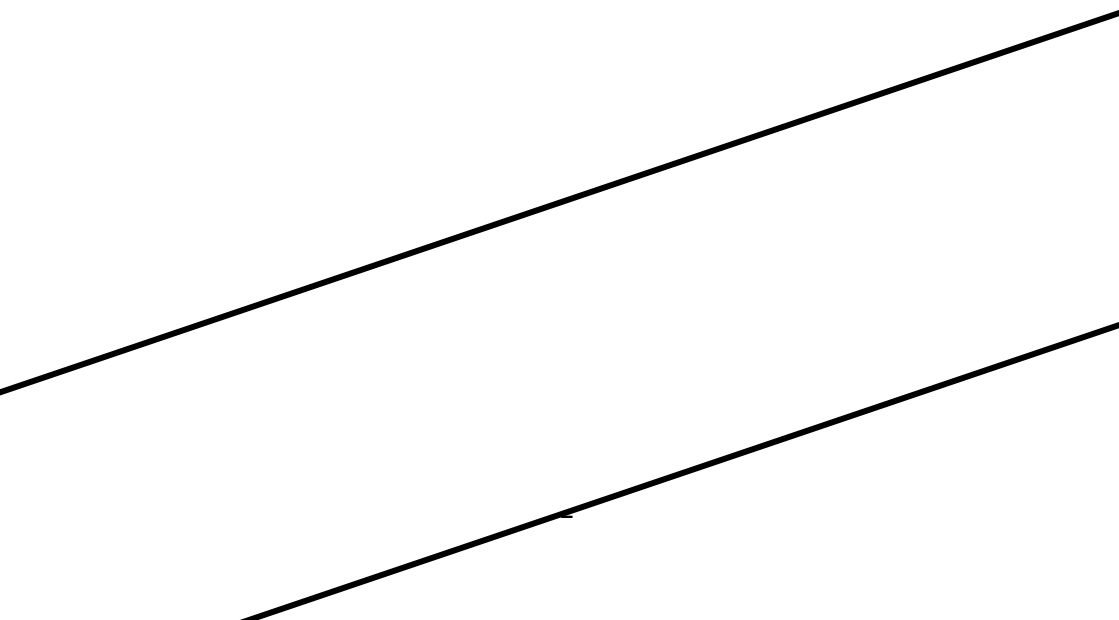
No final de cada capítulo há um item denominado "FAÇA O SEU RESUMO", composto de várias questões para você completar. Para permitir verificação, as respostas são fornecidas, após a última questão. "FAÇA O SEU RESUMO" tem dupla utilidade. A primeira é, após a leitura do capítulo, quando você deverá então completar as questões, verificar se o capítulo foi bem compreendido: caso você sinta dificuldade em completar as questões procure reler os pontos de dividas do capítulo. A segunda utilidade é, após a leitura completa do manual, servir de consulta rápida, ou seja, se durante a operação do TK-2000 COLOR surgir alguma dificuldade, provavelmente você encontrará a resposta num dos itens "FAÇA O SEU RESUMO", uma vez que eles reúnem os aspectos mais relevantes de cada capítulo.

Ainda assim, não se contente apenas com a leitura deste manual. Somente seu interesse, sua prática e pesquisa irão torná-lo um hábil operador desta máquina do futuro - O Computador TK-2000 COLOR- .

Antes de continuar, leia atentamente o Manual de Instalação do TK-2000 COLOR.

E proibida a reprodução total ou parcial deste manual sem prévia autorização por escrito da

MICRODIGITAL ELETRÔNICA LTDA.



INDICE

CAPITULO I. O COMPUTADOR

- I.1. O que é um computador? 11
- I.2. O Computador Pessoal 11
- I.3. Partes de um Computador 12
 - I.3.1. Memória 13
 - I.3.2. Unidade Central de Processamento (U.C.P.) 14
 - I.3.3. Entrada 14
 - I.3.4. Saída 19
 - I.3.5. Memória Auxiliar 15
- I.4. Sistema de Numeração Binário 15
- I.5. Programas e Linguagens 17

CAPITULO II. APRESENTAÇÃO DO COMPUTADOR TK-2000 COLOR

- II.1. Características Técnicas 19
- II.2. Início de Operação 21
- II.3. Utilização da Tela 21
 - II.3.1. O sinal de comando (>) 22
 - II.3.2. O Cursor (█) 22
- II.4. Utilização do Teclado 22
 - II.4.1. Teclas de Caracteres ao
 - II.4.1.a. Uso dos Caracteres Gráficos 24
 - II.4.2. Teclas de Mudança 24
 - II.4.2.a. SHIFT 24
 - II.4.2.b. CONTROL 25
 - II.4.3. Teclas de Funções 25
 - II.4.3.a. RETURN 25
 - II.4.3.b. RESET 26
 - II.4.3.c. Teclas de Movimentação do Cursor 26
 - II.4.4. REPEAT 27
- II.5. O GRAVADOR CASSETE 27
 - II.5.1. Cuidados com a Fita Cassete 27

- II.5.2. Organização das Fitas 28
- II.5.3. Proteção contra regravação 28
- II.5.4. Ajuste do Gravador 29
- II.5.5. Carga do Programa 29
 - II.5.5.a. Erro na carga do "TESTE" 30
 - II.5.5.b. Procedimento Normal para carga de Programas 31

CAPITULO III- O TK-2000 COLOR COMO CALCULADORA

- III.1. Como utilizar o modo imediato 35
- III.2. As Operações Aritméticas 35
- III.3. Utilização das Funções Matemáticas através do Modo Imediato 38
- III.4. O Formato dos Números no TK-2000 COLOR 38

CAPITULO IV- INICIANDO A PROGRAMAÇÃO

- IV.1. As primeiras instruções 41
 - IV.1.1. NEW 41
 - IV.1.2. PRINT 41
 - IV.1.3. HOME 43
 - IV.1.4. END 43
 - IV.1.5. LIST 43
 - IV.1.5.a. CONTROL-S 43
 - IV.1.5.b. CONTROL-O 44
 - IV.1.6. RUN 44
- IV.2. O Modo Programado 44
 - IV.2.1. Numeração das Linhas 44
 - IV.2.2. O Primeiro Programa 46
- IV.3. Uso de Dois Pontos (:) numa Linha de Comando 49
- IV.4. Uso de Ponto e Vírgula após o Comando PRINT 49

CAPITULO V- OUTROS COMANDOS

V.1. Alguns Conceitos Importantes 51

V.1.1. Dados 51

V.1.1.a. Cadeias 51

V.1.1.b. Números 51

V.1.2. Notação Científica 52

V.1.3. Arredondamentos 53

V.1.4. Variáveis 54

V.1.5. Nome de Variáveis no BASIC do TK-2000 COLOR 54

V.2. LET 55

V.3. GOTO 57

V.4. INPUT 59

V.4.1. Incrementando a instrução INPUT 60

V.4.2. Uso de INPUT com Variáveis numéricas 62

V.5. REM 63

CAPITULO VI- À INSTRUÇÃO <FOR...NEXT> E TECNICAS DE EDIÇÃO

VI.1. FOR...NEXT 65

VI.1.1. Uso de FOR...NEXT como artifício matemático 69

VI.1.2. Uso de FOR...NEXT como tempo de espera 70

VI.1.3. STEP 72

VI.2. Técnicas de Edição 72

VI.2.1. Supressão de Linhas de Programa (Comando DEL) 72

VI.2.2. Permutando Caracteres 73

VI.2.3. Supressão de Caracteres 74

CAPITULO VII- GRUPOS DE COMANDO E INSTRUÇÕES

VII.1. Operação de cadeias 77

| | | |
|----------|---|----|
| VII.1.1. | LEN | 77 |
| VII.1.2. | LEFT\$ | 78 |
| VII.1.3. | RIGHT\$ | 79 |
| VII.1.4. | MID% | 80 |
| VII.2. | Operação de Dados Numéricos | 82 |
| VII.2.1. | RND | 82 |
| VII.2.2. | INT | 83 |
| VII.3. | Comandos Relacionados à execução de Programas | 84 |
| VII.3.1. | LOAD e SAVE | 84 |
| VII.3.2. | STOP e CONT | 85 |
| VII.3.3. | CONTROL-C e RESET | 87 |
| VII.3.4. | TRACE E NOTRACE | 88 |
| VII.3.5. | POKE e PEEK | 89 |
| VII.4. | Instruções relativas a Edição Formato | 90 |
| VII.4.1. | TAB | 90 |
| VII.4.2. | VTAB E HTAB | 90 |
| VII.4.3. | SPC | 91 |
| VII.4.4. | POS | 91 |
| VII.4.5. | CLEAR | 92 |
| VII.4.6. | FRE (0) | 93 |
| VII.4.7. | INVERSE e NORMAL | 93 |
| VII.4.8. | SPEED | 93 |
| VII.4.9. | CONTROL-X | 93 |

CAPITULO VIII- OPERAÇÕES NO TK-2000 COLOR

| | | |
|-------------|-----------------------------------|-----|
| VIII.1. | Operações com Variáveis Numéricas | 97 |
| VIII.1.1. | Inteiros | 97 |
| VIII.1.2. | Reais | 98 |
| VIII.2. | Tipos de Operações Numéricas | 99 |
| VIII.2.1. | Operações Numéricas | 99 |
| VIII.2.2. | Operações Comparativas | 99 |
| VIII.2.3. | Operações Lógicas | 101 |
| VIII.3. | Novas Operações com Cadeias | 103 |
| VIII.3.1. | STR\$ (X) e "+" | 105 |
| VIII.3.2. | VAL (A\$) | 105 |
| VIII.3.3. | O Código ASCII | 105 |
| VIII.3.3.a. | ASC (AS) | 106 |
| VIII.3.3.b. | CHR\$ (X) | 107 |
| VIII.4. | Matrizes | 108 |
| VIII.4.1. | DIM | 110 |
| VIII.4.1.a. | Matrizes Unidimensionais | 111 |
| VIII.4.1.b. | Matrizes Bidimensionais | 112 |
| VIII.4.1.c. | Matrizes Tridimensionais | 113 |
| VIII.4.2. | STORE e RECALL | 113 |
| VIII.4.3. | Comando MOTOR | 114 |

CAPITULO IX- INSTRUÇÕES DE ENTRADA E SAIDA

- IX.1. READ...DATA 117
- IX.2. RESTORE 121
- IX.3. GET 122
- IX.4. DEF FN 123

CAPITULO X- TRAÇANDO GRAFICOS E FIGURAS

- X.1. COLOR 127
- X.2. PLOT, GR e TEXT 127
- X.3. HLIN e VLIN 129
- X.4. SCRN 129
- X.5. Traçados de Alta Resolução 130
 - X.5.1. HCOLOR 130
 - X.5.2. HPLOT, HGR e TEXT 130
 - X.5.3. HPLOT X1,Y1 TO X2,Y2 133
 - X.5.4. HGR2 134

CAPITULO XI- LIDANDO COM DESVIOS

- XI.1. Desvios Condicionais 137
 - XI.1.1. IF...GOTO 137
 - XI.1.2. IF...THEN 138
 - XI.1.3. ON...GOTO 139
 - XI.1.4. IF...THEN...GOTO 140
- XI.2. Sub-Rotinas 141
 - XI.2.1. GOSUB e RETURN 141
 - XI.2.2. Sub-Rotinas dentro de uma Sub-Rotina 142
 - XI.2.3. ON...GOSUB 143
 - XI.2.4. ONERR...GOTO 144
 - XI.2.5. POP 145

CAPITULO XII- FUNÇÕES MATEMATICAS

- XII.1. Funções Trigonométricas 149
 - XII.1.1. SIN(X) 149
 - XII.1.2. COS(X) 150
 - XII.1.3. TAN(X) 150

XII.2. Traçados de Gráficos de Funções 151

XII.2.1. Diagramas Simples 151

XII.2.2. Gráfico da Função Seno 152

XII.2.3. Gráfico da Função Cosseno 153

XII.2.4. Outras Funções 153

XII.2.4.a. Função Randômica - RND(X) 153

XII.2.4.b. Função Valor Absoluto - ABS(X) 154

XII.2.4.c. Função Trigonométrica Inversa -
ATN(X) 155

XII.2.4.d. Reconhecimento do Sinal - SBN(X) 155

XII.2.4.e. Função Exponencial - EXP(X) 155

XII.2.4.f. Função Logarítmica - LOG(X) 156

XII.2.4.g. Raiz Quadrada SQR(X) 156

CAPITULO XIII - MANIPULAÇÃO DE FIGURAS EM ALTA RESOLUÇÃO

XIII.1. INTRODUÇÃO 159

XIII.1.1. Modo Monitor 163

XIII.1.2. Carga da Tabela de Figuras 164

XIII.1.3. Indicando ao BASIC o local da Tabela de
Figuras 165

XIII.1.4. Proteção da Tabela de Figuras 165

XIII.1.5. Armazenando uma Tabela de Figuras 165

XIII.1.6. Saindo do Modo Monitor 166

XIII.2. Usando a Tabela de Figuras 166

XIII.2.1. DRAW 167

XIII.2.2. XDRAW 167

XIII.2.3. ROT e SCALE 168

XIII.2.4. SHLOAD 169

CAPITULO XIV – USO DE ROTINAS EM LINGUAGEM DE MAQUINA

XIV.1. CALL 173

XIV.2. WAIT 173

XIV.3. USR(X) 174

XIV.4. HIMEM: 174

XIV.5. LOMEM: 174

CAPITULO XV – O COMANDO SOUND

XV.1. O Comando SOUND 177

XV.1.1. SOUND X1,Y1 TO X2,Y2 177

XV.1.2. Pausa 178

XV.2. Executando Músicas 178

XV.3. Efeitos Sonoros em linguagem de máquina 181

- APENDICE A - FUNÇÕES TRIGONOMETRICAS 185
- APENDICE B - MENSAGENS DE ERRO 187
- APENDICE C - ECONOMIA DE ESPAÇO E TEMPO 191
- APENDICE D - CODIGOS DOS COMANDOS 193
- APENDICE E - PALAVRAS RESERVADAS 195
- APENDICE F - MAPA DA MEMORIA 197
- APENDICE G - TABELA DE CODIGOS ASCII 201
- APENDICE H - OPERAÇÕES EM LINGUAGEM DE MAQUINA 203
- APENDICE I - COMPARAÇÃO COM APPLE II PLUS 205

INDICE ALFABETICO - 207

CAPÍTULO 1

CAPITULO I : O COMPUTADOR

I.1. O QUE E O COMPUTADOR

Antes de iniciarmos a operação do computador TK-2000 COLOR, convém sabermos o que é um computador.

Inicialmente, podemos comparar o computador a uma criatura que, apesar de não ter um mínimo de criatividade ou capacidade de dedução, estando com as instruções minuciosas e corretas nas mãos, é capaz de realizar serviços com extrema rapidez e precisão. Porém, note bem, ele não é capaz de fazer absolutamente nada que não esteja nas instruções predeterminadas.

Seria praticamente impossível descrever todos os tipos de aplicação de um computador, porém alguma delas são:

- ARQUIVO DE DADOS: um computador é capaz de armazenar uma enorme quantidade de dados e, quando necessário, apresentá-los rapidamente.
- CONTROLE DE PROCESSO: Esta aplicação é bastante ampla, e pode ir desde o controle de máquinas operatrizes, até a monitoração de pacientes cardíacos.
- CALCULOS COMPLEXOS: A execução de cálculos repetitivos é uma das maiores utilidades do computador na área de projetos e pesquisas.
- CONTROLES ADMINISTRATIVOS: São poucas as empresas que, hoje em dia, não dispõem de um computador para executar suas folhas de pagamento, controle de estoque, mala direta, etc.
- EDUCAÇÃO: Nesta área, o uso do computador é relativamente recente no Brasil, porém muitas de nossas crianças já brincam entusiasmadas com a matemática e ciência da computação, bem como em outras áreas de conhecimento através dos computadores.
- LAZER: Os jogos realizados através do computador estão se expandindo a cada dia. Muitas famílias já passam horas divertindo-se diante de uma destas máquinas.

I.2. O COMPUTADOR PESSOAL

O computador é um equipamento que possuem uma gama extremamente variada de capacidade e preço. Dentro desta variedade, encontramos o computador pessoal que, embora de um preço acessível e capacidade restrita, é capaz de reproduzir quase tudo que um computador de maior porte faz.

Desta forma, um computador pessoal não é capaz de arquivar as fichas de todos os pacientes do Hospital das Clínicas, mas pode armazenar os arquivos necessários a um consultório particular, ou as receitas culinárias de uma dona de casa, Ele pode também executar controles administrativos, domésticos, ou até de pequenas empresas: auxiliar no ensino, e proporcionar horas de apaixonante lazer.

1.3. AS PARTES DE UM COMPUTADOR

Analisando de forma simplificada, as partes que compõem um computador são: Memória, Unidade Central de Processamento (U.C.P.), Entrada e Saída. Estes elementos se interagem conforme esquematiza a figura I.1.

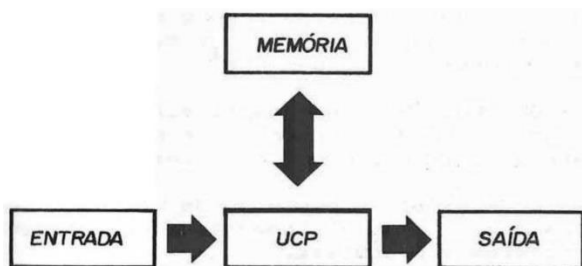


Fig. I.1

Podemos fazer uma analogia entre o computador e um escritório. A Entrada corresponde a recepcionista, ao receber os pedidos: a Memória, à pessoa que cuida do arquivo: a U.C.P. ao chefe do escritório, que além de dar ordens, possui uma máquina de calcular: a Saída ao entregador, que leva os resultados ao cliente. A figura I.2 ilustra o nosso exemplo.



Fig. I.2

Na sequência desse item o exemplo desta figura será melhor compreendido.

I.3.1. Memória

A memória é o elemento que permite ao computador armazenar ou reter as informações. Podemos dividir as informações em dois grupos, as instruções e os dados.

As instruções dizem ao computador o que fazer. No exemplo da fig.I.2, quando o arquivista diz ao chefe "Calcular a dívida de Antônio", ele está fornecendo uma instrução.

Pode-se entender por dados como sendo a matéria prima e o produto do computador, ou seja, a finalidade em si do computador e receber determinados dados, executar algum tipo de processamento sobre os mesmos e, quando solicitado, fornecer determinado resultado. Em nosso exemplo, os dados de entrada estão na "ficha do Antônio" e o dado de saída é o total da dívida.

Cada informação da memória está armazenada em um local determinado, chamado "endereço" ou "posição" da memória. Podemos imaginar a memória como um conjunto de caixas do correio, correspondendo a etiqueta de cada uma destas caixas aos endereços e, o conteúdo delas a informação.

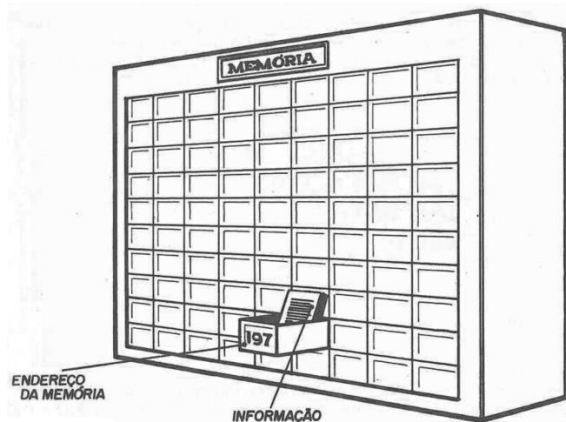


Fig. I.3

As memórias podem ainda serem divididas em duas famílias básicas: ROM (Read Only Memory), são memórias com informações fixas, OU seja, não se pode alterar o conteúdo de cada um de seus endereços. Estas memórias contêm informações de controle utilizadas para o próprio funcionamento do computador

RAM (Random Access Memory), estas memórias permitem a alteração do seu conteúdo, porém, quando o computador é desligado, todas as memórias deste tipo são apagadas, sendo então perdidas as informações nelas armazenadas. Nestas memórias são também armazenadas informações fornecidas pelo operador ao computador.

1.3.2. U.C.P.

A Unidade Central de Processamento é onde se encontra o "cérebro" do sistema. Suas funções podem ser divididas em funções de controle e aritméticas. No exemplo do nosso escritório, quando o chefe pergunta ao arquivista "Qual a próxima tarefa", pede a ficha do Antônio e fala para avisar o entregador do total da dívida, ele está exercendo suas funções de controle. A maquininha de calcular do chefe, através do qual ele encontra total da dívida corresponde a função (ou unidade) aritmética.

Pode-se dizer que a U.C.P. é a unidade mais complexa, ou a própria alma do computador. Há algumas décadas atrás, esta unidade, bem como todo o computador, eram feitos com componentes eletrônicos discretos (válvulas ou transistores) e ocupavam um grande espaço. Hoje, graças ao desenvolvimento da técnica - denominada L.S.I. (Large Scale Integration), unidades muito mais eficientes que as antigas ocupam uma área aproximada de 10 cm², com apenas alguns milímetros de altura. A U.C.P. do nosso computador TK-2000 COLOR é apresentada na figura 1.4,



Fig. I.4

I.3.3. Entrada

A entrada de um computador, representada em nosso exemplo pela recepcionista, é o elemento que recebe as informações (dados e instruções) e as transmite para a memória. Existem diversas formas de se executar esta entrada como através de fitas perfuradas, cartões etc.; porém as mais comuns e práticas são utilizadas pelo TK-2000 COLOR que são o teclado (geralmente semelhantes ao de uma máquina de escrever), e também o disco magnético e fita cassete (estes opcionais) através do qual você entrará com instruções e/ou dados.

1.3.4. Saída

Assim como no caso do entregador do nosso escritório, a Saída e que transmite ou apresenta os resultados do processamento feito pelo computador. Alguns dos tipos de equipamento de saída do TK-2000 COLOR são: impressora, terminais de vídeo e tela de televisão.

1.3.3. Memória Auxiliar

A Memória Auxiliar não entrou em nosso exemplo por não fazer parte intrínseca do computador, ou seja, ele pode operar sem esta unidade, se bem que estejam limitadas as possibilidades de aplicação.

Mesmo para a execução de uma operação relativamente simples, geralmente o computador necessita de um número bastante grande de instruções e dados. Estes são armazenados nas memórias RAM do computador (que como já dissemos, perdem seu conteúdo quando o equipamento é desligado). Para que estes não se percam, eles podem ser gravados na Memória Auxiliar e assim, quando necessário, requisitados novamente pelo computador. Portanto a Memória Auxiliar é uma unidade capaz de reter um grande número de informações, independente do computador estar ou não desligado.

O Computador TK-2000 COLOR usa para esta finalidade um gravador cassete. Note que em uma única fita cassete, pode-se gravar instruções e dados relativos a muitas operações. Além do mais, elas podem ser adquiridas já com informações gravadas, para execução de determinadas tarefas, assim como se pode comprar uma fita. com música já gravada.

Outro tipo de memória auxiliar é a unidade de discos que é um equipamento opcional do TK-2000 COLOR.

I.4. SISTEMA DE NUMERAÇÃO BINÁRIO

Normalmente, para executar nossos cálculos, utilizamos o sistema de numeração decimal, que é formado através da combinação de dez símbolos diferentes (0 a 9).

Por que dez e não oito ou vinte símbolos diferentes? Provavelmente devido ao número de dedos das nossas mãos, pois nada impede que se adote um número maior ou menor de símbolos, para um determinado sistema de numeração.

Os sistemas internos de numeração dos computadores usam apenas dois símbolos, ou seja, trabalham em base binária ao invés da base decimal. Estes dois símbolos podem ser representados por el, Sim ou Não, uma lâmpada apagada ou acesas uma chave ligada ou desligada, uma tensão de 5 Volts ou 0 Volts. Veja na tabela a seguir a correspondência entre os 21 primeiros números na base binária e na base decimal (a tabela pode continuar indefinidamente).

| BASE BINARIA | BASE DECIMAL | BASE BINARIA | BASE DECIMAL |
|--------------|--------------|--------------|--------------|
| 00000000 | 0 | 00001011 | 11 |
| 00000001 | 1 | 00001100 | 12 |
| 00000010 | 2 | 00001101 | 13 |
| 00000011 | 3 | 00001110 | 14 |
| 00000100 | 4 | 00001111 | 15 |
| 00000101 | 5 | 00010000 | 16 |
| 00000110 | 6 | 00010001 | 17 |
| 00000111 | 7 | 00010010 | 18 |
| 00001000 | 8 | 00010011 | 19 |
| 00001001 | 9 | 00010100 | 20 |
| 00001010 | 10 | 00010101 | 21 |

Tabela I.1

O motivo da utilização dos números binários no computador é que, para ele é muito mais fácil trabalhar com chaves abertas ou fechadas, presença ou não de tensão e lâmpadas acesas ou apagadas, do que com dez diferentes símbolos. Por exemplo, uma saída rudimentar de um sistema de computação pode ser feito a partir de lâmpadas.

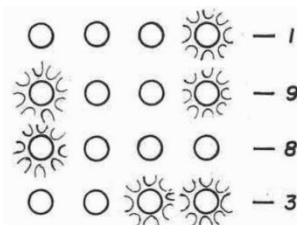


Fig. 1.5

Note que, a base binária permite os mesmos tipos de operadores da base decimal.

Exemplo:

$$\begin{array}{r}
 010 = 2 \\
 + 100 = 4 \\
 \hline
 110 = 6
 \end{array}$$

Um aprendizado mais profundo das operações na base binária pode ser obtido através do estudo da Álgebra Booleana.

Para fecharmos este item, é importante salientarmos duas definições muito usadas na computação: bit e byte. O bit é definido como um elemento da memória que pode ter duas situações complementares: 1 e 0, sim ou não, ligado ou desligado, etc. O byte

é o conjunto formado por 8 bits. (fig I.6).

| BYTE | | | | | | | |
|-------|-------|-------|-------|-------|-------|-------|-------|
| BIT 7 | BIT 6 | BIT 5 | BIT 4 | BIT 3 | BIT 2 | BIT 1 | BIT 0 |
| 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 |

obs. 0/1 significa 0 ou 1

Fig. 1.6

Assim um conjunto de 8 lâmpadas pode representar um byte, no qual cada lâmpada representa um bit.

OBSERVAÇÃO: um termo também muito comum na computação é o chamado quilo byte ou simplesmente KByte, que corresponde a 1024 bytes. Assim:

1 K bytes = 1 x 1024 bytes = 1024 bytes
4 K bytes = 4 x 1024 bytes = 4096 bytes
8 K bytes = 8 x 1024 bytes = 8192 bytes
16k bytes = 16 x 1024 bytes = 16384 bytes

1.5. PROGRAMAS E LINGUAGENS

Conforme dissemos, para que um computador execute qualquer tipo de operação, é necessário que lhe sejam fornecidas instruções detalhadas. Ao conjunto de instruções necessárias a execução de determinada tarefa dá-se o nome de programa.

Por enquanto, nenhum computador é capaz de entender instruções na linguagem coloquial (na linguagem em que normalmente falamos). Desta forma, as instruções fornecidas a ele são em número restrito e devem ter formatos apropriados. Uma determinada linguagem de computação define um conjunto de instruções dentro de um determinado formato.

O tipo de linguagem mais rudimentar, utilizada pelo computador é a linguagem de máquina, na qual as instruções são fornecidas através de bytes. Diz-se que uma linguagem é de alto nível quando as suas instruções procuram se assemelhar à linguagem humanas.

Dentre as linguagens de alto nível, BASIC, utilizada pelo nosso computador TK-2000 COLOR, se destaca pela sua versatilidade, facilidade de aprendizado e popularidade na área de informática.

FAÇA O SEU RESUMO

1. As principais partes de um computador são:,
..... e
2. Os dados e instruções do computador são armazenados na
.....
3. A é responsável pelo controle e funções
..... de um sistema de computação.
4. No computador TK-2000 COLOR, a tela da televisão corresponde a
unidade de é O teclado a unidade de
.....
5. Os computadores trabalham internamente dentro do sistema de
numeração
6. O pode ter o valor 1 ou 0. O conjunto de 8
..... é chamado O Kbyte corresponde a
.....
7. Um conjunto de instruções com determinado fim é chamado
..... . Um pode utilizar diversas
..... sendo que o TK-2000 COLOR utiliza a
- B. Os e dados utilizados pelo TK-2000 COLOR, podem
ser gravados em

respostas:

1. U.C.P., Memória, Entrada e Saída; 2. Memórias 3. U.C.P.,
aritméticas; 4. Saída, Entrada; 5. Binário; 6. Bit, Bits, bytes,
1024 bytes; 7. Programa, Programa, linguagem, linguagem BASIC; 8.
Programas, fitas cassete

CAPÍTULO **2**

CAPITULO II- APRESENTAÇÃO DO COMPUTADOR

Este capítulo visa colocá-lo pela primeira vez em contato com o computador TK-2000 COLOR. Inicialmente são apresentadas em rápidas palavras, as características mais relevantes do TK-2000 COLOR. Após a leitura deste primeiro item, você deverá estar com seu TK-2000 COLOR já instalado (de acordo com as instruções do Manual de Instalação) e, procurar acompanhar as demais descrições no seu próprio computador. Portanto prepare-se para ingressar no mundo prático da computação.

II.1. CARACTERISTICAS TECNICAS

O computador TK-2000 COLOR é constituído por um corpo, onde são alojados a U.C.P., circuitos lógicos, as memórias RAM e ROM, o teclado, e os conectores para ligação de um aparelho de televisão e/ou monitor de vídeo e gravador cassete. Assim mesmo, se encontram presentes conectores para ligação de impressora, do joystick, e um conector para ligação de periféricos vários, tais como cartuchos com programas fixos, discos magnéticos, saídas RS 232, gerador de som e voz, etc.

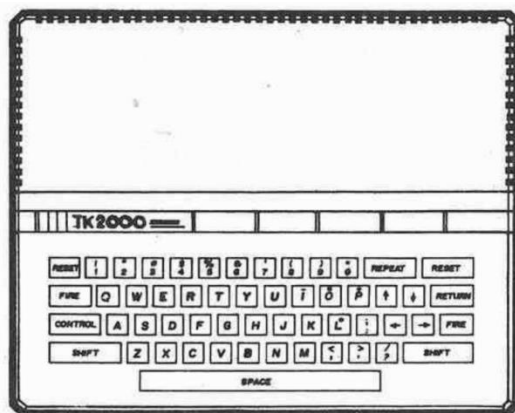


Fig. II.1

Como U.C.P. o TK-2000 COLOR utiliza o Microprocessador 6502.

Para armazenar seu sistema de controle, TK-2000 COLOR dispõe de 16 kBytes de ROM e, para a área que deverá conter programas carregados pelo operador (através do teclado ou fita cassete) 48 kBytes de memória RAM, que permitem armazenar o equivalente a 49.142 caracteres.

O teclado do TK-2000 COLOR é constituído por 54 teclas, que podem ser divididas em teclas de caracteres, de mudança e de funções, fig II.2 conforme será apresentado adiante.

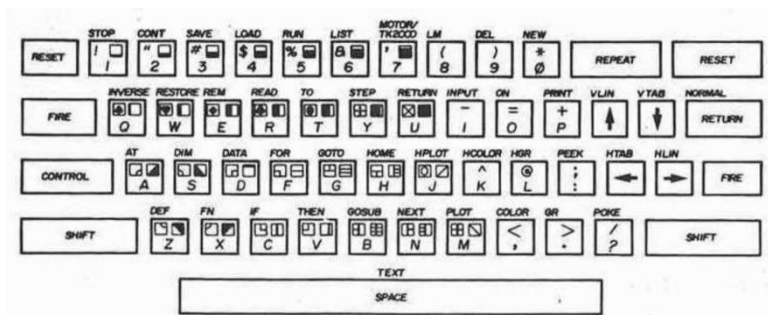


Fig. II.2

No conector do aparelho de televisão deve-se ligar qual quer tipo de aparelho, preto e branco ou colorido. Naturalmente, através do televisor colorido serão obtidos maiores recursos visuais, uma vez que o computador TK-2000 COLOR permite a definição de diversas cores para gráficos e figuras (fig II.3). Da mesma maneira, no conector para o gravador, pode ser ligado qualquer modelo do tipo cassete. Lembre-se, a falta do gravador não impedirá a operação do TK-2000 COLOR, porém este é um importante recurso para armazenamento dos programas que você irá desenvolver, bem como para a utilização de fitas cassete com programas já gravados que você poderá adquirir para criar uma boa biblioteca de programas.

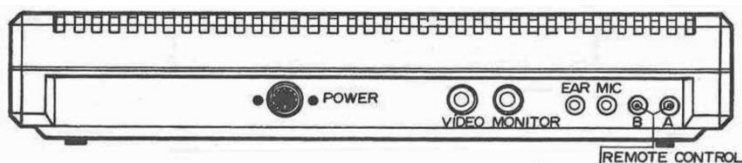


Fig II.3

Para detalhes da montagem do seu sistema, consulte o Manual de Instalação do Computador TK-2000 COLOR, fig.II.4

II.2 INICIO DE OPERAÇÃO

Como já dissemos, a partir deste ponto, para melhor aproveitamento do conteúdo deste manual, você deverá estar com seu sistema montado.

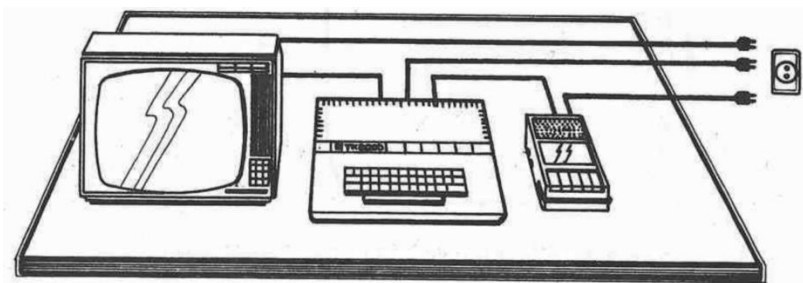


Fig II.4

Ligue agora o televisor (no canal 3) e, em seguida o TK-2000 COLOR. O TK-2000 COLOR emitirá então um sinal sonoro e uma luz vermelha (POWER) próxima ao teclado se acendera, indicando que o computador está pronto para operar.

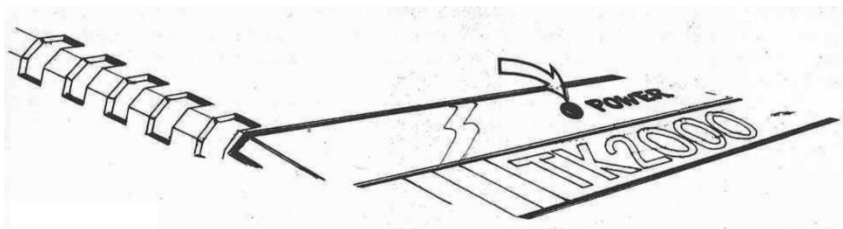


Fig II.5

Surge então na tela do televisor, a legenda "TK-2000" na parte central superior do vídeo, o sinal de comando (>) e cursor (■) no canto superior esquerdo.

II.3. UTILIZAÇÃO DA TELA

A tela do televisor utilizada pelo TK-2000 COLOR, tem por função apresentar os dados e instruções que você digita no teclado, bem como os resultados das instruções executadas pelo TK-2000 COLOR.

Existem três modos de operação da tela. Um deles é usado para a apresentação de textos em preto e branco e os outros dois, que serão vistos em capítulos posteriores, são usados principalmente para gráficos coloridos.

No modo texto, a tela pode conter ate 24 linhas de 40 caracteres cada. (fig. II.6)

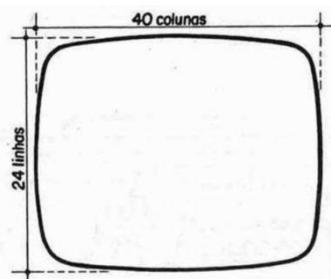


Fig II.6

II.3.1. O sinal de comando (>)

O sinal de COMANDO (>), que você vê na primeira posição da tela (primeira coluna da primeira linha), indica que o TK-2000 COLOR esta pronto para receber instruções através do teclado.

II.3.2. O cursor (█)

Quando o sinal de comando está presente na tela, qualquer carácter digitado e apresentado na posição em que se encontrava o cursor, o qual passa imediatamente para a próxima posição na tela.

Concluindo, o cursor sempre indica a posição da tela para a qual será registrado o próximo caráter digitado no teclado.

II.4 – UTILIZAÇÃO DO TECLADO

A finalidade do teclado (fig. II.4) e permitir que você registre caracteres na tela e na memória do TK-2000 COLOR, e execute determinadas funções.

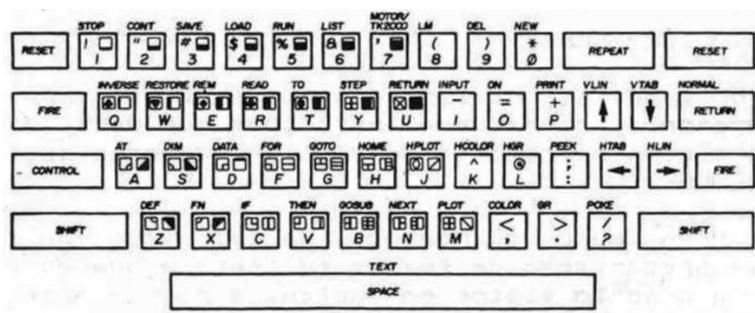


Fig. II.7

II.4.1. Teclas de Caracteres

Os caracteres emitidos por estas teclas podem ser divididos em alfabéticos, numéricos, simbólicos, gráficos e de instruções.

No computador TK-2000 COLOR, as teclas que emitem os caracteres alfabéticos estão dispostos de forma semelhante a usada em uma máquina de escrever. Entretanto, as letras são geradas somente no formato maiúsculo, que é o suficiente para a programação BASIC.

Na computação é indispensável distinguir a letra O do número 0 (zero). Para tanto, o zero é apresentado na tela como um círculo cortado (Ø). Os demais caracteres numéricos não possuem qualquer diferença dos convencionais.

Além dos símbolos normalmente usados na máquina de escrever (sinais de pontuação, \$, &, etc.), o TK-2000 COLOR permite ainda a emissão dos sinais ">" (maior que), "<" (menor que) e @ (arroba).

Existem 50 diferentes símbolos gráficos, e que permitem a execução de tabelas e figuras. Entre estes símbolos, encontram-se também os naipes de baralho, que serão úteis no desenvolvimento de jogos através do computador.

Os caracteres de instruções são na realidade grupos de caracteres emitidos através da digitação de somente uma tecla, que emitem palavras da língua inglesa correspondentes as instruções mais utilizadas pela linguagem BASIC.

Os caracteres emitidos pela simples digitação deste grupo de teclas, são indicados na figura a seguir.

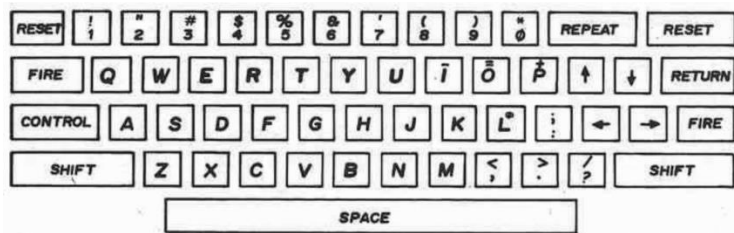


Fig. II.8

Procure digitar alguma destas, por exemplo, o número zero e a letra O, e observe os resultados na tela. Note que as teclas FIRE emitem caráter "." (ponto) e correspondem exatamente a tecla > quando pressionada sozinha.

II.4.1.a. Uso dos Caracteres Gráficos

Antes de se usar os caracteres gráficos deve-se pressionar o conjunto de teclas CONTROL e B simultaneamente e então apertar a tecla SHIFT e uma tecla alfanumérica, que fará com que o símbolo gráfico seja apresentado na tela. Para se utilizar os símbolos gráficos restantes basta manter pressionadas as teclas SHIFT e CONTROL simultaneamente e a seguir digitar a tecla cujo símbolo é desejado.

Quando você desejar voltar ao modo normal de operação do teclado, pressionar novamente o conjunto CONTROL-E.

A seguir será apresentado um programa em BASIC que permite a impressão dos caracteres gráficos.

```
>10 FOR I = 193 TO 242
>20 PRINT I;" ";CHR$(242);CHR$(I)
>25 REM, A FUNCAO CHR$(242) DEVE SER COLOCADA SEMPRE ANTES
    DO COMANDO CHR$(I) PARA SE OBTER A IMPRESSAO DOS
    CARACTERES GRAFICOS
>30 NEXT
```

II.4.2. Teclas de Mudança

As teclas de mudança, quando digitadas sozinhas, não exercem função alguma, porém enquanto pressionadas, modificam a atuação de outras teclas, conforme descrevem os próximos sub-itens.

II.4.2.a SHIFT

A tecla SHIFT faz com que as demais teclas passem a gerar os caracteres representados em suas metades superiores. A figura II.9 indica a posição da tecla SHIFT, bem como os caracteres que as demais teclas passam a gerar, quando esta estiver pressionada.

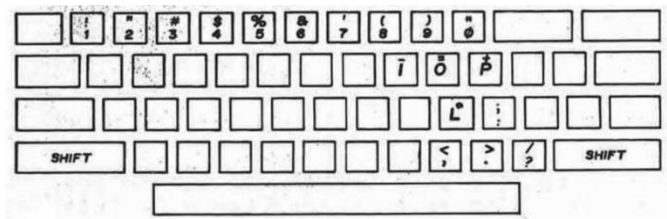


Fig. 11.9

Confira o apresentado, abaixando a tecla SHIFT e digitando alguns caracteres.

II.4.2.b. CONTROL

A tecla CONTROL pressionada em conjunto com SHIFT faz com que os comandos conforme a figura II.10 passem a ser gerados, enquanto estiverem pressionada, conforme indica a figura II.10.

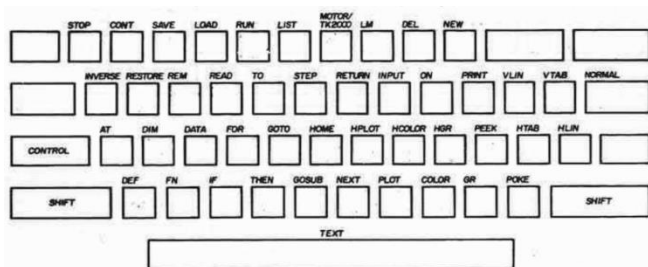


Fig II.10

Note também que certas teclas, quando pressionadas em conjunto com CONTROL, fazem com que o TK-2000 COLOR execute determinadas funções. Por exemplo:

CONTROL + C = Interrompe um programa ou uma listagem.
 CONTROL + X = Cancela a linha que esta sendo digitada.
 CONTROL + G = Gera um beep
 CONTROL + B = Entra e sai do modo gráfico
 CONTROL + S = Para a listagem de um programa BASIC.
 CONTROL + Q = Permite continuar a listagem.

II.4.3. Teclas de Funções

À este grupo pertencem as teclas que, quando pressionadas, ao invés de gerarem caracteres, executam uma função.

II.4.3.a. RETURN

Ao operar o teclado, os caracteres são apresentados na tela e retidos na memória, num lugar especialmente dedicado para tal função, porem eles não são interpretados, até que se pressione a tecla RETURN.

A tecla RETURN indica ao TK-2000 COLOR que a linha de

comando foi terminada. Quando pressionada, os caracteres a direita do cursor são suprimidos e os da esquerda são examinados. Se formarem uma instrução que o TK-2000 é capaz de entender, a ação apropriada é executada, caso contrário, é emitido um sinal sonoro e a mensagem "?SINTAXE #ERRO" é apresentada na tela, caso em que o comando, com a devida correção, deve ser novamente digitado. A localização desta tecla é apresentada na figura II.11.

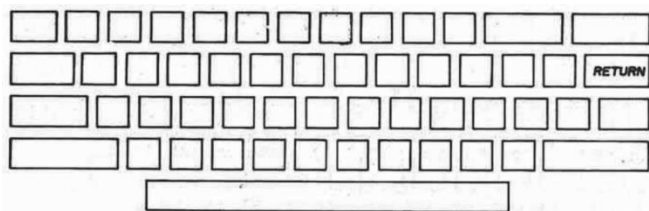


Fig. II.11

II.4.3.b RESET

Quando as teclas RESET são pressionadas simultaneamente, qual quer operação que o TK-2000 COLOR esteja fazendo é interrompida e o símbolo de comando (>) é apresentado na tela permitindo a entrada de uma nova instrução (fig. II.12)

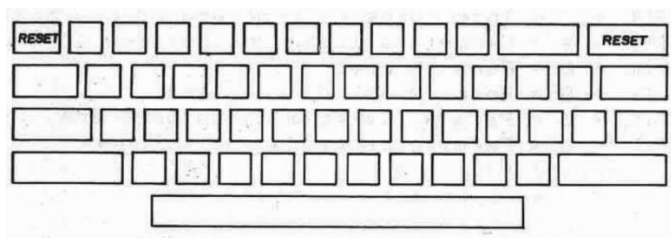


Fig. II.12

Esta tecla somente deverá ser utilizada como última opção, quando não for possível sair de determinada situação de outra forma.

II.4.3.c. Teclas de Movimentação do Cursor

São apresentadas na figura II.13 as teclas de movimentação do cursor.

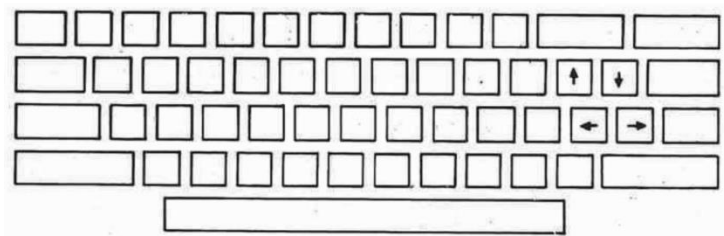


Fig. II.13

Cada uma destas teclas movem o cursor, uma posição na direção indicada pela flecha: uma coluna para a direita (→), uma coluna para a esquerda (←), uma linha para cima (↑), e uma linha para baixo (↓).

Lembre-se, numa determinada linha, ao se pressionar a tecla RETURN, sô serão considerados os caracteres que estiverem a esquerda do cursor, ou de outra forma, os caracteres que estiverem da posição do cursor para a direita serão ignorados pelo TK-2000 COLOR, não sendo então registrados na memória.

II.4.4. REPEAT

Pressionando-se qualquer tecla de caracteres alfanuméricos e simultaneamente a tecla REPEAT, o carácter em questão se reproduzirá automaticamente.

II.5. O GRAVADOR CASSETTE

Conforme já dissemos, este equipamento é opcional (embora muito útil), e permite o uso de programas já vendidos em fita cassete ou a gravação de seus próprios programas.

II.5.1. Cuidados com a Fita Cassete

Antes de começarmos a descrever o uso do gravador cassete, é muito importante que você tenha em mente certos cuidados que se deve ter com as fitas cassete. Para você sentir a importância destes cuidados, lembre-se que, uma pequena alteração numa fita musical é quase imperceptível ao ouvinte, ao

passo que, um dado alterado num programa pode inutilizá-lo totalmente.

Os principais cuidados com as fitas cassete são:

1. por mais limpa que esteja sua mão, nunca toque na superfície da fita;
2. quando as fitas não estiverem em uso, guarde-as sempre dentro da respectiva caixa, evitando assim que sejam expostas a poeira e a umidade.
3. nunca permita que as fitas fiquem expostas à luz direta do Sol, ou de forma geral que seja guardadas em locais que possam vir a se esquentar.
4. não aproxime a fita de campos magnéticos, por exemplo, os gerados por motores ou transformadores elétricos, ímãs permanentes (como os contidos na maioria dos alto falantes), aparelhos de televisão, etc.

II.5.2. Organização das Fitas

Para que você não perca um grande tempo procurando encontrar em que fita está determinado programa, é importante manter sua fitoteca organizada. Desta forma, utilize em todas as fitas etiquetas indicando os programas que cada uma contém.

II.5.3. Proteção contra Regravação

Toda fita cassete tem dois orifícios em sua face posterior. A maioria dos gravadores, quando localizam a existência destes orifícios, não permitem a gravação. As fitas cassete virgens possuem linguetas cobrindo estes orifícios, permitindo assim sua gravação. Você pode proteger programas importantes quebrando estas linguetas.

Para determinar qual lingueta deve ser removida, segure a fita cassete com os orifícios voltados para você e a face que se quer proteger para cima. Remova então a lingueta do lado direito. (fig. II.4)



Fig.11.14

II.5.4. Ajuste do Gravador

Certifique-se que seu gravador não marca a superfície das fitas já que isto pode gerar erros ou estragar a fita.

Depois de conectar o gravador de acordo com as instruções do "Manual de Instalação", pressione a tecla RETURN, que causará a apresentação do sinal ">" na margem esquerda da tela, indicando que está pronto para novas instruções.

Para enviar informações contidas na fita para o computador, o gravador deverá estar acionado no modo PLAY. Existe porém uma faixa de volume certo para que o TK-2000 COLOR possa entender estas informações. Para deste volume, o TK-2000 COLOR avisará o ajuste inadequado (acima ou abaixo do correto) através de uma mensagem de erro na tela.

Para obter o volume certo coloque inicialmente o gravador no modo PLAY, com um volume em torno de 80% e verifique se as informações são enviadas corretamente. Se isto não ocorrer, repita a operação com o volume um pouco mais baixo e assim sucessivamente, até que o TK-2000 COLOR emita um sinal sonoro indicando a correta regulagem do volume.

Se sentir ainda alguma dificuldade, tente ajustar a posição da cabeça do gravador, com uma chave de fenda, sobre o parafuso que controla essa posição: O critério a ser usado no ajuste é o da obtenção do som mais puro e alto que você possa obter.

II.5.5. Carga de Programa .

O TK-2000 COLOR pode ser carregado com dois tipos de fitas cassete gravadas: as fitas no formato TK-2000 e as no formato APPLE II. Note que nem todas as fitas no formato APPLE II podem ser utilizadas no TK-2000 COLOR, de modo que, se você possui fitas nesse formato, deverá tentar usá-las e verificar se funcionam ou não.

Dois comandos podem ser usados para carregar um programa no TK-2000 COLOR:

LOADT ou LOADA

LOADT "nome-de-programa"- para carregar programas no formato TK-2000 COLOR

LOADA - para carregar programas no formato APPLE II

OBSERVAÇÃO: Em LOADT, o **nome-de-programa** é opcional (somente necessário quando a fita tiver mais do que um programa) e corresponde ao nome de um programa ou de um conjunto de dados específicos, que conforme dito tem no máximo 31 caracteres (exemplo: JOGO I, CONTAS, DERITO, ANTONI, etc) e deve ser sempre digitado entre aspas.

Uma das vantagens de se usar o comando LOADT é que durante a carga do programa, aparece no vídeo o número de registros do arquivo (na base hexadecimal) e também um contador progressivo que indicará quando a carga do arquivo for completada. Se o arquivo foi carregado sem erro, o sinal OK é apresentado na tela e, em caso contrário surge uma mensagem de erro.

E outra vantagem é a possibilidade de gravar e ler com nome específico para cada programa. O nome pode ter no máximo 6 caracteres.

O seu computador TK-2000 COLOR vem acompanhado de uma fita cassete com um programa denominado TESTE. Este tem como finalidade constatar o perfeito funcionamento do equipamento. Agora siga os seguintes passos para carregar o programa "TESTE" no seu TK-2000 COLOR. Para cada posição do controle de volumes sugerimos o seguinte:

1. Rebobine a fita até o início

2. Digite:

>LOADT "TESTE"

em seguida pressione a tecla RETURN

3. Pressione o botão PLAY do gravador.

Após o terceiro passo, o cursor desaparecerá da tela. Aguarde então cerca de 15 segundos, após o que surgirá a mensagem:

TESTE 14 00 WAIT

onde TESTE corresponde ao nome do programa, 14 ao número total de blocos do programa (em hexadecimal), 00 ao contador progressivo e WAIT à mensagem indicando que o programa está sendo lido e que você deve aguardar.

Após algum tempo, é apresentado então na tela:

TESTE 14 14 WAIT OK

>■

Neste momento o programa está corretamente carregado e pronto para uso.

II.5.5.a. Erro na carga do "TESTE"

Existem quatro possibilidades que podem ocorrer antes que um programa tenha sido carregado com sucesso:

- a) Surge a mensagem "SINTAXE #ERRO"

- b) Nada ocorre
- c) A mensagem "ERR" ou "ERRO" é apresentada
- d) O alarme do computador soa (beep), porém nada mais acontece

A mensagem apresentada no caso "a" indica que você digitou errado o comando LOADT ou esqueceu de incluir o nome "TESTE" entre aspas ("TESTE").

No caso "b" ou "c", assegure-se inicialmente que você aguardou o intervalo de 15 segundos. Se o símbolo de comando (>) ou cursor não aparecerem, e o TK-2000 COLOR não responder a o teclado, pressione a tecla RESET, aumente um pouco o volume do gravador e repita os três passos indicados para a carga do programa. Pode acontecer também que o cursor apareça na tela imediatamente após a execução do comando LOADT, sem que qualquer outro fato ocorra, Neste caso, desligue e ligue novamente o TK-2000 COLOR (através da chave LIGA/DESLIGA), repetindo em seguida a operação de carga do programa.

Quando o TK-2000 COLOR emitir o sinal sonoro (beep) e a tela apresentar o sinal de comando PRONTO (>) e o cursor, você está no caminho certo. Caso somente o sinal sopro seja emitido aguarde por mais 15 segundos. Se surgir uma mensagem de erro, como a apresentada no caso c, siga o procedimento citado no parágrafo anterior. Aparecendo o sinal de comando e o cursor, aperte o botão de STOP do gravador e rebobine a fita. Marque a posição de volume do gravador, para que possa voltar facilmente a ela sempre que quiser carregar um programa. Digite então:

RUN RETURN

II.5.5.b Procedimento Normal para Carga de Programas

Uma vez que o volume do gravador já esteja corretamente regulado, já que o volume pode variar de fita para fita, os passos usuais para carga de um programa são os seguintes:

1. Rebobine a fita
2. Digitar LOADT "nome-de-programa"
3. Pressionar o botão PLAY do gravador
4. Pressionar RETURN (o cursor desaparece da tela). Nada acontece por 5 a 2% segundos, e então soa o alarma (beep) do TK-2000 COLOR. Isto significa que as informações da fita começaram a ser carregadas no TK-2000 COLOR. Após mais algum tempo, dependendo da quantidade de informação da fita, o alarme do TK-2000

COLOR soa novamente (este tempo não costuma ser maior do que alguns segundos), e o sinal de comando juntamente com o cursor reaparecem na tela.

5. Apertar o botão STOP do gravador e rebobinar a fita.
6. Digitar RUN e pressionar RETURN. Após este último passo, o programa começará a ser executado.
7. Como opção ao item 3 utilize o cabo remote do seu gravador e ligue-o através do comando MOTOR antes da etapa 4. O comando MOTOR é específico para ligar ou desligar os gravadores e está descrito com maiores detalhes na seção VIII.4.3.

FAÇA O SEU RESUMO

1. O sinal apresentado na tela informa que o TK-2000 COLOR está apto para receber instruções, enquanto o indica a posição em que o próximo carácter digitado deverá surgir.
2. No TK-2000 COLOR o zero é representado pelo símbolo "...."
3. Às teclas de mudança, que são e enquanto pressionadas alteram a função das teclas de caracteres.
4. Os caracteres apresentados na tela, e que estejam a do cursor, são interpretados quando a tecla é digitada.
5. Qualquer operação que o TK-2000 COLOR esteja realizando é interrompida quando as teclas são pressionadas.
6. As teclas de são muito úteis para correção de erros de digitação, ou para modificar informações já registradas.
7. O comando e seguidos opcionalmente do nome-de-arquivo entre aspas (é necessário quando a fita tiver mais do que um arquivo), são usadas para carregar programas de fitas nos formatos e respectivamente no computador TK-2000 COLOR.
8. Se um comando é digitado com erro, surge a mensagem na tela.
9. Após a correta regulação do volume do gravador, a sequência necessária para a carga de um programa é 1) a fita; 2) Digitar ".....", e pressione 3) Pressionar a tecla do gravador:
10. Os caracteres gráficos do teclado são acionados pressionando-se o conjunto de teclas e simultaneamente. Um novo pressionamento deste conjunto de teclas faz com que o teclado volte do seu estado normal.
11. Uma vez acionados os caracteres gráficos do teclado, os símbolos que se encontram no canto superior esquerdo das teclas alfanuméricas são obtidos através do pressionamento desta tecla em conjunto com é para o uso dos símbolos do canto superior direito das teclas alfanuméricas, deve-se somente a tecla desejada.

Respostas: 1. >, █; 2. 0; 3. CONTROL, SHIFT; 4. esquerda, RETURN; 5. RESET; 6. Movimentação do cursor; 7. LOADA, LOADT, APPLE, TK-2000 COLOR; 8. SINTAXE #ERRO; 9. Rebobine, LOADT, nome de arquivo, RETURN, PLAY; 10. CONTROL,B; 11.SHIFT

CAPÍTULO 3

CAPITULO III – O TK-2000 COLOR COMO CALCULADORA

Quem imagina que a operação de um computador é algo complexo e exige um profundo estudo, verificará que, através do chamado MODO IMEDIATO da linguagem BASIC, poderemos solucionar expressões aritméticas com a mesma facilidade (ou até mais facilmente) que pelo uso de uma calculadora convencional.

III.1. COMO USAR O MODO IMEDIATO

Quando você utiliza o BASIC no TK-2000 COLOR, ele já está pronto para operar no MODO IMEDIATO (também chamado Direto ou Modo Calculadora). Nesta forma de operação, o computador responde imediatamente as instruções desejadas. Digite a instrução PRINT, um espaço e o número 385, conforme se vê abaixo:

```
>PRINT 385
```

pressione agora a tecla RETURN. O número 385 surge então na linha seguinte do vídeo. Pronto, você acabou de executar uma instrução na linguagem BASIC. Difícil não?

Os itens a seguir apresentam a forma de se solucionar expressões aritméticas através do MODO IMEDIATO da linguagem BASIC.

III.2. AS OPERAÇÕES ARITMETICAS

Para se executar uma operação aritmética através do MODO IMEDIATO, basta emitir o comando PRINT, um espaço e em seguida digitar a expressão desejada em linguagem BASIC.

Porém lembre-se, o computador, embora rápido e eficiente, é incapaz de deduzir qualquer coisa. Portanto, você deve usar somente os símbolos matemáticos definidos dentro do BASIC. A tabela a seguir apresenta tais símbolos bem como a ordem de precedência dos operadores.

| | | | |
|-------|-------------|---------------|-----------|
| ***** | | | |
| * | ORDEM DE | TIPO DE | SÍMBOLO |
| * | PRECEDENCIA | OPERAÇÃO | UTILIZADO |
| ***** | | | |
| * | 1 | Potenciação | ^ |
| ----- | | | |
| * | | Multiplicação | * |
| * | 2 | ----- | ----- |
| * | | Divisão | / |
| ----- | | | |
| * | | Adição | + |
| * | 3 | ----- | ----- |
| * | | Subtração | - |
| ***** | | | |

TABELA III.1

Exemplo : suponha que você queira somar o número 3 ao 5. Para isso, digite:

```
>PRINT 3 + 5 (RETURN)
8
```

o número 8 deverá então surgir na próxima linha do vídeo. Se você desejar subtrair deste resultado o número 6, prossiga assim:

```
>PRINT 8 - 6 (RETURN)
2
```

o resultado destas duas operações (2) pode ser obtido diretamente, digitando-se numa só linha:

```
>PRINT 3 + 5 - 6 (RETURN)
2
```

Atenção: 1. Na sequência deste capítulo, tal qual acontece no vídeo, as respostas, após o pressionamento da tecla RETURN serão apresentadas na linha abaixo de onde se encontra o comando.

2. Os espaços emitidos entre números e operadores, tem somente a função estética. Desta maneira, o comando acima poderia ser digitado na forma:

```
>PRINT 3+5-6
2
```

Execute agora as seguintes operações:

```
>PRINT 672 + 254 - 511 (RETURN)
415
>PRINT 827 - 12 + 88 - 706
197
```

As operações de multiplicação e divisão são similares. Por exemplo, para multiplicar o número 4 pelo 5 digitamos:

```
>PRINT 4 * 5 (RETURN)
20
```

Se quisermos dividir o número 20 (resultado da operação acima), por 2 executamos:

```
>PRINT 20/2
10
```

O mesmo resultado das duas operações apresentadas pode ser obtido numa única linha:

```
>PRINT 4 * 5 / 2 (RETURN)
19
```

Seguem mais alguns exemplos:

```
>PRINT 8 * 2 / 4 * 3 (RETURN)
12
>PRINT 2 * 6 / 3 * 5 / 10 (RETURN)
2
```

Para elevarmos o número 4 a quinta potência ($4*4*4*4*4$) podemos fazer:

```
>PRINT 4*4*4*4*4 (RETURN)
1024
ou simplesmente,
>PRINT 4 ^ 5
1024
```

Da mesma forma podemos elevar 5 ao quadrado, ou 3 ao quadrado, ou qualquer número desejado:

```
>PRINT 5 ^ 2 (RETURN)
25
PRINT 3 ^ 2 (RETURN)
9
```

Agora, digite o comando abaixo, porém, antes de pressionar a tecla (RETURN), tente você mesmo encontrar a resposta.

```
>PRINT 5 + 3 * 4 ^ 2
```

Já calculou? Pressione então RETURN. Se o seu resultado foi 53 e por que você já está sabendo usar 4 ordem de precedência dos operadores. Se o resultado foi diferente, vamos fazer juntos a sequência das operações, passo a passo. Consultando-se a tabela 5.1, vemos que a operação prioritária (precedência 1) e a potenciação, logo a primeira operação a ser feita é:

$$4 ^ 2 = 16$$

a segunda é a multiplicação

$$3 * 16 = 48$$

e a ultima é a soma :

$$5 + 48 = 53$$

Procure encontrar o resultado dos comandos (antes de pressionar a tecla RETURN):

```
>PRINT 2 ^ 3 + 5 * 2
18
>PRINT 4 / 2 + 3 ^ 2 * 2 - 20 / 2
10
>PRINT 8 + 2 ^ 3 * 4 - 4 ^ 2
24
```


III.3. USO DAS FUNÇÕES MATEMATICAS ATRAVES DO MODO IMEDIATO

O TK-2000 COLOR permite diversas funções matemáticas. Uma vez que este tema será desenvolvido em capítulo posterior. nosso objetivo neste item é apenas mostrar como estas funções são usadas no MODO IMEDIATO.

Vamos usar como exemplo a raiz quadrada, que no BASIC corresponde ao comando SQR (X). Desta forma, se desejarmos extrair a raiz quadrada de 16 deveremos digitar:

```
>PRINT SQR (16) (RETURN)
4
```

Observação: a operação acima pode também ser executada, de forma mais lenta (em termos da operação do computador) através do comando :

```
>PRINT 16 ^ 0.5
4
```

III.4 A FORMA DOS NUMEROS NO COMPUTADOR TK-2000 COLOR

Por motivos de padronização e limitações do próprio computador são estabelecidas certas regras para a apresentação de números pelo TK-2000 COLOR.

REGRA 1: Zeros a direita do número após o ponto decimal, ou zeros a esquerda do número são eliminados.

Exemplo:

```
>PRINT 045.340 (RETURN)
45.34
```

REGRA 2: Números muito próximos de zero, entre $-3E-39$ e $3E-39$ (ou em nossa notação $-3*10^{-39}$ e $3*10^{-39}$) são arredondados para zero. Procure você mesmo executar alguns exemplos.

REGRA 3: Números com partes inteiras e decimais, com mais de 9 algarismos, são arredondados.

Exemplo:

```
>PRINT 985788.6898
985788.69
```

REGRA 4: Números com mais de 9 algarismos inteiros passam a ser representados através da Notação Científica.

Exemplo:

```
>PRINT 1234567890  
1.23456789E+09
```

repare que E+09 representa 1000000000 (ou $*10^9$), portanto como sabemos:

$$1234567890 = 1.23456789 \times 10^9$$

Se você não está habituado ao uso da Notação Científica não se preocupe, quando for necessário você aprenderá facilmente a lidar com ela.

Repare que as regras apresentadas não irão prejudicar qual quer tipo de operação no TK-2000 COLOR.

E possível fugir as regras apresentadas? Sim, vamos utilizar os mesmos exemplos apresentados durante a explicação das regras e digitar os números correspondentes entre aspas:

```
>PRINT "045.340" (RETURN)  
045.340  
>PRINT "985788.5898" (RETURN)  
985788.5898  
>PRINT "1234567890" (RETURN)  
1234567890
```

O uso das aspas será descrito mais adiante, por enquanto e suficiente entendermos esta explicação.

FAÇA O SEU RESUMO

1. Para usar o MODO IMEDIATO basta usar o no TK-2000 COLOR
2. Através do o computador pode ser usado como se fosse uma calculadora.
3. As operações matemáticas, de acordo com a ordem de precedência são: 1)....., 2)..... e , 3) e ; e os símbolos usados no BASIC para representá-las são respectivamente: e
4. Para se extrair a raiz quadrada do número 36 devemos digitar (36)
5. O TK-2000 COLOR aceita um número com parte inteira e decimal de até dígitos.
6. Números com mais de nove dígitos inteiros são apresentados pelo TK-2000 COLOR através da
7. Para não modificar a forma do TK-2000 COLOR representar qualquer tipo de número, devemos digitar este número entre

respostas:

1. BASIC; 2. MODO IMEDIATO; 3. Potenciação, Multiplicação e Divisão, Adição e Subtração- símbolos: ^, *, /, +, -; 4. PRINT, SQR, RETURN; 5. 9; 6. Notação Científica; 7. aspas.

CAPÍTULO **4**

CAPITULO IV - INICIANDO A PROGRAMAÇÃO

Conforme dissemos, um programa é constituído por um conjunto de instruções que se interagem com um determinado objetivo.

No final deste capítulo, você já será capaz de desenvolver seus primeiros programas, que apesar de simples, terão interessantes resultados.

IV.1. AS PRIMEIRAS INSTRUÇÕES

Para desenvolver seus programas, este item descreve algumas instruções muito utilizadas na linguagem BASIC. Se na definição inicial das instruções surgirem dúvidas, não se preocupe pois, na sequência do capítulo os exemplos tornarão suas utilizações claras.

IV.1.1. NEW

Esta instrução faz com que todo conteúdo da memória RAM do TK-2000 COLOR seja apagada. Por exemplo, após a utilização de algum programa, você deve digitar.

```
>NEW          (RETURN)
```

para iniciar a carga de um novo programa, garantindo assim a não interferência das instruções anteriormente carregadas.

IV.1.2. PRINT

À instrução PRINT, já utilizada no capítulo anterior, faz com que o programa apresente determinado conteúdo na tela, Ela pode assumir diversas formas conforme podemos ver a seguir:

1) PRINT "conteúdo"

Quando o conteúdo após PRINT é digitado entre aspas, ele aparece exatamente da forma que foi digitado.

Digite os seguintes comandos no computador e confira os resultados, que apresentamos abaixo da linha de comando do nosso exemplo.

```
>PRINT "COMPUTADOR TK-2000 COLOR"      (RETURN)
COMPUTADOR TK-2000 COLOR
PRINT "3 + 5"                          (RETURN)
3 + 5
```

Repare que no segundo exemplo o TK-2000 COLOR não executou a operação, apresentando-a exatamente da forma em que foi digitada.

Importante: Sempre que quisermos apresentar um conjunto de caracteres alfabéticos (por exemplo um nome) ou um conteúdo fixo (inalterável) na tela, este deverá ser digitado entre aspas, após um comando PRINT. No próximo capítulo este assunto será melhor detalhado.

2) PRINT conteúdo

Este modo é utilizado para a apresentação de um número pré-determinado, ou o resultado de uma operação. Execute no TK-2000 COLOR os exemplos.

```
>PRINT 385                                (RETURN)
385
>PRINT 3 + 5                              (RETURN)
8
>PRINT COMPUTADOR                         (RETURN)
0
?SINTAXE #ERRO
>PRINT "3 + 5 ="; 3 + 5                   (RETURN)
3 + 5 = 8
```

Note que no segundo exemplo, ao invés do TK-2000 COLOR executar o comando, surgiu o carácter é e uma mensagem de erro. Este procedimento ocorreu pois, como já dissemos, os grupos de caracteres alfabéticos devem ser digitados entre aspas. A mensagem de erro ocorreu porque, como veremos adiante, o BASIC não atribuiu á palavra computador um nome de variável.

3) PRINT conteúdo, conteúdo, conteúdo.....

Se a vírgula é digitada entre dois conteúdos, eles são apresentados com uma tabulação padrão de 15 espaços entre cada.

Exemplo:

```
>PRINT 53, 67, 230                        (RETURN)
53    67    230
>PRINT "UM", "DOIS", "TRES"              (RETURN)
UM    DOIS  TRES
```

4) PRINT conteúdo; conteúdo

O ponto e vírgula entre dois conteúdos faz com que eles sejam apresentados sequencialmente, sem que haja intervalo entre os mesmos.

Exemplo:

```
>PRINT 53;67
5367
>PRINT "5 + 3 = "; 5 + 3, "7 + 2 = "; 7 + 2
5 + 3 = 8    7 + 2 = 9
```

Note que, no último exemplo, são resumidos os quatro modos básicos de utilização do comando PRINT. No decorrer dos capítulos serão vistos ainda, mais algumas formas de utilização desta instrução.

Observação: Na digitação de um programa qualquer ao invés de se digitar a palavra PRINT, é possível substituí-la por um sinal de interrogação (?).

Por exemplos:

```
>?"BOM DIA !"
```

```
BOM DIA !
```

```
>?"3 + 4 = "; 3 + 4
```

```
3 + 4 = 7
```

Note que agora o TK-2000 COLOR interpretou o caráter "?" como PRINT.

IV.1.3. HOME

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

Digite:

```
>HOME (RETURN)
```

e observe o resultado.

IV.1.4. END

A instrução END faz com que a execução de um programa seja interrompida ou finalizada. Esta instrução, bem como as próximas duas serão explicadas em mais detalhes no item IV.2.

IV.1.5. LIST

Através desta instrução todas as linhas de comando do programa são apresentadas, ou seja, listadas na tela.

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

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

onde X é a primeira linha que se deseja listar e Y é a última linha cuja apresentação é desejada. X e Y correspondem aos números de linha.

IV.1.5.a. CONTROL-S

Quando a listagem de um programa é muito longa, pode-se interrompê-la mantendo-se pressionada a tecla CONTROL e digitando-se a tecla S.

IV.1.5.b. CONTROL-Q

Após uma listagem ter sido interrompida por CONTROL-S pode-se dar prosseguimento à mesma através de CONTROL-Q.

IV. 1.6. RUN

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

As vezes, para analisar um programa, é conveniente que este seja executado a partir de uma linha de programa. Para tanto, podemos emitir o comando:

RUN XYZ

representando XYZ uma linha genérica de programa a partir da qual a execução é desejada.

IV.2. O MODO PROGRAMADO

Você já aprendeu a usar o TK-2000 COLOR através do MODO IMEDIATO, que apesar de poder auxiliá-lo em algumas tarefas tão bem quanto uma calculadora, é muito limitado em termos da capacidade real de um computador. Neste item, você começará a ter contato com a programação, através da qual o computador poderá ter seu potencial realmente explorado.

IV.2.1. Numeração das linhas

Para se executar um programa, é necessário que cada linha deste seja relacionada a um número inteiro. Não existem números fixos a serem utilizados para esta numeração porém, normalmente as linhas devem ser numeradas em ordem crescente, no decorrer do desenvolvimento do programa.

Por exemplo, digite a linha:

>10 PRINT "O TK-2000 COLOR INFORMA"

pressione agora a tecla RETURN. Como você pode perceber, o cursor foi para a próxima linha, porém a instrução não foi executada. Digite agora a palavra RUN e pressione a tecla RETURN. Neste momento a tela apresentará:

>10 PRINT "O TK-2000 COLOR INFORMA"

>RUN

O TK-2000 COLOR INFORMA

O TK-2000 COLOR acabou de executar um passo de programa.

Digite então:

```
>20 PRINT "20 + 25 = "; 28 + 25      (RETURN)
>RUN                                  (RETURN)
```

Como resultado surge na tela:

```
O TK-2000 COLOR INFORMA
20 + 25 = 45
```

Embora normalmente as linhas de programa sejam desenvolvidas em ordem crescente, nada impede que, na sequência da programação, sejam acrescentadas linhas no início ou entre as linhas já definidas. Suponha que se deseje acrescentar no início do nosso programinha a frase "ATENCAO". Basta para isso que se digite a linha:

```
>3 PRINT "ATENCAO"                  (RETURN)
```

A linha de comando acima poderia ter qualquer número entre 1 e 9 (inclusive). Só para brincar mais um pouco, logo após "O TK-2000 COLOR INFORMA" vamos incluir a frase "SOB O PATROCINIO DE (escreva aqui seu nome)". Neste caso, a linha de comando que iremos definir deverá ter um número entre 11 e 19 (inclusive). Por exemplo:

```
>18 PRINT "SOB O PATROCINIO DE (escreva aqui seu nome)"
(RETURN)
```

Observação: não , se incomode com o fato da instrução não poder ser incluída numa só linha. Prossiga a digitação, normalmente, na linha seguinte. Pode-se usar ate 259 caracteres (incluindo brancos) numa mesma linha de comando.

Digite RUN e a tecla RETURN, quando então surgirá:

```
ATENCAO
O TK-2000 COLOR INFORMA
SOB O PATROCINIO DE .....
20 + 25 = 45
```

Se você quiser ver a listagem do seu programa, digite:

```
>LIST                               (RETURN)
```

A tela apresentará:

```
3 PRINT "ATENCAO"
10 PRINT "O TK-2000 COLOR INFORMA"
18 PRINT "SOB O PATROCINIO DE (escreva aqui seu nome)"
20 PRINT "20 + 25 = "; 20 + 25
```

ATENÇÃO:

1. Note que as linhas de comando serão sempre executadas de acordo com a ordem de numeração
2. Não é conveniente que as linhas de comando tenham números seguidos (1,2,3.....), pois é muito frequente a necessidade de incluir uma nova linha de comando entre duas já existentes. Para seguir uma certa padronização, é interessante que, pelo menos no início, se numere as linhas de 10 em 10 (10, 20, 30, 40.....)
3. Se for dado o mesmo número de linha para dois comandos diferentes, prevalecerá a última definição estabelecida. Desta forma, se após executarmos o nosso programinha, digitarmos:

```
>20 PRINT "O HOMEM CHEGOU A LUA "      (RETURN)
```

ao digitarmos novamente,

```
>RUN                                     (RETURN)
```

teremos

ATENCAO

O TK-2000 COLOR INFORMA

SOB O PATROCINIO DE

O HOMEM CHEGOU A LUA

IV.2.2. O Primeiro Programa

O intuito deste item é mostrar que, com algumas instruções básicas, já podemos montar um programa O qual, embora simples, será bastante ilustrativo.

Nosso programa tem por objetivo montar na tela do TK-2000 COLOR uma tabela com os multiplicadores de 1 a 10 dos números 2 e 3. Para isso todas as operações que você deverá executar no seu TK-2000 COLOR estão descritas a seguir, passo a passo.

Atenção: lembre-se que após cada linha de comando você deve pressionar a tecla RETURN para que as informações sejam registradas pelo computador.

Passo 1: limpe a memória do TK-2000 COLOR digitando:

```
>NEW
```

Passo 2: prepare o título da tabela:

```
>10 PRINT "TABELA DE MULTIPLICACAO"
```

Passo 3: forneça uma linha em branco entre o título e a tabela:

```
>20 PRINT
```

Passo 4: apresente na primeira linha os multiplicadores:

```
>30 PRINT , "X2", "X3"
```

Passo 5: prepare agora a tabela:

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

Passo 6: finalize o programa

```
>140 END
```

Passo 7: vamos verificar se em nenhuma linha houve erro de digitação, ou se não esquecemos alguma instrução. Para tanto digite o comando.

```
>LIST
```

Após o que a tela deve apresentar a lista abaixo.

```
10 PRINT "TABELA DE MULTIPLICACAO"  
20 PRINT  
30 PRINT, "X2", "X3"  
40 PRINT 1,1*2,1*3  
50 PRINT 2,1*2,2*3  
60 PRINT 3,1*2,3*3  
70 PRINT 4,1*2,4*3  
80 PRINT 5,1*2,5*3  
90 PRINT 6,1*2,6*3  
100 PRINT 7,1*2,7*3  
110 PRINT 8,1*2,8*3  
120 PRINT 9,1*2,9*3  
130 PRINT 10,1*2,10*3  
140 END
```

Passo 8: suponha que após termos verificado que todas as linhas estejam corretas nos lembremos que o programa deve inicialmente limpar a tela e só então executar a tabela

que deverá ser apresentada a partir do início da tela. Neste caso, devemos incluir uma instrução HOME (que apaga os caracteres e posiciona o cursor na primeira coluna da primeira linha), em qualquer linha de programa de 1 a 9 (inclusive). Podemos digitar portanto:

>5 HOME

E interessante também que haja uma linha de divisão entre os multiplicadores e multiplicando. Para tanto, podemos emitir a ordem:

>35 PRINT "-----"

Agora aciona novamente o comando LIST e verifique a listagem do programa.

Passo 9: conferida toda a lista do programa, pode-se dar início a sua execução através de:

>RUN

Surge então o seguinte, a partir do início da tela:

TABELA DE MULTIPLICACAO

| | X2 | X3 |
|-------|----|----|
| ----- | | |
| 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:

Se você digitou errado alguma instrução, ao invés de ser apresentada a tabela surgirá uma mensagem de erro, indicando a linha na qual este ocorreu, na forma:

?SINTAXE #ERRO EM

Neste caso, digite o comando LIST e, na linha indicada procure localizar o erro. Uma vez localizado o erro, digite novamente a linha corrigida, utilizando o mesmo número da linha anteriormente errada (lembre-se que no caso de duas linhas com a mesma numeração, prevalece a que foi digitada por último).

V.3. USO DE DOIS PONTOS (:) NUMA LINHA DE PROGRAMA.

Através do uso de dois pontos (:) podemos incluir mais do que um comando numa única linha de programa. Por exemplo, no caso do programa apresentado no item anterior poderíamos reunir a linha 5 e 10 digitando:

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

Este artifício também pode ser usado no MODO IMEDIATO.

IV.4. USO DE PONTO E VÍRGULA (;) APOS O COMANDO PRINT

O ponto e vírgula após o comando PRINT faz com que no próximo comando PRINT, o conteúdo seja apresentado na mesma linha. Digite os exemplos a seguir e note a diferença:

EXEMPLO 1:

```
>NEW  
>10 PRINT "TK-2000 COLOR"  
>20 PRINT "TK-2000 COLOR"  
>30 END  
RUN
```

EXEMPLO 2:

```
>NEW  
>10 PRINT "TK-2000 COLOR";  
>20 PRINT "TK-2000 COLOR"  
>30 END  
>RUN
```

FAÇA O SEU RESUMO

1. À instrução faz com que todo conteúdo da memória RAM do computador seja apagada.
2. Para que o programa apresente um conteúdo fixo ou que utilize caracteres alfabéticos deve-se usar a instrução após a qual digitar este conteúdo entre
3. Todos os caracteres são suprimidos e o cursor se posiciona na primeira coluna da primeira linha quando se utiliza a instrução
4. Para se verificar a listagem de um programa emite-se o comando
5. A execução de um programa é iniciada através do comando
6. Para que uma linha de programa seja registrada pelo computador, após a sua digitação deve-se pressionar a tecla
7. Se for dado o mesmo número de linha para dois comandos diferentes prevalece o
8. Para se usar dois comandos em uma mesma linha, eles devem ser separados por (....)
9. Desejando-se que dois comandos exibam os respectivos conteúdos na mesma linha da tela, deve-se digitar um e (.....) após o primeiro.

respostas:

- 1) NEW; 2) PRINT, aspas; 3) HOME; 4) LIST; 5) RUN; 6) RETURN; 7) último; 8) dois pontos (:) 9) PRINT, ponto e vírgula (;)

CAPÍTULO **5**

CAPITULO V - OUTROS COMANDOS

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

V.1. ALGUNS CONCEITOS IMPORTANTES

V.1.1. Dados

O objetivo principal do computador é receber determinados dados, manipulá-los ou processá-los, e então apresentar os novos dados. Desta maneira, é muito importante a forma através da qual a linguagem de programa manipula tais dados. Dentro da linguagem BASIC do TK-2000 COLOR, existem dois principais tipos de dados, que são: cadeias e números. Seguem suas instruções:

V.1.1.a Cadeia (String)

Uma cadeia é qualquer sequência de caracteres incluída entre aspas. Já utilizamos cadeias com o comando PRINT para apresentar mensagens na tela, como por exemplo:

```
>PRINT "TK-2000 COLOR"  
TK-2000 COLOR  
>PRINT "COMPUTADOR"  
COMPUTADOR
```

etc..

As cadeias podem ser formadas por uma-sequência de a 255 caracteres quaisquer, com exceção de aspas (").

V.1.1.b Números

Existem dois tipos de números que podem ser armazenados no TK-2000 COLOR: inteiros. e reais (também chamados números com ponto flutuante), que podem ter parte fracionária, conforme detalhado a seguir.

OBSERVAÇÕES:

1. Assim como nas máquinas de calcular, nos computadores o ponto (.) substitui a vírgula que costumamos utilizar para separar a parte inteira do número de sua parte fracionária. Assim, o número 3,27 deve ser digitado 3.27.
2. Quando um número só tem parte fracionária, não é necessário indicar o zero a esquerda do ponto. Desta forma o número 0,527 pode ser digitado como .527.

especifica a magnitude do número, isto é, o número de casas a direita (se o expoente for positivo) ou a esquerda (se for negativo) que o ponto decimal deve ser deslocado, para dar a sua localização no conjunto dos números reais.

Seguem alguns exemplos de Notação Científica

| Notação Usual | Notação Científica |
|---------------|--------------------|
| 1000000000 | 1E+09 |
| .000000001 | 1E-09 |
| -1000000000 | -1E+09 |
| -0.0000123456 | -1.23456 E-05 |

Os valores limites da Notação Científica são:

números positivos : 1 E-38 e 1.7 E+38

números negativos : -1 E-38 e -1.7 E+38

V.1.3. Arredondamentos

Já mencionamos antes que, os números reais podem ter no TK-2000 COLOR até nove algarismos de precisão. Para números maiores que 1 ou menores do que -1, isto significa que, não mais do que nove algarismos podem ser diferentes de zero. O TK-2000 COLOR aproxima todos os números com mais de nove algarismos. Seguem alguns exemplos para você executar no TK-2000 COLOR.

```
>PRINT 1234567891
```

```
1.23456789 E+09
```

```
>PRINT -123456789123456789
```

```
-1,23456789 E+17
```

```
>PRINT -150000475.75
```

```
-15000476
```

```
>PRINT 90000000.7558
```

```
90000000.8
```

Números fracionários (entre 1 e -1) estão sujeitos as mesmas limitações. Nestes casos, entretanto, os nove algarismos de precisão começam a ser contados a partir do primeiro algarismo diferente de zero a direita do ponto decimal. Por exemplo:

```
>PRINT .1234567894
```

```
.123436789
```

```
>PRINT .001234567894
```

```
1.23456789 E-03
```

V.1.4. Variáveis

Se você conhece álgebra elementar, não terá dificuldade em entender o que é uma variável. Caso contrário, imagine a variável como um rótulo que especifica uma caixa. Suponha que o nome do rótulo que especifica nossa caixa é TI. Se pusermos uma tesoura dentro desta caixa, cada vez que quisermos nos referir a tesoura podemos chamá-la de TI. Assim se quisermos a tesoura poderemos solicitar "Por favor, me dê a caixa TI" ou simplesmente "Por favor, me dê TI". Para guardarmos a tesoura poderemos dizer "Guarde TI". Podemos tirar a tesoura da caixa TI e colocar dentro dela uma régua. Neste caso, quando nos mencionarmos a caixa TI, passaremos agora a nos referir a uma régua e não mais a uma tesoura.

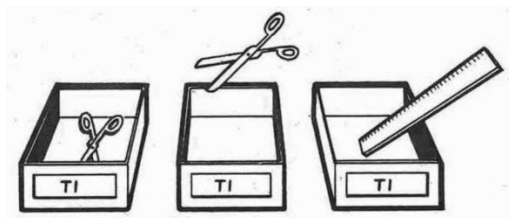
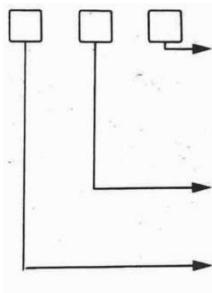


Fig V.2

Como você pode ver, a variável é apenas uma etiqueta ou um nome que se pode associar a qualquer constante. Caso você não tenha captado bem o conceito de variáveis os exemplos do próximo item (LET) deixará a ideia mais clara.

V.1.5. Nome de variáveis no BASIC do TK-2000 COLOR.

O nome de uma variável pode ter qualquer tamanho, mas apenas os dois primeiros caracteres serão reconhecidos como nome pelo BASIC do TK-2000 COLOR. Para se determinar este nome deve-se observar as seguintes regras:



- último caractere: \$: para variável do tipo cadeia
%: para variável do tipo inteira.
Nenhum símbolo para variáveis reais.
- segundo caractere: (opcional) pode ser qualquer letra ou dígito.
- primeiro dígito: é necessariamente uma letra.

Notamos então que podemos ter três tipos variáveis de acordo como tipo de dados que elas contêm: as variáveis tipo cadeia, que são relacionadas com qualquer conjunto de até 255 caracteres (com exceção das aspas), as variáveis entouras que correspondem a números sem parte fracionária, e as variáveis reais associadas a qualquer número inteiro, fracionário ou com parte inteira e fracionária.

V.2. LET

No item anterior discutimos a respeito das variáveis. Pois neste item vamos aprender como, através do comando LET, relacionamos um nome de variável a um valor constante. Digite os exemplos abaixo e verifique os resultados:

OBSERVAÇÕES: 1- Não esqueça de pressionar RETURN após as linhas de comando
2- Os resultados das instruções no TK-2000 COLOR, serão aqui apresentadas logo abaixo das mesmas.

```
>LET TI$ = "TESOURA"  
>PRINT TI$  
TESOURA
```

Atribuímos ao nome-de-variável TI\$ a cadeia TESOURA, Fogo, quando pedimos para o computador apresentar a variável TI\$, ele apresentou a cadeia "TESOURA".

```
>LET TI$ = "REGUA"  
>PRINT TI$  
REGUA
```

Agora, fizemos com que o nome-de-variável TI\$ passe a se relacionar com "REGUA". Portanto, quando ordenamos que o TK2000 COLOR apresentasse a variável TI\$ surgiu a cadeia REGUA. Note que o sinal de cifrão (\$) no final do nome da variável se faz necessário por se tratar de uma cadeia.

Também, de acordo com a regra vista no subitem anterior, podemos utilizar apenas uma letra e o caráter \$ para definir o nome de uma variável tipo cadeia. Exemplo:

```
>LET C$ = "TK-2000 COLOR"  
>PRINT C$  
TK-2000 COLOR
```

Na realidade, O comando LET é opcional, de forma que podemos omiti-lo para executar a associação entre um nome de variável e o seu valor. Por exemplo:

```
>A$ = "COMPUTADOR"  
>PRINT A$  
COMPUTADOR
```

Lembre-se, uma variável pode ser associada a somente um valor por vez, portanto se lhe atribuirmos um novo valor, o antigo será suprimido. Assim, se digitarmos:

```
>A$ = "TUDO BEM"
```

ao digitarmos o comando:

```
>PRINT A$
```

o TK-2000 COLOR apresentará a cadeia:

```
TUDO BEM
```

Agora digite:

```
>LET D8$ = "CINZA"
```

```
>PRINT D8$
```

```
CINZA
```

```
>LET 5F$ = "VERDE"
```

```
?SINTAXE #ERRO
```

```
>█
```

Note que no segundo caso, utilizando o nome 5F\$, quebramos uma das regras de formação dos nomes-de-variável, pois, um número não pode ser o primeiro caráter do nome. Para corrigir este erro podemos digitar por exemplo:

```
>LET F5$ = "VERDE"
```

Veja agora este exemplo:

```
>LET X$ = 7
```

```
?INCOMPATIVEL #ERRO
```

A mensagem de erro foi apresentada porque utilizamos um nome-de-variável do tipo cadeia (com o símbolo de cifrão no lugar do último caráter) para definir uma variável real. Utilizando agora a forma correta.

```
>LET X = 7
```

```
>PRINT X
```

```
7
```

Nos exemplos a seguir, procure prever, antes de executar no computador, o resultado dos comandos.

a)

```
>C = 57
```

```
>PRINT C
```

- b)
- ```
>LET D = 25.27589322
>PRINT D
```
- c)
- ```
>LET E = 725
>LET E = 10
>PRINT E
```
- d)
- ```
>LET F = 5
>LET W = F
>PRINT W
```
- e)
- ```
>LET G = 2
>LET H = 13
>LET I = G + H
>PRINT I
```

Se suas previsões forem corretas, você já deve estar percebendo a versatilidade e utilidade das variáveis. Caso não tenha entendido algum exemplo, releia os itens anterior em deste capítulo e procure fazer seus próprios exemplos para melhor compreensão e fixação.

No decorrer deste livro, dificilmente será usado "LET", porém, se você encontrá-lo em algum programa, saberão que ele significa e que pode perfeitamente ser suprimido, ou seja, os comandos:

```
>LET B$ = " ABC2% #?"
>LET N = 10
```

são exatamente iguais a:

```
>B$ = " ABC2% #?"
>N = 10
```

com a vantagem que a segunda forma é mais rápida de digitar e ocupa menos memória do computador.

V.3. GOTO

Este comando deve ser digitado no formato:

```
>GO TO número
```

onde "número" indica um número de linha para o qual o programa deve ser desviado.

Vamos entender melhor. Para ilustração, suponha que se queira fazer um programa que apresente o nome TK-2000 COLOR no início de todas as linhas da tela. Para tanto digite

inicialmente o comando NEW para limpar toda a memória do computador, e em seguida:

```
>10 HOME
>20 PRINT "TK-2000 COLOR"
>30 GOTO 20
>RUN
```

e veja o resultado na tela. Quando quiser interromper o programa, digite o conjunto de teclas CONTROL C ou as teclas RESET. A figura V.3 indica o fluxo do programa.

```
RUN (dá início a execução do programa)
↓
HOME (limpa a tela e posiciona o cursor em seu início)
↓
PRINT "TK-2000 COLOR" (apresenta TK-2000 COLOR na tela)
↓
GO TO 20 (desvia o programa para a linha 20) -----↑
```

Fig V.3.

Vamos agora modificar um pouco o programa, digitando inicialmente NEW e em seguida:

```
>10 HOME
>20 PRINT "TK-2000 COLOR"
>25 PRINT
>30 GOTO 10
```

Repare agora que o nome TK-2000 COLOR fica piscando na primeira linha da tela. O novo fluxo do programa passa a ser apresentado na fig. V.4

```
RUN
↓
HOME ←-----
↓
PRINT "TK-2000 COLOR"
↓
PRINT
↓
GO TO 10 -----
```

Fig. V.4.

Note que enquanto você não interromper o programa (através de um CONTROL-C) ele continua rodar indefinidamente, pois sempre que alcança a instrução GOTO, o programa é desviado para uma instrução anterior a ela. Portanto, neste caso não adianta incluir uma instrução END após GOTO, pois o programa nunca a atingirá.

V.4. INPUT

O formato utilizado por esta instrução é:
INPUT nome-de-variável (,nome-de-variável,)

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

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

Surge então no canto esquerdo da tela um ponto de interrogação e o cursor, conforme se vê na figura V.5.

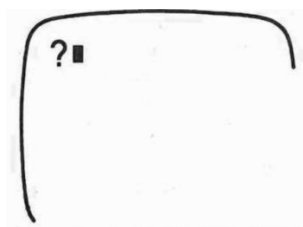


Fig V.5

Portanto a instrução "2 INPUT A\$" executa então as seguintes operações:

1. apresenta um ponto de interrogação na tela;
2. aciona o cursor;
3. espera pela digitação de alguma cadeia. Quando a cadeia é digitada e a tecla RETURN pressionada, faz com que esta seja associada á variável A\$ e sô então permite que o programa prossiga para a próxima linha de comando.

Para prosseguir o programa vamos digitar o nome JOAO e pressionar a tecla RETURN. Observe então o resultado na tela (para interromper o programa, pressione CONTROL-C).

Se houver várias variáveis num INPUT, deve ser digitado o atributo a uma variável por vez repetindo os passos 1,2 e 3 tantas vezes quanto variáveis existir no comando.

Vamos agora a mais um exemplo. Digite os comandos:

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


A figura V.7 indica o fluxograma do programa anterior:

```

RUN (dá início ao programa)
↓
INPUT N$ (apresenta o ponto de interrogação e o cursor na
          tela, aguardando a digitação de uma cadeia para
          associar ao nome de variável N$) -----
↓
PRINT N$ (apresenta na tela o valor associado a N$)
↓
G0 TO 20 (desvia o programa para a linha da instrução INPUT)--

```

Fig. V.6.

Após o ponto de interrogação, digite o nome "PERLA"



Fig. V.7.

Após pressionar a tecla RETURN, a tela passa a apresentar:

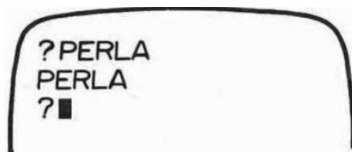


Fig. V.8.

Desta forma, o computador apresentará as cadeias que você digitar após o ponto de interrogação, até que se pressione o conjunto CONTROL C, interrompendo então o programa.

V.4.1. Incrementando a Instrução INPUT

De acordo com o que vimos até agora, o comando INPUT apresenta o ponto de interrogação e o cursor na tela, aguardando então que se digite uma cadeia e a tecla RETURN. É interessante, entretanto, que seja apresentada também uma mensagem ou um incremento indicando qual a cadeia que se deseja. Digite no TK-2000 COLOR o programa abaixo:

```

>NEW
>10 HOME
>20 INPUT "QUAL O SEU NOME ?"; N$
>30 PRINT N$
>40 GOTO 30
>RUN

```

Vamos analisar o que a instrução da linha 20 executa:

- a) interrompe o programa
- b) apresenta na tela a frase "QUAL E O SEU NOME? ";
- c) aguarda a digitação da cadeia, seguida pelo pressionamento da tecla RETURN;
- d) associa a cadeia digitada ao nome-de-variável N\$.

A forma do comando INPUT incrementado é a seguinte:

```
<INPUT> <"> <texto> <"> <;> <variável> ; <variável>...
```

repare que os símbolos menor que (<) e maior que (>) tem apenas efeito ilustrativo para separar cada uma das partes do comando e portanto não devem ser digitados. Também é importante notar que, quando utilizamos o incremento, temos de acrescentar o sinal de interrogação (?) dentro deste, pois o TK-2000 COLOR, neste caso não imprime a interrogação automaticamente.

Ao se digitar o comando RUN e a tecla RETURN, surge na tela do TK-2000 COLOR (fig.V.9)

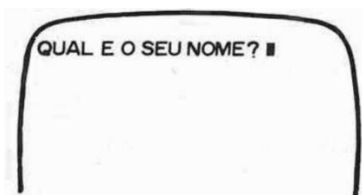


Fig. V.9.

Digitando agora o seu nome (suponha que seja PAULO), o início de todas as linhas da tela serão preenchidas com ele, conforme indica a figura V.10:

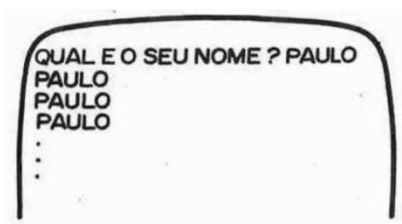


Fig. V.10.

V.4.2. Uso de INPUT com Variáveis Numéricas

Podemos também 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 fluxograma deste programa (fig.V.11)

HOME (limpa a tela)

INPUT A (apresenta o ponto de interrogação e o cursor, aguardando a digitação de um número real a ser relacionado com o nome-de-variável A)

PRINT A (apresenta o valor real associado a variável A)

PRINT (deixa uma linha em branco)

GO TO 20 (desvia o programa para a instrução INPUT A)

Fig. V.11.

Após o comando RUN, o ponto de interrogação e o cursor surgem na tela. Digitando o número 521 e pressionando RETURN, a linha seguinte apresentará este número e após uma linha, o ponto de interrogação e o cursor surgirão novamente na tela. Podemos em seguida digitar outros números como -67, 5.32, etc.. Note que os números não devem ser digitados entre aspas, e se isso acontecer, o TK-2000 COLOR não aceitará e emitirá uma mensagem de erro. A figura V.12 representa o que foi descrito:

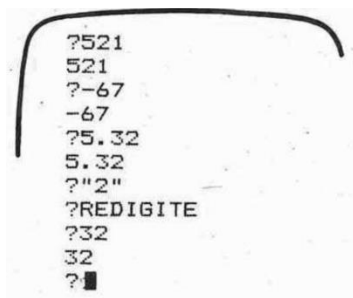


Fig.V.12.

Devemos observar ainda que, podemos associar um número

entre aspas ao nome de uma variável de tipo cadeia.
Por exemplo :

A\$ = "5"

porém, neste caso, este número não poderá ser usado em operações aritméticas, a não ser através de uma determinada instrução que será vista posteriormente.

V.5. REM

Quando utilizamos programas curtos, é bastante fácil lembrarmos, mesmo após algum tempo, o que cada parte deste programa está executando, porém com o passar do tempo e a sofisticação dos programas esta tarefa se torna cada vez mais difícil, ou seja, depois de alguns meses, se um indivíduo com muita boa memória pode lembrar a sequência lógica de um programa de 200 linhas, imagine então vários programas com este número de linhas.

Para solucionar este problema, existe a instrução REM, que permite acrescentar lembretes ou comentários dentro de um programa, sem que o computador tome qualquer conhecimento. Desta forma, se os três primeiros caracteres de uma linha de programa forem REM, durante a execução do programa pula linha é ignorada. 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
```

Note que durante o programa surgirá apenas a pergunta "QUAL. E O NUMERO?" e, após digitado o numero e a tecla RETURN, o valor deste ao quadrado. Porém sempre que listar-nos o programa (através do comando LIST), a linha 10 nos indicará que a finalidade deste é elevar um número ao quadrado.

Os programadores experientes utilizam frequentemente este recurso.

Para concluirmos, digite o programa.

```
>NEW
>10 REM CALCULO DA AREA DE UM RETANGULO
>20 INPUT "QUAL O VALOR DO PRIMEIRO LADO ?";A
>30 PRINT
>40 INPUT "QUAL O VALOR DO OUTRO LADO ? ";B
>50 PRINT A*B
>70 PRINT : PRINT
>80 GOTO 20
>RUN
```

FAÇA O SEU RESUMO

1. O principal objetivo do computador é receber manipulá-los e apresentar então novos
2. Uma é constituída por qualquer sequência de até 239 caracteres incluídos entre
3. Um número real pode ser , parte e parte. ou somente
4. Números com mais de algarismos a esquerda do ponto decimal, ou dentro dos limites de -0.01 e $+0.01$ (exceto zero) são representados através da
5. No TK-2000 COLOR os nomes das variáveis do tipo cadeia devem ter um ou dois caracteres seguidos pelos símbolos de (...) , sendo que o primeiro caráter deve ser necessariamente Os nomes das variáveis reais seguem a mesma regra, porém, o símbolo de não deve ser acrescentado no final.
6. À associação entre um nome-de-variável e seu valor pode ser feita diretamente, ou então através de um comando
7. O comando permite desviar a execução do programa para qualquer número de linha pré-determinada.
8. O comando INPUT executa as seguintes tarefas: interrompe o programa; apresenta um na tela seguido pelo ; aguarda a digitação do a ser associado com o nome-da-variável pré-determinada e o pressionamento da tecla
9. Os termos necessários para se utilizar o comando INPUT incrementado são:
<INPUT> <....> <.....> <....> <.....> <..... .. >
10. Um recurso frequentemente usado para a mais rápida compreensão de um programa é a inclusão de comentários após a instrução , os quais são ignorados pelo computador durante a execução do programa.

respostas: 1. dados, dados; 2. cadeia, aspas; 3. inteiro, inteira, fracionária, fracionário; 4. nove, Notação Científica; 5. cifrão (\$), uma letra, cifrão; 6. LET; 7. GO TO; 8. ponto de interrogação, cursor, dado, RETURN; 9. ", incremento, ", ;, nome-da-variável; 10. REM

CAPÍTULO **6**

CAPITULO VI A INSTRUÇÃO <FOR..NEXT> E TECNICAS DE EDIÇÃO

VI.1. FOR.....NEXT

A instrução FOR.....NEXT é na realidade dividida em duas ordens que são: a ordem FOR e a ordem NEXT. Ela tem por finalidade básica repetir um determinado número de vezes, as instruções que se encontram entre as ordens FOR e NEXT. Por exemplo, digite:

```
>NEW
>10 HOME
>20 FOR I = 1 TO 5
>30 PRINT "TK-2000 COLOR"
>40 NEXT I
>50 END
>RUN
```

Como você pode observar, o nome TK-2000 COLOR foi apresentado cinco vezes na tela. Vamos analisar o formato geral desta instrução:

```
<FOR> <nome-da-variável-real><=><valor inicial da variável> <TO>
<valor final da variável>
:
linhas de instruções a serem repetidas
:
<NEXT><nome-da-variável> (,<nome-da-variável>)
```

À definição da instrução FOR....NEXT deve ser feita então, digitando-se:

- a) FOR e em seguida atribui-se um valor inicial a um nome-de-variável;
- b) TO, após o que o máximo valor que a variável deverá assumir (valor final da variável);
- c) As linhas de instruções que se deseja repetir "n" vezes onde "n" é a diferença entre o valor final menos o valor inicial da variável ou o maior valor inteiro menor que a diferença +1.

$\langle n \rangle = (\langle \text{valor final} \rangle - \langle \text{valor inicial} \rangle);$

- d) Após a última linha que se deseja repetir, a ordem NEXT seguida sozinha ou acompanhada por um ou dois nomes-de-variáveis (este último separada por vírgulas) dependendo do caso.

O funcionamento desta instrução é então:

- a) O computador verifica se o valor inicial da variável é menor que o valor final da mesma;
- b) Se for menor, faz com que o programa prossiga normalmente, até encontrar a ordem "NEXT <nome da variável>", Neste momento o valor inicial da variável é acrescido de uma unidade;
- c) O novo valor da variável (valor inicial + 1) é novamente comparado com o valor final da variável e se ainda for menor, pula para a linha seguinte a FOR e o ciclo se repete: as instruções do programa são executadas normalmente até que se encontre a ordem NEXT, quando então o novo valor da variável é acrescido de mais uma unidade, passando a valer o valor inicial +2, e efetua uma mova comparação entre o valor atual (inicial +2) e o final da variável.
- d) O ciclo acima, se repete até que o valor da variável se iguale ou seja maior que o valor final desta (definido na linha da ordem FOR, após a palavra TO). Neste momento, ele não pula para. A linha seguinte do FOR e sim continua executando na próxima instrução após o NEXT.

Para tornar mais claro o que foi descrito, vamos executar o seguinte programa:

```
>NEW  
>10 HOME  
>20 FOR N = 1 TO 3  
>30 PRINT N  
>40 NEXT N  
>50 PRINT "JA!"  
>60 END
```

Após utilizar o comando RUN, a tela conterà os seguintes caracteres indicados pela figura VI.1:

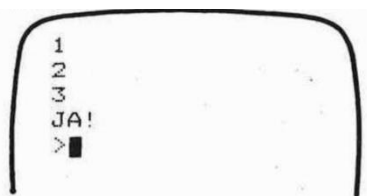


Fig. VI.1.

A figura VI.2 descreve o fluxo deste programa:


```

HOME (limpa a tela e posiciona o cursor em seu início)
FOR N = 1 TO 3 (faz inicialmente N = 1 e compara com 3)
PRINT N (apresenta 1 na tela)
NEXT N (faz N = 1 + 1 = 2 e compara com 3)
PRINT N (apresenta 2 na tela)
NEXT N (faz N = 2 + 1 = 3 e compara com 3)
PRINT N (apresenta 3 na tela)
NEXT N (faz N = 3 + 1 = 4 e compara com 3)
PRINT "JA!" (apresenta Já! na tela)
END (finaliza o programa)

```

Fig. VI.2.

Procure agora observar as seguintes linhas de programa:

a)

```

>10 FOR X = 1 TO 5
>20 PRINT X
>30 NEXT Y

```

b)

```

>10 FOR A$ = 1 TO 7
>20 PRINT A$
>30 NEXT A$

```

Se você as tentou executar, verificou que em ambos os casos foram apresentados mensagens de erros e os programas não funcionaram. Isto porque no caso a, foi usado um nome de variável na ordem FOR e outro nome na ordem NEXT. No caso b, o erro está em se utilizar um nome de variável do tipo cadeia, ou seja, as variáveis utilizadas nas instruções FOR.... NEXT devem ser necessariamente reais e inteiras.

Execute os seguintes programas.

a)

```

>NEW
>10 HOME
>20 FOR I = 1 TO 5
>30 PRINT I
>40 NEXT I
>50 END
>RUN

```

b)

```
>NEW
>10 HOME
>20 FOR K = 3 TO 5
>30 PRINT K
>40 NEXT K
>50 END
RUN
```

c)

```
>NEW
>10 HOME
>20 FOR A = 0 TO 3
>30 PRINT A
>40 NEXT A
>50 END
>RUN
```

d)

```
>NEW
>10 HOME
>20 FOR Z = 1 TO 1
>30 PRINT Z
>40 NEXT Z
>50 END
RUN
```

Os exemplos anteriores apresentarão os seguintes números:

- a) 1, 2, 3, 4 e 5
- b) 3, 4 e 5
- c) 0, 1, 2 e 3
- d) 1

Pode-se ainda reunir várias ordens numa única linha de programa. Desta forma os exemplos a e b poderiam também ser escritos da seguinte forma:

a)

```
>NEW
>10 HOME
>20 FOR I = 1 TO 5 : PRINT I : NEXT I
>30 END
```

b)

```
>NEW
>10 HOME
>20 FOR K = 3 TO 5 : PRINT K : NEXT K
>30 END
```

Verifique que, quando várias instruções são digitadas numa única linha (sempre separadas por dois pontos), elas são executadas da esquerda para a direita da linha. Por uma questão de estética, é conveniente (mas não necessário), que antes e depois dos dois pontos sejam deixados um espaço. Tente agora você mesmo escrever os exemplos c e d num menor número de linhas.

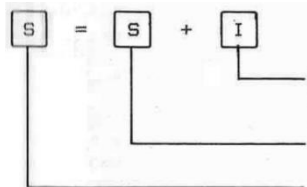
O potencial desta instrução é muito grande. Nesta seção daremos um exemplo da aplicação de FOR.....NEXT num artifício matemático e descreveremos a forma de utilização desta para se determinar um "tempo de espera" dentro de um programa. Outras aplicações serão vistas em capítulos posteriores, porém estaremos longe de esgotar o assunto.

VI.1.1.1. Uso de FOR.....NEXT como Artifício Matemático

O programa exemplo que iremos descrever 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 S = 0
>40 FOR I = 1 TO N
>50 S = S + I
>60 NEXT I
>70 PRINT "A SOMA E ";S
>80 GO TO 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. À linha 50, entre os comandos FOR e NEXT é executada para I = 1, I = 2, I = 3 e assim por diante até I = N e o novo valor de S passa a ser:



valor mais recente de I, definido pela instrução FOR... «.»«NEXT
último valor atribuído a S
novo valor atribuído a S

Por exemplo, suponha que se atribua o valor 3 para a variável N, Então S terá os valores:

linha 30 S = 0

execução da linha 50 S = S + I = 0 + 1

2 execução da linha 50 $S = S + I = 1 + 2 = 3$

última execução da linha 50 $S = S + I = 2 + 3 = 5$

Vamos iniciar a execução do programa (através do comando RUN). A figura VI.3 apresenta alguns exemplos:

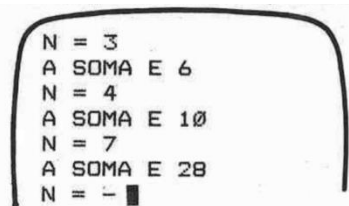


Fig.VI.3

VI.1.2. Uso de FOR....NEXT como Tempo de Espera

Execute no seu TK-2000 COLOR os dois exemplos a seguir (para interrompê-los digite CONTROL C):

a)

```
>NEW
>10 HOME
>20 PRINT "TK-2000 COLOR"
>30 GOTO 10
>RUN
```

b)

```
>NEW
>10 HOME
>20 FOR I = 1 TO 500
>30 NEXT I
>40 PRINT "TK-2000 COLOR"
>50 FOR I = 1 TO 500
>60 NEXT I
>70 GOTO 10
>RUN
```

OBSERVAÇÃO: Após digitar RUN, aguarde um instante para que surja o nome TK-2000 COLOR na tela.

Como você pode observar, no primeiro caso não se consegue ler o nome TK-2000 COLOR que pisca na tela. Já no caso b, os intervalos de tempo em que a tela fica em branco e em seguida que aparece o nome TK-2000 COLOR na tela são bem maiores. O que acontece na realidade, é que quando incluímos na linha 20 a ordem FOR I = 1 TO 500 logo em seguida (sem que haja nenhuma outra instrução entre eles), o comando NEXT, fizemos com que o computador "conte" internamente, de 1 a 500 e só então prossiga o programa. O mesmo acontece na linha 50. Na realidade o tempo de contagem é muito rápido (menos que 1 segundo), porém suficiente para ser

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 ate que I = 500, após o que a próxima ordem NEXT é ignorada).

30 NEXT I (acrescenta 1 a variável I e desvia o programa para a ordem FOR anterior - linha 20)

40 PRINT "TK-2000 COLOR" (apresenta o nome "TK-2000 COLOR")

50 FOR I = 1 TO 500 (similar a ordem da linha 20)

60 NEXT I (similar a ordem da linha 30, porém note que agora a ordem FOR anterior se encontra na linha 50)

70 GO TO 10 (desvia o programa para a linha 10)

Fig. VI.4.

Se agora quisermos que o nome TK-2000 COLOR pisque mais lentamente, basta aumentar as contagens efetuadas pelo programa, substituindo, por exemplo, as linhas 20 e 50 por:

```
>20 FOR I = 1 TO 1000
```

```
>50 FOR I = 1 TO 1000
```

De forma inversa, para que TK-2000 COLOR pisque mais rapidamente, devemos diminuir as contagens realizadas pelo programa, alterando as linhas 20 e 50. Exemplo:

```
>20 FOR I = 1 TO 100
```

```
>50 FOR I = 1 TO 100
```

Para terminar, esta seção, execute o programa abaixo e observe o resultado:

```
>NEW
```

```
>10 REM EDUCANDO O TK-2000 COLOR
```

```
>20 HOME
```

```
>30 PRINT "MEU NOME E TK-2000 COLOR"
```

```
>40 FOR B = 1 TO 800 : NEXT B
```

```
>50 REM ODETENDO 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
```

VI.1.3. STEP

Até agora, sempre que utilizamos FOR....NEXT, a cada vez que o programa passava em NEXT, o valor da variável indicada na ordem FOR era acrescida de uma unidade, até que seu valor se igualasse do valor final. Podemos então dizer que o passo padrão da instrução FOR....NEXT é 1.

O passo da instrução FOR....NEXT pode ser alterado acrescentando-se junto a FOR a ordem STEP. Observe o exemplo:

```
>NEW
>10 HOME
>20 FOR I = 0 TO 10 STEP 2
>30 PRINT I
>40 NEXT I
>50 END
```

Após a ordem RUN, os números 0, 2, 4, 6, 8 e 10 serão acrescentados na tela. Como podemos concluir, a cada instrução NEXT a variável I passou a ser incrementada com o valor indicado após a ordem STEP.

VI.2. TECNICAS DE EDIÇÃO

Em capítulos anteriores já apresentamos a forma através da qual você pode corrigir erros em uma linha de programa, antes de ter digitado a tecla RETURN. Vamos revê-las rapidamente.

- a) a tecla (<-) movimenta o cursor para esquerda de forma que os caracteres que ficarem a direita dele serão ignorados, ou seja, embora permaneçam na tela não serão registrados na memória do TK-2000 COLOR;
- b) a tecla (->) movimenta o cursor para a direita, tornando os caracteres de uma linha válidos, a medida que o cursor passe por eles;
- c) o conjunto de teclas [CONTROL][X] faz com que a linha em que se encontra o cursor seja ignorada;
- d) o comando HOME apaga toda a tela e leva o cursor para a primeira posição da primeira linha.

Veremos a seguir as novas técnicas de edição especialmente úteis no MODO PROGRAMADO.

V.2.1. Supressão de Linhas de Programa (Comando DEL)

Para suprimir uma linha inteira do programa, basta digitar o número desta linha e logo em seguida pressionar a

tecla RETURN. Por exemplo, suponha que o programa abaixo foi editado:

```
>NEW
>10 REM PROGRAMA EXEMPLO
>20 HOME
>30 INPUT "QUAL O SEU NOME"; N$
>40 PRINT N$
>50 HOME
>60 PRINT "ESTE E O COMPUTADOR TK-2000 COLOR"
>70 END
```

Desejamos então suprimir a linha 50:

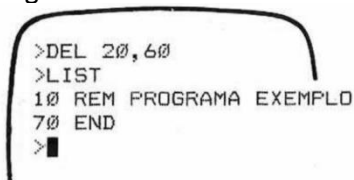
```
>50 <RETURN>
```

Agora emitindo-se o comando LIST teremos:

```
10 REM PROGRAMA EXEMPLO
20 HOME
30 INPUT "QUAL E O SEU NOME"; N$
40 PRINT N$
60 PRINT "ESTE E O COMPUTADOR TK-2000 COLOR"
70 END
```

Como podemos ver, tanto o conteúdo da linha 50 como o próprio número da linha foram suprimidos.

Você deve usar o comando DEL para suprimir um conjunto de linhas. A figura abaixo indica um exemplo deste procedimento:



```
>DEL 20,60
>LIST
10 REM PROGRAMA EXEMPLO
70 END
>■
```

Fig. VI.5

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

VI.2.2. Permutando Caracteres

A permuta de caracteres de uma linha é bastante simples. Basta movimentar o cursor, através das teclas adequadas (↑ → ↓ ←), para a parte que se deseja alterar e digitar os novos caracteres sobre os antigos. Por exemplo, se na linha:

```
100 PRINT " QUANTIDADE REAL"
```

desejarmos trocar REAL por ESTIMADA, basta posicionar o cursor sobre a letra R e digitarmos ESTIMADA". A linha passará então a forma:

```
100 PRINT "QUANTIDADE ESTIMADA"
```

Para se efetivar a alteração devemos pressionar a tecla RETURN.

V1I.2.3. Supressão de Caracteres

Pode-se suprimir caracteres, individualmente, digitando-se um espaço em branco sobre eles. Note que no BASIC os espaços brancos a mais são ignorados, a não ser que sejam inseridos entre aspas.

FAÇA O SEU RESUMO

1. A instrução FOR....NEXT tem por finalidade básica um determinado número de vezes as instruções que se encontram entre as ordens e
2. O formato da ordem FOR é
<FOR><.....-.....><.....><.....-.....>
<.....><.....-.....>
e o da ordem NEXT é
<NEXT><.....-.....>
3. Várias instruções separadas (.....) podem ser reunidas numa única linha, sendo que a ordem de execução destas será da para a
4. Pode-se usar também a instrução FOR.....NEXT como um dentro de um programa. Para tanto a ordem deve estar logo em seguida a , sem que haja nenhuma instrução entre elas.
5. Quando se usa a instrução FOR.....NEXT como , pode-se variar o intervalo determinado alterando-se a contagem desta instrução, ou seja, quanto maior a diferença entre os valores e da variável definidas na ordem FOR, maior o tempo e quanto menor esta diferença menor o tempo.
6. Para que a linha em que se encontra o cursor seja ignorada, deve-se manter a tecla pressionada e digitar então a tecla
7. O comando permite suprimir um conjunto de linhas do programa, seu formato é:
<.....><A><.....>
onde A e B correspondem respectivamente aos números da e linhas a serem suprimidas.
8. Para trocar um caracter de uma linha de programa, basta posicionar o no local ocupado por este e o novo caracter.
9. A supressão de um caracter é feita digitando-se um sobre este.
10. Normalmente o passo da instrução FOR...NEXT é Este passo pode ser alterado através da ordem junto a FOR.

RESPOSTAS:

1. repetir, FOR, NEXT; 2. nome-de-variável-real, =, valor inicial da variável, TO, valor final da variável real, nome da variável real; 3. dois pontos (:), esquerda, direita; 4. tempo de espera, NEXT, FOR; 5. tempo de espera, final, inicial; 6. CONTROL, x; 7. DEL, DEL, , , primeira, última; 8. cursor, digitar; 9. espaço em branco; 10. 1, STEP.

CAPÍTULO 7

CAPITULO VII- GRUPOS DE INSTRUÇÕES E COMANDOS

Temos usado no decorrer deste livro, os nomes, instruções e comandos, aparentemente sem distinção. Na realidade, existe uma diferença sutil entre estes dois termos. Enquanto os comandos dizem diretamente ao computador o que fazer, as instruções aparecem como parte de um programa. For exemplo, RUN e LIST são comandos, pois indicam diretamente para o computador iniciar a execução do programa e apresentar as linhas de programação respectivamente; não faz sentido utilizar estes comandos dentro de um programa (embora possam ser usados). INPUT é um típico caso de instrução, pois deve ser usada necessariamente dentro de um programa. Estes dois termos passam a se confundir, por exemplo, em HOME, que pode tanto ser usado individualmente como um comando ou dentro de Um programa, como uma instrução. A diferenciação entre comandos e instruções é apenas conceitual, sendo que na prática não tem maior relevância.

Neste capítulo, as novas instruções e comandos são apresentados dentro de grupos, de acordo com a finalidade a que se destinam. Após sua leitura, você passará a estar a par da maioria das instruções básicas do BASIC, e já terá recursos razoáveis para começar a desenvolver seus próprios programas.

VII.1. OPERAÇÃO COM CADEIAS

Esta seção tem por finalidade permitir que você manipule cadeias, dividindo-as em partes, contando o número de caracteres que elas possuem etc...

VII.1.1. LEN

Esta instrução faz com que o computador conte o número de caracteres de uma cadeia. Ela pode ser usada em dois tipos de formato, conforme se vê a seguir:

```
<LEN> <( <cadeia> <)>
ou
<LEN> <( <nome-de-variável tipo cadeia> <)>
```

Por exemplo, digite as linhas:

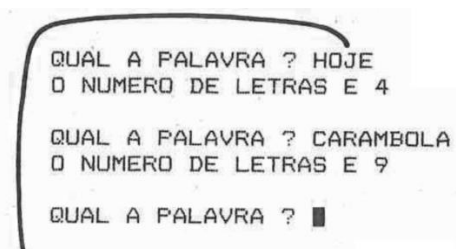
```
>PRINT LEN ("ANTENA")
ou
>A$ = "ANTENA"
>PRINT LEN (A$)
```

Como se pode ver, nas duas formas o resultado foi o mesmo, Ou seja, a apresentação do número é na tela, uma vez que ANTENA possui 6 caracteres.

Entre agora com o seguinte programa no seu 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 ao nome-de-variável real N. A figura VII.1 apresenta um possível resultado da execução deste programa.



```
QUAL A PALAVRA ? HOJE
O NUMERO DE LETRAS E 4

QUAL A PALAVRA ? CARAMBOLA
O NUMERO DE LETRAS E 9

QUAL A PALAVRA ? █
```

Fig. VII.1.

Para interromper o programa pressione CONTROL C ou RESET.

VII.1.2. LEFT\$

Esta instrução tem por função selecionar caracteres de uma cadeia, a partir da esquerda. Execute o seguinte exemplo

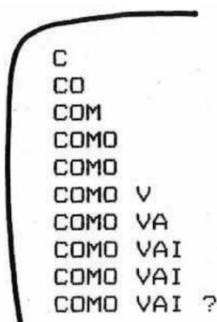
```
>A$ = "COMO VAI VOCE"
>PRINT LEFT$ (A$,4)
```

Como resultado teremos os quatro primeiros caracteres da cadeia apresentados na tela, ou seja, a palavra COMO.

Vamos rodar agora 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
```

Surge então uma tela, conforme indica a figura seguinte



```
C
CO
COM
COMO
COMO
COMO V
COMO VA
COMO VAI
COMO VAI
COMO VAI ?
```

Fig. VII.2.

O fluxograma do programa é (fig.VIII.3)

| | |
|-----------------------|---|
| HOME | (limpa a tela e posiciona o cursor em seu início) |
| A\$ = "COMO VAI ?" | (associa ao nome-de-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 igual ao número de caracteres da cadeia. Uma vez que neste particular caso a cadeia tem 10 caracteres, esta instrução corresponderia a FOR I = 1 TO 10). |
| PRINT LEFT\$ (A\$,I) | (apresenta os I primeiros caracteres da y cadeia) |
| NEXT I | (soma 1 a variável I e desvia o programa para a ordem FOR até que I atinja seu valor final - no caso 10). |
| END | (finaliza o programa) |

Fig. VII.3

E importante você notar que, para o computador, o espaço em branco também é processado como qualquer outro caracter, portanto quando a instrução LEN (A\$) foi utilizada, ela contou 1% (incluindo os dois espaços em branco).

VII.1.3. RIGHT\$

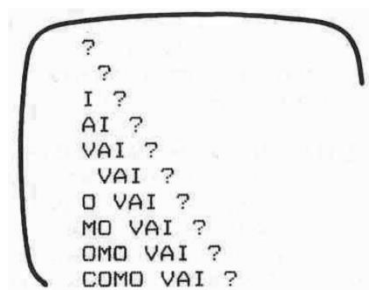
A instrução RIGHT\$ opera de forma semelhante a anterior, porém neste caso a seleção de caracteres é feita em ordem inversa, ou seja, da direita para a esquerda. Por exemplo:

```
>A$ = "COMO VAI VOCE"
>PRINT RIGHT$ (A$,4)
```

Neste caso teremos então a palavra VOCE apresentada no vídeo. Execute agora o programa:

```
>NEW
>5 HOME
>10 A$ = "COMO VAI ?"
>20 FOR I = 1 TO LEN (A$)
>30 PRINT RIGHT$ (A$,I)
>40 NEXT I
>50 END
>RUN
```

A tela apresentará então (fig.VII.4)



```
?
?
I ?
AI ?
VAI ?
VAI ?
O VAI ?
MO VAI ?
OMO VAI ?
COMO VAI ?
```

Fig. VII.4.

O fluxograma deste programa é idêntico ao apresentado na seção anterior, exceto que na instrução PRINT RIGHT\$ (A\$,I) passam a ser apresentados os I últimos caracteres da cadeia.

VII.1.4. MID\$

MID\$ permite selecionar caracteres intermediários de uma cadeia. Por exemplo, a instrução MID\$ (A\$,3,5) faz com que sejam selecionados, a partir do terceiro caracter da cadeia A\$, cinco caracteres. Digite agora.

```
>PRINT MID$ ("COMO",2,2)
```

Serão apresentados então os caracteres OM, ou seja, os dois caracteres a partir do 2 da cadeia. Entre com o programa a seguir:

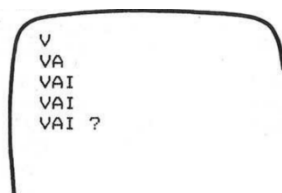
```
>NEW
>5 HOME
>10 A$ = "COMO VAI ?"
```

```

>20 FOR I = 1 TO LEN (A$)- 5
>30 PRINT MID$ (A$,6,I)
>40 NEXT I
>50 END
>RUN

```

Teremos então na tela (fig. VII.5)



```

V
VA
VAI
VAI
VAI ?

```

Fig. VII.5

Note que na instrução `FOR I = 1 TO LEN(A$)- 5`, foi subtraído 5, uma vez que sô existem 5 caracteres a serem apresentados. Desta forma, neste caso está instrução fica equivalente a:

```

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

```

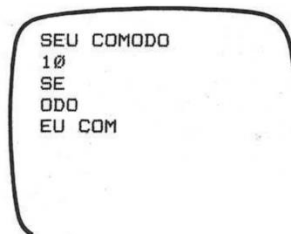
Para confirmar sua boa compreensão a respeito das instruções tratadas neste item, digite o programa abaixo e, sem olhar a figura que segue a listagem do programa procure rever os resultados na tela.

```

>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)
>70 END

```

Já fez sua previsão? Então digite o comando `RUN` e confira seus resultados que devem estar de acordo com a figura abaixo:



```

SEU COMODO
10
SE
ODO
EU COM

```

Fig. VII.6.

VII.2. OPERAÇÃO COM DADOS NUMERICOS

As operações com dados numéricos serão apresentadas de forma mais ampla em capítulos posteriores. For hora, daremos apenas uma introdução neste assunto através dos comandos RND e INT.

VII.2.1. RND

O nome desta instrução corresponde a abreviatura de randômico. Uma série de números randômicos significa números sem qualquer relação entre si, aleatórios, ou que não existe função que possa representá-los. Se você não tem formação matemática, pode lhe ser difícil entender este conceito, mas o que importa é observar sua ação e saber que, em determinadas áreas (por exemplo, jogos) é uma instrução especialmente útil.

No TK-2000 COLOR, RND(1) faz com que sejam selecionados números randômicos entre 0 e 1 ($0 < n < 1$). Digite o exemplo:

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

A primeira vez que você rodar este programa, o TK-2000 COLOR apresentará os resultados indicados a seguir. Note entretanto que, cada vez que o programa for novamente rodado, os resultados serão diferentes, porém sempre com números entre 0 e 1.

```
>RUN

.270011996          .139756248
.690102028          .141352116
.152267027          .690658204
.360889732          .628262312
.581121379          .768268873
```

Se você desejar sequências 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
```

A primeira vez que este programa for executado, os resultados serão:

RUN

| | |
|------------|------------|
| 2.70011996 | 1.39756248 |
| 6.90102029 | 1.41352116 |
| 1.52267027 | 6.90658204 |
| 3.60889731 | 6.28262312 |
| 5.81121379 | 7.68268873 |

Rode este-programa mais algumas vezes e observe que os novos resultados dificilmente serão repetidos.

VII.2.2. INT

A instrução INT faz com que a parte fracionária de um número real seja desprezada. Por exemplo:

```
>PRINT INT(1.896)
>PRINT INT(2.571)
>PRINT INT(0.62328)
```

Os comandos acima terão resultados 1 2 e 0 respectivamente.

E bastante frequente a necessidade de se gerar números randômicos inteiros e para tanto utilizamos as instruções INT e RND em conjunto. Suponha que se deseja fazer um programa para a seleção de cinco números (entre 0 e 99) de um cartão da Loto.

```
>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
```

Observações: 1. Após digitar o programa, o primeiro conjunto de resultados (quando você rodar pela primeira vez o programa), será sempre igual, bem como o segundo, terceiro, quarto, etc... porém o primeiro conjunto será diferente do segundo por sua vez diferente do terceiro, do quarto, etc. Assim, se após rodar várias vezes o programa você apagá-lo desligando o TK-2000 COLOR, e em seguida recarregá-los, as sequências de conjuntos de resultados serão iguais a anterior.

2. Se você não entendeu por que RND(1). Foi multiplicado por 100 basta verificar exemplos extremos.

a) $RND(1) = 0.00125 \rightarrow (100 * 0.00125) \rightarrow 0.12625 \rightarrow INT(0.12625) \rightarrow 0$

b) $RND(1) = 0.99978 \rightarrow (100 * 0.99978) \rightarrow 99.978 \rightarrow INT(99.978) \rightarrow 99$

Para gerar 10 números inteiros entre 1 e 10 podemos usar o seguinte programa.

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

Procure agora desenvolver seus próprios programas utilizando as instruções RND e INT. For exemplo, gere conjuntos de números dentre de diversos limites.

VII.3. COMANDOS RELACIONADOS A EXECUÇÃO DE PROGRAMAS

Em capítulos anteriores já apresentamos alguns comandos relacionados a este grupo, que foram: LOAD, NEW, CONTROL C, RUN e END. Neste capítulo vamos recordar as instruções LOAD e CONTROL C, e analisar as novas instruções deste grupo.

VII.3.1. LOAD e SAVE

Estas duas instruções são usadas somente quando o TK-2000 COLOR está equipado com gravador cassete. Na realidade elas são divididas em:

- a) LOADT "nome-do-programa": carrega um programa cujo nome (que é opcional, neste caso) está entre aspas do cassete para o TK-2000 COLOR.
- b) LOADA: carrega programa de fita cassete no formato Apple II, no TK-2000 COLOR.
- c) SAVET"nome-do-programa": grava o programa cujo nome opcional) está entre aspas do TK-2000 COLOR para a fita.
- d) SAVEA: grava programas e dados em fitas no formato Apple II.

Se você já está com o seu gravador conectado ao TK-2000 COLOR, devidamente regulado (conforme descrito na seção II.5) e possui uma fita virgem (ou qualquer outra cujo conteúdo não lhe seja importante), execute a seguinte operação.

1. Se a fita não for virgem, limpe-a, gravando-a em branco (com a entrada de gravação desconectada- não esquecendo de reconectar a entrada no final desta operação).
2. Rebobine a fita e ligue o gravador em RECORD (gravar)

3. Escreva algum programa no TK-2000 COLOR e verifique se ele está em ordem. Por exemplo:

```
>NEW  
>10 HOME  
>20 FOR I = 1 TO 5  
>30 PRINT "TESTANDO"  
>40 NEXT I  
>RUN
```

Note que este programa deve apresentar a palavra "testando" no início das cinco primeiras linhas do vídeo.

4. Pense num nome com até 6 caracteres para este programa, como por exemplo "TANDO"
5. Digite então

```
>SAVET "TANDO"
```

e aguardando alguns instantes, até o gravador cessar suas atividades.

6. Digite:

```
>NEW
```

Agora se você tentar rodar novamente o programa (através de RUN), não haverá resultado algum, uma vez que o programa foi apagado da memória do TK-2000 COLOR.

7. Rebobine a fita e acione o gravador em PLAY

8. Digite:

```
>LOADT "TANDO"
```

e aguarde a atividade do gravador cessar.

9. Agora, digitando o comando RUN, verifique como o programa roda, uma vez que foi recarregado na memória do gravador.

Observação: o programa TANDO está gravado permanentemente na fita cassete, ou seja, mesmo após desligarmos o sistema (computador e gravador), sempre que desejarmos, poderemos carregá-lo no TK-2000 COLOR através do comando LOADT "TANDO", a não ser que a área da fita cassete onde ele se encontra gravado seja alterada.

VII.3.2. STOP e CONT

Tanto a instrução STOP como END interrompem a execução de um programa. A diferença entre estas duas é que várias instruções STOP podem ser usadas durante um programa enquanto a instrução END deve ser usada apenas para finalizar o programa. Quando o computador executa um STOP, ele apresenta a mensagem:

```
BREAK EM xx
```

onde xx representa o número que indica a 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
```

será apresentando então:

```
12345678910  
BREAK EM 40  
>■
```

Quando o programa é interrompido com STOP, pode-se dar prosseguimento a 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
```

Ao se digitar RUN teremos então:

```
>RUN  
12  
BREAK EM 40  
>■
```

Digitando-se então CONT, o programa passa a executar as linhas 50 a 70.

```
>CONT  
12345
```

Note nos exemplos abaixo, que a instrução END pode ser usada de forma semelhante a STOP, porém neste caso nenhuma mensagem é apresentada.

Exemplo A

```
>NEW
>10 FOR I = 1 TO 10
>20 PRINT I;
>30 NEXT I
>40 END
>RUN
```

Resultado

```
123345678910
>■
```

Exemplo B

```
>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
>■
```

Digitando-se então CONT teremos:

```
>CONT
12345
>■
```

Como se pode notar, o comando CONT também pode ser usado quando o programa é interrompido pela instrução END.

VII.3.3. CONTROL C e RESET

A função que se obtém mantendo-se a tecla CONTROL pressionada e digitando-se em seguida a tecla C é idêntica a da instrução STOP, ou seja, a interrupção do programa seguida pela apresentação da linha na qual este foi interrompido. Entretanto, CONTROL C é acionado pelo operador em qualquer ponto do programa. Exemplo:

```
>10 PRINT "BOM DIA"
>20 GOTO 10
>RUN
```

Após termos entrado com o programa acima, no momento em que quisermos interrompê-lo bastará manter CONTROL abaixada e

digitar C, quando então teremos uma tela do tipo da figura VII.7

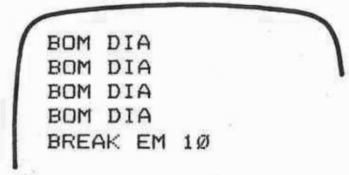


Fig. VII.7

Após termos interrompido o programa através do comando mencionado, poderemos dar novamente prosseguimento a ele acionando o comando CONT.

O pressionamento da tecla RESET faz com que o TK-2000 COLOR execute função semelhante a CONTROL C, porém sem apresentar a linha na qual o programa foi interrompido, e não aceita o comando CONT.

VII.3.4. TRACE e NOTRACE

Se você quiser seguir o processo de execução de um programa, poderá usar TRACE. Antes de começar o programa, pressione TRACE e RETURN. Consequentemente, quando o programa estiver rodando, o computador apresentará o número de cada linha conforme ela esteja sendo executada. 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
```

A figura a seguir indica o significado do resultado deste programa.

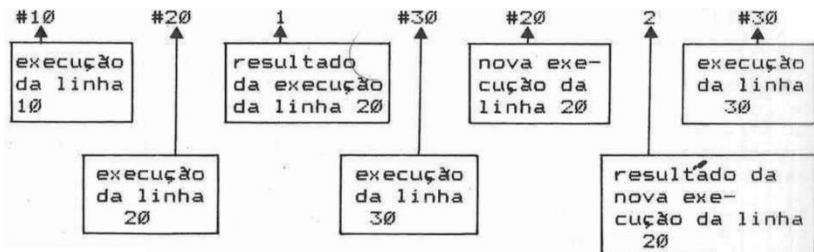


Fig. VII.8.

A principal função do TRACE é auxiliar o programador a

descobrir erros durante a elaboração de um programa.

O único comando que desativa o modo TRACE e NOTRACE. Portanto note que uma vez que TRACE esteja acionado, NEW ou RESET não cessarão sua atividade.

VII.3.5. POKE e PEEK

A utilização destas duas funções, requer uma certa experiência em programação. Vamos fornecer aqui suas funções, porém se você não tem uma certa prática e algum conhecimento do hardware do microcomputador não é necessário que se detenha muito nesta seção.

POKE serve para introduzir dados diretamente na memória do computador, lembre-se que os dados são introduzidos nas memórias RAM, uma vez que o conteúdo das memórias ROM é inalterável. Para se executar esta introdução deve-se indicar duas coisas:

1. O dado a ser introduzido
2. O endereço da memória RAM para o qual se deseja enviar o dado desejado.

Exemplo:

```
>NEW
>POKE 1500,56
      ↑  ↑
      Endereço dado
      (decimal) (decimal)
```

POKE 1500,56 indica a introdução do valor 56 no endereço 1500. Aqui, para facilitar a nossa apresentação, 56 e 1500 são números decimais os quais serão transformados em binários no computador para serem armazenados.

Os dados no microprocessador são armazenados em bytes (8 bits), assim sendo o dado a ser introduzido diretamente na memória deve estar compreendido entre 0 e 255. Se for tentada a introdução de um número fora destes limites, o computador apresentará a mensagem:

```
?VALOR ILEGAL #ERRO
```

Esta mesma mensagem será também apresentada se o endereço for maior que 65.535, uma vez que este é o maior endereço da memória RAM do TK-2000 COLOR.

Desejando-se ter acesso direto a um dado de um determinado endereço da memória RAM do TK-2000 COLOR, 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 desta seção (POKE 1500,56), ao digitar:

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

Surgirá na tela o número 56, que foi justamente o conteúdo introduzido no endereço 1500 através da instrução POKE.

VII.4. INSTRUÇÕES RELATIVAS A EDIÇÃO E FORMATAÇÃO

Este grupo de instruções se refere a apresentação de determinados dados ou instruções de programas, bem como a forma em que estes são apresentados. Algumas das instruções deste grupo já foram descritas:

LIST, DEL, REM, HOME e NEW

VII.4.1 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. TAB é normalmente associado a PRINT.

No exemplo abaixo os números 5 e 7 serão apresentados na 10 e 15 coluna respectivamente.

```
>PRINT TAB(10); 5; TAB(15); 7
```

VII.4.2. VTAB e HTAB

Através de VTAB pode-se mover o cursor para qualquer linha do vídeo. Este movimento é sempre 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-2000 COLOR emita a mensagem de erro:

```
?VALOR ILEGAL #ERRO
```

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

Confira no exemplo a seguir as funções desta instruções:

```
>NEW  
>10 HOME  
>20 VTAB 12  
>30 HTAB 18  
>50 PRINT "TK-2000 COLOR"  
>60 END
```

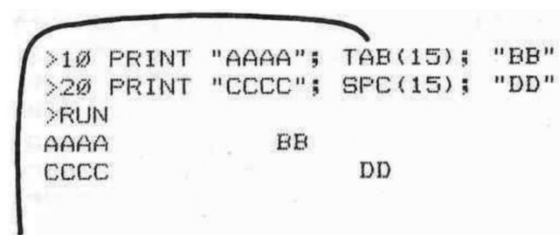
Como você deve ter concluído, o nome TK-2000 COLOR foi apresentado a partir da linha 12, coluna 18 do vídeo.

VII.4.3. SPC

SPC é a abreviação de espaço. SPC(20) indica que o cursor deve ser movido 20 espaços horizontalmente. Vamos observar no próximo exemplo a diferença entre os efeitos das instruções SPC e TAB.

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

Ao executarmos este programa aparecerá na tela algo semelhante a próxima figuras.



```
>10 PRINT "AAAA"; TAB(15); "BB"  
>20 PRINT "CCCC"; SPC(15); "DD"  
>RUN  
AAAA          BB  
CCCC          DD
```

Fig. VIII.9

Conforme se pode observar, o primeiro B se encontra na 15 coluna, contada a partir do início da tela enquanto o primeiro D é apresentado 15 posições após o último C.

VII.4.4. POS

POS é a abreviação de posição. Esta instrução apresenta a última posição ocupada pelo cursor em uma linha. Normalmente, deve-se apresentar parênteses após POS, sendo que o número entre parênteses não faz diferença, por exemplo, POS (0) e POS(15) tem o mesmo significado.

Observe o exemplo abaixo para entender melhor a função deste comando.

```

>NEW
>10 PRINT "AAA"; POS(0)
>20 PRINT "BBBB"; POS(12)
>30 PRINT "AAA"; TAB(10); "BBBB"; POS(0)
>40 PRINT "CCC"; SPC(10); "DDDD"; POS(0)
>50 PRINT TAB(23); POS(0)
>60 PRINT SPC(23); POS(0)

```

Após digitarmos RUN, teremos:

```

>RUN
AAA3
BBBB4
AAA      BBBB13
CCC      DDDD17
                22
                23

```

Observações:

1. Nas linhas 10 e 20 a instrução POS faz com que seja indicado exatamente o número de caracteres da linha, uma vez que a apresentação destes começa no início da linha, e que não há espaços entre eles.
2. Se você não entendeu o resultado da execução das linhas 30 e 40, recapitule as seções VII.4.1. e VII.4.3. (instruções TAB e SPC respectivamente)
3. Através dos resultados da execução das linhas 50 e 60, pode-se concluir que não é necessário que seja apresentado qualquer caracter na tela para que a última posição ocupada pelo cursor, em determinada linha seja diferente de zero.

VII.4.5. CLEAR

O comando CLEAR permite que se "limpem" da memória todas as variáveis existentes. Por exemplo, execute as seguintes linhas:

```

>A = 7
>PRINT A

```

Como resultado teremos o número 7 apresentado na tela. Agora faça:

```

>CLEAR
>PRINT A

```

Conforme se pode observar, ao invés de 7 passa a ser apresentado 0 (0 é apresentado sempre que no comando PRINT não

há nenhum valor atribuído ao nome-de-variável utilizado).

VII.4.6. FRE(0)

Este comando informa a área ou a quantidade de memória RAM já ocupada pelo BASIC. Ele deve ser usado juntamente com PRINT sendo que o resultado sempre será negativo quando a área ocupada for menor que 32767. Seu formato é:

```
>PRINT FRE(0)
```

Para se ter o tamanho real da memória disponível bastará somar 65536 ao número gerado pela instrução acima.

VII.4.7. INVERSE E NORMAL

Ao usar o comando INVERSE os caracteres aparecerão no vídeo em preto sobre fundo branco. O comando NORMAL volta a situação original, de caracteres em branco sobre fundo preto.

VII.4.8. SPEED

Podemos determinar a velocidade com que os caracteres do apresentados na tela. Quando SPEED = 0, é a velocidade mais baixas e quando SPEED = 255, a velocidade mais alta. Uma tentativa de usar um número maior que 255 resulta na mensagem de erro:

```
?VALOR ILEGAL #ERRO
```

Esta diferença de velocidade pode ser facilmente notada através do exemplo a seguir:

```
>NEW
>10 SPEED = 1
>20 FOR I = 1 TO 100
>30 PRINT I; " ";
>40 NEXT I
>50 SPEED = 255
>60 FOR I = 101 TO 200
>70 PRINT I; " ";
>80 NEXT I
>RUN
```

Observe a diferença de tempo entre a contagem de 1 a 100 e de 101 a 200.

VII.4.9. CONTROL X

Mantendo-se a tecla CONTROL abaixada e pressionando-se em seguida a tecla X, uma sinal de barra (N) será colocado na linha em que se encontra o cursor, fazendo com que esta linha seja ignorada pelo computador. Por exemplo:

```
>NEW  
>10 PRINT "BARRA"
```

Agora pressionando a tecla CONTROL em conjunto com X, a linha acima ficará de forma:

```
>10 PRINT "BARRA" \
```

Pode-se então digitar RUN e observar que esta linha foi anulada não tem qualquer efeito.

FAÇA O SEU RESUMO

1. Para que o computador apresente o número de caracteres de uma cadeia devemos digitar:

```
<PRINT><.....><.><"cadeia"><...>
```

ou então associarmos a cadeia a um nome-de-variável e efetuar o comando:

```
<PRINT><.....><.....><nome-de-variável tipo cadeia><.....>
```

2. O comando:

```
<PRINT><.....><(><.....><.....><.....><)>
```

faz com que os I primeiros caracteres da cadeia associada a variável A\$ sejam apresentados na tela.

3. Desejando-se que sejam apresentados na tela os I últimos caracteres da cadeia associada a variável A\$ utiliza-se:

```
<PRINT><.....><(><.....><.....><.....><)>
```

4. Se o nome-de-variável A\$ for associada a palavra BORBOLETA (A\$ = "BORBOLETA"), para que o TK-2000 COLOR apresente a cadeia "OLE", através de um único comando que utilize A\$ deve-se digitar:

```
PRINT ..... (A$, ..... , .....)
```

5. Para se obter uma série de números randômicos entre 0 e 1 pode-se usar a instrução:

```
<.....> <.....> <1> <.....>
```

6. O resultado apresentado pelo comando:

```
<PRINT> <INT> <(> <número real> <)>
```

será a parte do número real

7. Os comandos usados para carga de programas do tipo TK-2000 COLOR e Apple-II são respectivamente:

..... "nome de arquivo"

e

.....

8. À instrução faz com que a execução de um programa seja interrompida e o número da linha em que houve esta interrupção apresentada no vídeo. Para que o programa tenha prosseguimento deve-se usar o comando

9. O operador pode interromper a execução de um programa, fazendo com que seja apresentada a linha em que este foi interrompido, mantendo pressionada a tecla e digitando a tecla Se for desejado simplesmente que o programa seja interrompido, sem que a linha em que foi feita a interrupção seja apresentada, basta digitar a tecla

10. O comando permite que o processo de execução de um programa seja seguido pelo operador. Para se desativar este modo de operação deve-se usar o comando

11. Para se introduzir um dado numa posição da memória RAM do

computador usa-se o comando:

<.....> <endereço da memória> <.....> <dado>

De forma inversa, para se acessar um dado de uma determinada posição da memória RAM utiliza-se:

<.....> <.....> <endereço da memória> <.....>

12. A instrução

<.....> <.....> <n> <.....>

faz com que o cursor se desloque para a n-ésima coluna da tela.

13. Para mover o cursor para a linha n da tela podemos utilizar a instrução De forma semelhante, para que o cursor seja levado para a posição y a partir do início de uma linha utiliza-se a instrução

14. O cursor se deslocará n espaços, a partir da última posição em que estiver, através da instrução:

<.....> <(<n> <)>

15. A apresentação da última posição ocupada pelo cursor em uma linha é obtida através da instrução:

<.....> <(<n> <)>

onde n pode ser qualquer número.

16. Para se "limpar" as variáveis da memória usa-se o comando

<.....>

17. Quando se quiser saber a quantidade de memória livre do TK-2000 COLOR deve-se digitar:

<.....> <.....> <(<.....> <)>

18. A velocidade de apresentação dos caracteres na tela pode ser controlada através da digitação de:

<.....> <.....> <n>

onde n varia de 0 (menor velocidade) a 255 (maior velocidade)

19. Para que a linha em que se encontra o cursor seja ignorada, deve-se manter a tecla pressionada e digitar a tecla

respostas.

1. LEN, (, LEN,); 2. LEFT\$, A\$, , , I; 3. RIGHT\$, A\$, , , I; 4. MID\$, 5, 3; 5. RND, (,); 6. Inteira; 7. LOADT, LOADA; 8. STOP, CONT; 9. CONTROL, C, RESET; 10. TRACE, NOTRACE; 11. POKE, , , PEEK, (,); 12. TAB, (,); 13. VTAB n, HTAB y; 14. SPC; 15. POS; 16. CLEAR; 17. PRINT, FRE, 0; 18. SPEED, =; 19. CONTROL, X

CAPÍTULO **8**

CAPITULO VIII- OPERAÇÕES NO TK-2000 COLOR

Em programas de computador, podemos usar variáveis para representar um endereço de memória. A função de cada variável é semelhante a da memória do computador.

As variáveis podem ser divididas em várias categorias:

1. Variáveis numéricas como A, B, C%
2. Variáveis tipo cadeia como A\$, B\$
3. Variáveis tipo matriz (que serão vistas adiante), tais como A(10), A(3,5), A(4,5,8), A\$(10,12)

As operações de um computador podem ser subdivididas em três categorias:

1. Operações Aritméticas, tais como: +, -, *, /.
2. Operações Comparativas, tais como: >, <, >=, <=.
3. Operações Lógicas, tais como: NOT, AND e OR.

VIII.1. OPERAÇÕES COM VARIÁVEIS NUMÉRICAS

No TK-2000 COLOR às variáveis numéricas são de duas categorias:

Inteiras- devemos acrescentar o sinal "%" após o nome destas variáveis.

Números Reais- o TK-2000 COLOR assume um número real se o nome da variável não é especificamente assinalada.

VIII.1.1. Inteiros

Cada valor inteiro (seguido por %) ocupa dois bytes, isto é 16 bits. Os inteiros são limitados entre -32.767 e 32.767. Se quisermos trocar um número real por um inteiro, podemos especificar uma variável inteira do lado esquerdo de um sinal de igualdade. Execute o programa a seguir para que esta operação se torne mais clara.

```
>NEW
>10 FOR 1 = 1 TO 6
>20 A = RND(1) * 6 + 1
>30 A% = A
>40 PRINT A, A%
>50 NEXT I
```

Após digitar RUN teremos um resultado do tipo:

| | |
|-------------|---|
| >RUN | |
| 6.83882197 | 6 |
| 1.61870576 | 1 |
| 1.106289 | 1 |
| 5.67606013 | 5 |
| 4.31100663 | 4 |
| 4.470451467 | 4 |

Observação: Quando utilizar inteiros, não use qual quer sinal para designar milhares, milhões, etc..

| | | |
|------------|--------|-------|
| Ex. 32.000 | 32,000 | 32000 |
| X | X | √ |

VIII.1.2. Reais

No TK-2000 COLOR, os limites dos números reais são: 1.70141183 E+38, com o máximo de 9 algarismos.

Para apresentar um número real, se você desejar determinar seu formato, deverá observar as seguintes regras.

- a) Suponha que o número considerado seja negativo: o sinal negativo será sempre apresentado, antes do número. Exemplo:

```
>PRINT 132 * -12
-1584
```

- b) Se o valor absoluto estiver entre 0 e 999999999, ele será apresentado sob a forma de um número inteiro. Exemplo:

```
>PRINT 111111111 * 9
999999999
```

- c) Quando o valor absoluto do número é igual ou maior que 0.01 é menor que 999999999.2, ele será apresentado com ponto decimal fixo e sem expoente. Exemplo:

```
>PRINT 0.01
.01
>PRINT 999999999.199
999999999
```

- d) Suponha que a limitação do número não esteja contida nos itens b e c, ele será apresentado na notação científica. Exemplo:

```
>PRINT 111111112 * 9
1E+10
>PRINT 999999999.21
1E+09
```

Se o número real for apresentado na notação científica, é necessário que depois do ponto decimal haja um número significativo (diferente de zero) e no máximo 8 casas depois do ponto decimal. A letra E representa um expoente e deve ser seguido por um número positivo ou negativo de no máximo dois algarismos. Havendo somente zeros a esquerda do ponto decimal, eles não serão apresentados. Se todos os algarismos depois do ponto decimal forem zeros, então tanto o ponto decimal como estes zeros serão omitidos. Exemplos:

```
>PRINT 0.2, 0.460
.2 .46
```

Na presença de um expoente, os sinais positivo e negativo serão apresentados. Um "+" é apresentado quando o expoente for positivo, e um "-" quando o expoente for negativo. Vejamos os seguintes exemplos:

| Notação Convencional | Notação Científica |
|----------------------|--------------------|
| 1000000000 | 1E+09 |
| .000000001 | 1E-09 |
| -123456789 | -1.23456789E+08 |
| -.00123456789 | -1.23456789E-09 |

VIII.2. TIPOS DE OPERAÇÕES NUMERICAS

Conforme vimos no início deste capítulo, as operações no TK-2000 COLOR podem ser divididas em três categorias, conforme descrevem os sub-itens a seguir.

VIII.2.1. Operações Numéricas

Os símbolos usados neste tipo de operação são: +, -, *, /, =

Este tipo de operação já foi bastante discutido na seção III.2

VIII.2.2. Operações Comparativas

À tabela a seguir indica os tipos de operações comparativas, bem como os respectivos símbolos usados pelo TK-2000 COLOR.

| | | |
|--------------------|-----------|---|
| ***** | | |
| * OPERAÇÃO | * SIMBOLO | * |
| *-----* | | |
| * menor que | * < | * |
| *-----* | | |
| * maior que | * > | * |
| *-----* | | |
| * igual | * = | * |
| *-----* | | |
| * diferente de | * <> | * |
| *-----* | | |
| * maior ou igual a | * >= | * |
| *-----* | | |
| * menor ou igual a | * <= | * |
| ***** | | |

Tabela VIII.1.

Estes sinais de operação são usados para comparar dois números. No TK-2000 COLOR, se a comparação for verdadeira, ele fornecerá o valor um (1); se não for, ele fornecerá o valor zero (0).

Observe o programa seguinte.

```

>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 |

Após digitar RUN teremos então:

```

>RUN
1
1
0
1
0
1
0

```

VIII.2.3. Operações Lógicas

Pode-se ver, na tabela VIII.2 as operações lógicas disponíveis, bem como seus respectivos símbolos.

| OPERAÇÃO | SÍMBOLO |
|----------|---------|
| E | AND |
| Ou | OR |
| Negação | NOT |

Tabela VIII.2.

Cada um destes operadores lógicos tem a sua tabela verdade, que indicam, a partir de certas entradas (1 ou 0), resultados determinado. Nestas tabelas, as entradas são apresentadas antes do sinal de igualdade, e as saídas após estes, conforme se vê nas Tabelas VIII.3, VIII.4 e VIII.5

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

Tabela VIII.3.

| TABELA VERDADE DA OPERAÇÃO E | |
|------------------------------|-------|
| Entrada | Saída |
| 1 OR 1 | 1 |
| 1 OR 0 | 1 |
| 0 OR 1 | 1 |
| 0 OR 0 | 0 |

Tabela VIII.4.

| TABELA VERDADE DA OPERAÇÃO E | |
|------------------------------|-------|
| Entrada | Saída |
| NOT 1 | 0 |
| NOT 0 | 1 |

Tabela VIII.5.

Seguem alguns exemplos do uso destas operações

```
1.) >PRINT (5 > 4) AND (3 > 2)
1
```

Para entender melhor o resultado da operação acima, podemos dividi-la em duas partes, que são as operações comparativas e a operação lógica.

| Operações Comparativas | Operação Lógica |
|------------------------|------------------------------|
| 5 > 4 → verdadeiro → 1 | |
| 3 > 2 → verdadeiro → 1 | 1 AND 1 → ver tabela AND → 1 |

```
2.) >PRINT (3 < 2) OR (1 < 0)
0
```

Vamos esquematizar da mesma forma que no exemplo anterior,

| Operações Comparativas | Operação Lógica |
|------------------------|----------------------------|
| 3 < 2 → falso → 0 | |
| 1 < 0 → falso → 0 | 0 OR 0 → ver tabela OR → 0 |

```
3.) >PRINT NOT (5 > 4)
0
```

Utilizando o mesmo esquema dos exemplos anteriores temos:

| Operação Comparativa | Operação Lógica |
|------------------------|----------------------------|
| 5 > 4 → verdadeiro → 1 | NOT 1 → ver tabela NOT → 0 |

Agora digite o programa a seguir e antes de pressionar RUN, procure prever seus resultados:

```
>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)
```

Apos pressionar RUN teremos:

```
>RUN
0
1
1
0
1
1
```

VIII.3. NOVAS OPERAÇÕES COM CADEIAS

Como já vimos, um símbolo \$ deve ser acrescentado a variável tipo cadeia. As operações tipo cadeia utilizadas no TK-2000 COLOR são mostradas a seguir:

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

Já apresentamos no capítulo V, algumas operações de cadeia:

| | | | |
|----------|---------------|----------------|----------------|
| LEN(A\$) | LEFT\$(A\$,X) | RIGHT\$(A\$,X) | MID\$(A\$,X,Y) |
|----------|---------------|----------------|----------------|

Há cinco operações que não foram estudadas ainda, e que serão apresentadas neste capítulo.

STR\$(X), +, VAL(A\$), CHR\$(X) e ASC(A\$)

VIII.3.1. STR\$(X) e "+"

STR representa "cadeia". STR\$ significa que o valor do número X entre parênteses será convertido numa cadeia. Por exemplo:

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

Após o comando RUN teremos:

```
>RUN
123456
```

Note que no exemplo anterior, ao invés dos dados 123 e 456 serem tratados como números, cujo resultado da soma seria 579, eles foram tratados como cadeias, e portanto o sinal "+" fez com que fossem apresentados em ordem sequencial (123456).

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

Exemplo:

| | |
|---------|----------|
| 2879742 | 287-9742 |
| 3498126 | 549-8126 |
| 8815979 | 881-5979 |

```

>NEW
>10 HOME
>20 INPUT "NUMERO DO TEL. ";A
>30 B$ = STR$(A)
>40 C$ = LEFT$(B$,3)
>50 D$ = RIGHT$(B$,4)
>60 PRINT C$ + "-" + D$
>70 GOTO 20

```

Antes de rodarmos este programa, vamos analisar sua sequência de instruções (fig.VIII.1)

| | |
|----------------------------|--|
| 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 com a variável numérica A). |
| B\$ = STR\$(A) | (associa a variável numérica A a variável-tipo-cadeia B\$) |
| C\$ = LEFT\$(B\$,3) | (associa os três primeiros caracteres a cadeia B\$ a variável tipo cadeia C\$) |
| D\$ = RIGHT\$(B\$,4) | (associa os quatro últimos caracteres a variável-tipo-cadeia D\$) |
| PRINT C\$ + "-" + D\$ | (faz com que sejam apresentadas em ordem sequencial a cadeia C\$, o carácter hífen (-) e a cadeia B\$) |
| GO TO 20 | (desvia o programa para a linha 20) |

Fig. VIII.1.

A figura VIII.2 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.

```

Fig. VIII.2.

VIII.3.2. VAL(A\$)

VAL é a abreviação de valor. A instrução VAL(A\$) executa exatamente a função inversa de 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 emitir o comando RUN serão apresentados os números:

```
>RUN
1213                25                156
```

Note que enquanto X\$ corresponde á soma das cadeias "12" e "13", Y vale a soma dos números 12 e 13.

VIII.3.3. O Código ASCII

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

Sabemos que o computador opera somente na base binária. Podemos representar caracteres alfabéticos (A, B, C,...), numéricos (0, 1, 2,...) ou simbólicos (+, -, *, /,...) usados no computador, por meio de códigos binários. Poderíamos associar de várias formas estes caracteres a códigos binários, porém a padronização desta associação é muito conveniente é permite a compatibilidade entre diversos produtos, tanto em termos de "software" como de "hardware". Por este motivo, o código ASCII é um dos mais populares.

À letra A no código ASCII é representada pelo número binário 01000001 que é 65 no sistema decimal. O "L" em ASCII é representado por 01001100 que é 76 na base decimal. Cada caracter alfabético, numérico ou simbólico corresponde a um código ASCII específico e para facilitar seu reconhecimento, pode-se representá-lo na base decimal. À tabela a seguir indica código ASCII na base decimal dos caracteres alfabéticos. Uma vez que um byte é formado por 8 bits, o que torna possível 256 combinações ($2^8 = 256$), os caracteres representados pelo código ASCII na base decimal estão sempre entre 0 e 255.

| CARACTER | 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 |

| CARACTER | 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 |

Tabela VIII.6

Na linguagem BASIC, usa-se duas instruções para converter valores ASCII em caracteres e vice-versa, conforme descrevem os próximos sub-itens.

VIII.3.3.a ASC(A\$)

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

ASC(cadeia)

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

```
>PRINT ASC("A")
65
>PRINT ASC("AB")
65
```

O computador apresenta 65 nos dois exemplos acima. Podemos concluir portanto que o caracter B no segundo exemplo não fez diferença e que, o computador apresenta o valor no código ASCII em decimal.

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

1 programa:

```
>NEW
>10 A$ = "MICRO"
>20 B$ = MID$(A$,3,1)
>30 PRINT B$; "="; ASC(B$)
>40 END
```

2 programa:

```
>NEW
>10 A$ = "MICRO"
>20 PRINT MID$(A$,3,1); "="; ASC(MID$(A$,3,1))
>30 END
```

Nos exemplos apresentados, sendo A\$ relacionado com a cadeia MICRO, obtemos o terceiro caracter 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.

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

VIII.3.3.b. CHR\$(X)

CHR\$ indica character. CHR\$ é usado para transformar o código ASCII decimal em caracteres ASCII. Por exemplo:

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

Conforme podemos observar, esta é a operação inversa de PRINT ASC("A").

Para construir uma tabela de valores decimais e seus correspondentes símbolos ASCII, pode-se executar o próximo programa. Este programa começa com o valor decimal 32, uma

vez que antes de 32 há alguns códigos de controle de tela que podem influenciar a operação do computador.

```
>NEW
>10 HOME
>20 FOR I = 32 TO 94
>30 PRINT I; "    "; CHR$(I),
>40 FOR K = 1 TO 500: NEXT K
>50 NEXT I
>60 END
```

Após digitar RUN, podemos observar formação da tabela a seguir:

| | | | | | |
|----|---|----|----|----|---|
| 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 | ^ |

Note que o caracter correspondente ao número F2 está em branco. Isto porque, conforme já pudemos observar, para o computador, o espaço em branco também é considerado um caracter, cujo valor ASCII decimal é 32.

VIII.4. MATRIZES

As variáveis apresentadas até agora eram sempre relacionadas a um único valor por vez. Através das matrizes, pode-se associar mais que um valor a uma variável. Por exemplo, suponhamos que se tenha uma fileira de 5 caixas. Podemos nos referir a esta fileira de 5 caixas pelo 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.

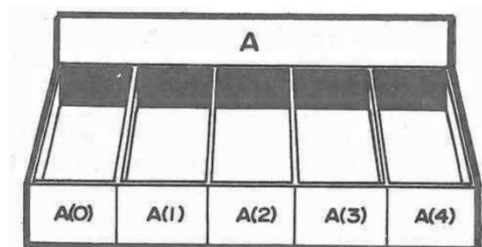


Fig. VII.3.

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

De forma semelhante podemos ter matrizes bidimensional. Assim, os preços e veículos, de acordo com o ano de fabricação, podem ter seus valores registrados numa matriz bidimensional. Para encontrarmos então um determinado valor deveremos fornecer o modelo do veículo (na coluna) e seu ano de fabricação (linha).

| | 1983 | 1982 | 1981... |
|----------------|-----------|-----------|-----------|
| Passat SL | 4.740.325 | 3.191.782 | 1.732.521 |
| Ford Corcel SL | 4.605.713 | 2.640.128 | 1.631.677 |
| Belina SL | 4.934.501 | 2.697.151 | 2.141.120 |
| Monza SL | 5.245.292 | 2.937.286 | - |
| Fiat SL | 4.165.292 | 2.341.526 | 1.437.261 |

Tabela VIII..7.

A figura a seguir representa uma matriz bidimensional:

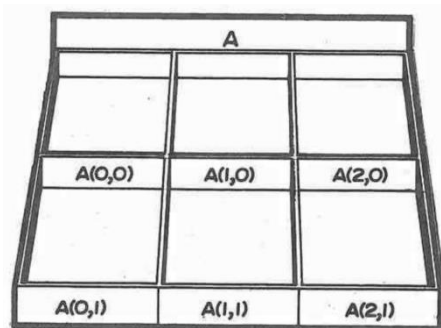


Fig. VIII.4

A matriz representada tem dimensões 2x3, Ou seja, tem duas linhas e três colunas.

Numa matriz tridimensional, além de colunas e linhas, podemos ter níveis. Suponha que, num prédio, não saibamos a numeração dos apartamentos. Poderemos localizar então um determinado apartamento dizendo: 3o andar (nível), de frente para a rua (linha), do canto direito (coluna). Na figura VIII.5 é representada uma matriz tridimensional cujas dimensões são 2x3x2 que corresponde a 2 linhas, 3 colunas por 2 níveis.

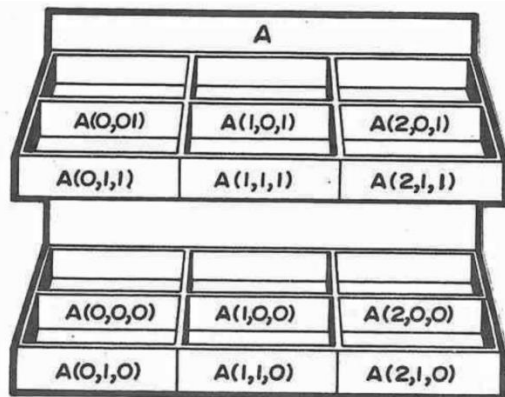


Fig. VIII.5

Poderíamos ainda ter mais de três dimensões numa matriz porém, a representação visual além da tridimensional não é possível.

No TK-2000 COLOR, as matrizes podem ainda ser divididas em três grupos:

- a) variáveis-tipo-matriz inteiras, tais como:
A%(8,5), B%(2,4)
- b) variáveis-tipo-matriz reais, tais como:
A(9,9), D(5,9)
- c) variáveis-tipo-matriz cadeias, tais como:
A\$(10,5), C\$(6,8)

VIII.4.1 DIM

A instrução DIM é imprescindível para o uso de matrizes num programa. Uma instrução permite que se dimensione na memória do TK-2000 a área necessária para conter todos os dados de cada matriz.

VIII.4.1.a Matrizes Unidimensionais

DIM A(20) é uma matriz unidimensional, e indica que o nome da matriz é A e existem ao todo 21 variáveis, de A(0) até A(20). O método de acesso a estas 21 variáveis é o mesmo que usado normalmente para qualquer variável comum. Toda a variável contida numa matriz de quaisquer dimensões e sempre acessível. Por exemplo, pode-se utilizar uma ordem do tipo:

$$A(5) = 32 + A(6)$$

Esta ordem faz com que, após adicionar 32 ao número associado a variável A(6), coloque-se o resultado em A(5). Como já descrito, 5 e 6 são chamados 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
>80 END
```

Ao rodarmos o programa, teremos então o seguinte resultado:

```
>RUN
1      1      1
1      1      1
1      1      1
1      1      1
```

Vamos analisar o fluxo deste programa na figura VIII.6

```

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

```

Fig. VIII.6

Vamos agora executar o programa:

```

>NEW
>10 DIM A%(9)
>20 A%(I) = 1
>30 FOR I = 2 TO 9
>40 E = I - 1
>50 A%(I) = A%(E) + I
>60 PRINT "A% ("; I; ") = "; A%(I),
>70 NEXT I
>80 END

```

Teremos como resultado deste programa:

```

>RUN
A%(2) = 3   A%(3) = 6
A%(4) = 10  A%(5) = 15
A%(6) = 21  A%(7) = 28
A%(8) = 36  A%(9) = 45

```

VIII.4.1.b. Matrizes Bidimensionais

O programa seguinte mostra como utilizar matrizes de duas dimensões.


```

>NEW
>10 DIM A(3,3)
>20 FOR I = 1 TO 3
>30 FOR J = 1 TO 3
>40 A(I,J) = I + J
>50 NEXT J
>60 NEXT I
>70 FOR I = 1 TO 3
>80 FOR J = 1 TO 3
>85 PRINT "A("; I; ", "; J; ") = "; A(I,J); "    ";
>90 NEXT J
>100 PRINT
>110 NEXT I
>120 END

```

Segue sua execução

```

>RUN

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

```

VIII.4.1.c Matrizes Tridimensionais

Observe no exemplo a seguir o uso de matrizes de três dimensões.

```

>NEW
>10 DIM A(3,3,3)
>20 FOR I = 1 TO 3
>30 FOR J = 1 TO 3
>40 FOR K = 1 TO 3
>50 A(I,J,K) = I * J * K
>60 NEXT K : NEXT J : NEXT I
>70 FOR I = 1 TO 3
>80 FOR J = 1 TO 3
>90 FOR K = 1 TO 3
>100 PRINT "A("; I; ", "; J; ", "; K; ") = ";
      A(I,J,K); "    ";
>110 NEXT K
>130 PRINT
>140 NEXT J : NEXT I
>150 END

```

Verifique você mesmo o resultado deste programa.

VIII.4.2 STORE E RECALL

Estes comandos são específicos para gravação ou leitura em fita cassete de arrays numéricos, inteiros ou reais.

Deve-se observar que estes arrays devem ter sido dimensionados igualmente na gravação e na posterior leitura.

Tome como exemplo o programa do item VIII.4.1.c., na linha 150 poderia-se colocar:

```
>150 STORE A
>160 END
```

e com o gravador ligado em RECORD, a matriz ficaria armazenada em fita cassete para uso posterior.

Para se recuperar a matriz, utiliza-se o comando RECALL:

```
>10 DIM X(3,3,3)
>20 RECALL X
>30 FOR I = 1 TO 3
>40 FOR J = 1 TO 3
>50 FOR K = 1 TO 3
>60 PRINT "X(";K;";";J;";";I;") = ";X(K,J,I)
>70 NEXT : NEXT
>80 END
```

Note que para se recuperar a matriz não é necessário se dar o mesmo nome para o comando DIM, porém as dimensões da matriz devem ser as mesmas, ou na pior das hipóteses, a terceira dimensão poderá ser maior no RECALL, como por exemplo DIM X(5,3,5). Neste caso, os pontos excedentes da matriz estão contendo dados aleatórios.

VIII.4.3. COMANDO MOTOR

O comando MOTOR é utilizado para se ligar ou desligar o gravador através de seu controle remoto.

Como podem ser usados dois gravadores, há duas saídas, atrás de seu TK-2000 COLOR específicas para este comando.

```
MOTOR0 - desliga o gravador A
MOTOR1 - liga o gravador A
MOTOR2 - desliga o gravador B
MOTOR3 - liga o gravador B
```

Este é um étimo comando no caso de atualização de matrizes, no qual, num gravador estão as matrizes antigas e no outro irão ser gravadas as matrizes atualizadas.

Recomenda-se usar um comando FOR X = 1 TO 100 : NEXT após cada comando de MOTOR1 ou MOTOR3 devido a inércia dos motores dos gravadores e antes de cada comando MOTOR2 ou MOTORZ pelo mesmo motivo.

FACA O SEU RESUMO

1. As operações comparativas e seus respectivos símbolos no TK-2000 COLOR são: (<), (>), (=), diferente de (.....), e menor ou igual a (.....)
2. Quando uma comparação for verdadeira, seu resultado será é quando falsa
3. Às operações lógicas permitidas pelo TK-2000 COLOR são (E), (OU) e (negação).
4. Os resultados das operações abaixo são:
..... AND = 1
1 AND = 0
0 AND 0 =
1 OR 1 =
0 OR = 1
..... OR = 0
..... 1 = 0
5. A instrução (X) converte o valor do número X em uma cadeia.
6. O resultado da ordem PRINT A\$ + B\$, é a apresentação das cadeias A\$ e B\$ em
7. À instrução que permite a uma cadeia A\$ formada por algarismos numéricos ser transformada em variáveis numéricas é : (A\$)
8. O código utilizado pelo TK-2000 COLOR, que relaciona caracteres a números na base binária é o
9. Para que o valor ASCII na base decimal, da letra C seja apresentado no vídeo, pode-se usar a ordem:
PRINT ("C")
10. De forma inversa, se quisermos que o caracter associado ao número 77 seja apresentado na tela, podemos emitir a ordem: PRINT (77)
11. A instrução DIM permite uma a ser utilizada no programa.
12. A instrução: DIM A\$(8,5) define uma matriz de dimensões, com até variáveis

Respostas

1. menor que, maior que, igual, <>, maior ou igual a, >=, <=; 2. 1, 0; 3. AND, OR, NOT; 4. 1, 1, 0, 0, 1, 1, 0, 0, NOT; 5. STR\$; 6. Sequência; 7. VAL; 8. ASCII; 9. ASC; 10. CHR\$; 11. dimensionar, matriz; 12. Duas, 40, inteiras.

CAPÍTULO 9

CAPITULO IX- INSTRUÇÕES DE ENTRADA E SAIDA

As instruções descritas neste capítulo tem por função a entrada e saída de dados em um programa ou a apresentação de dados na tela. Uma instrução de entrada e uma de saída já foram utilizada em capítulos anteriores: são elas respectivamente: INPUT e PRINT. Os itens a seguir apresentarão as novas instruções.

IX.1. DATA READ

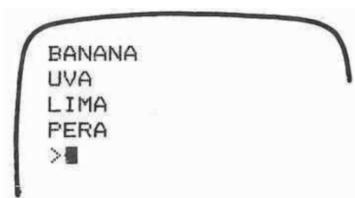
As instruções DATA e READ são normalmente usadas em conjunto, ou sejam se um programa contiver uma ou mais instruções DATA, deverá ter pelo menos uma instrução READ.

À instrução DATA permite definir um conjunto de dados, separados por vírgulas, que deverão ser lidos sequencialmente 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 ela seja colocada em qualquer outra parte do programa. Os exemplos a seguir esclarecem a utilização destas instruções.

Exemplo 1.

```
>NEW  
>10 DATA "BANANA", "UVA", "LIMA", "PERA"  
>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$  
>110 END
```

A figura IX.1 apresenta os resultados deste programa.



```
BANANA  
UVA  
LIMA  
PERA  
>■
```

Fig. IX.1

Exemplo 2:

O resultado do primeiro exemplo pode também ser obtido usando-se apenas uma vez a instrução READ.

```

>NEW
>10 DATA "BANANA", "UVA", "LIMA", "PERA"
>20 HOME
>30 READ A$, B$, C$, D$
>40 PRINT A$
>50 PRINT B$
>60 PRINT C$
>70 PRINT D$
>80 END

```

Exemplo 3:

Pode-se também usar apenas um nome de variável:

```

>NEW
>10 DATA "BANANA", "UVA", "LIMA", "PERA"
>20 HOME
>30 FOR I = 1 TO 4
>40 READ A$
>50 PRINT A$
>60 NEXT I
>70 END

```

Exemplo 4:

Os dados utilizados podem ser distribuídos em mais de uma instrução DATA.

```

>NEW
>10 DATA "BANANA"
>20 DATA "UVA", "LIMA"
>30 DATA "PERA"
>40 HOME
>50 FOR I = 1 TO 4
>60 READ A$
>70 PRINT A$
>80 NEXT I
>90 END

```

Esta modalidade é particularmente til quando a quantidade de dados é muito grande.

Exemplo 5:

A instrução DATA também pode ser 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 "BANANA", "UVA", "LIMA", "PERA"
>70 END

```

Exemplo 6:

Da mesma forma que no início ou no fim, a instrução DATA pode ser incluída durante o programa.

```
>NEW
>10 HOME
>20 DATA "BANANA", "UVA", "LIMA", "PERA"
>30 FOR I = 1 TO 4
>40 READ A$
>50 PRINT A$
>60 NEXT I
>70 END
```

Exemplo 7:

Os dados contidos na instrução DATA podem ser cadeias, inteiros ou reais, devendo-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, "BANANAS", "UVAS"
>20 DATA "LIMAS", "PERAS"
>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$
>90 END
```

Apos digitar o comando RUN, teremos uma tela na forma mostrada na fig. IX.2

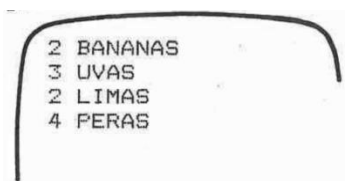


Fig. IX.2

Exemplo 8:

Os dados lidos pela instrução READ e associados a nome de variáveis podem ser usadas 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
>50 END
```

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

Exemplo 9:

O uso 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
>80 END
```

Após emitir RUN teremos:

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

Exemplo 10:

Se nem todos os dados da instrução DATA forem 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
```

O resultado deste programa será:

```
>RUN
51      62      74
```

Importante: No uso da instrução READ deve-se estar atento aos seguintes fatos:

- 1) associar o nome-de-variável correto a cada tipo de dado (cadeia, real ou inteiro).
- 2) a operação READ não pode ser utilizada mais vezes do que o número de dados contidos na(s) instrução(ões) DATA.

Por exemplo:

```
>NEW
>10 DATA "BANANA", "UVA", "LIMA", "PERA"
>20 FOR I = 1 TO 10
>30 READ A$
>40 NEXT I
```

Este tipo de erro faz com que o TK-2000 COLOR apresente a mensagem de erro:

```
?NAO HA MAIS DATA #ERRO EM 30
```

IX.2. RESTORE

A instrução RESTORE faz com que os dados da instrução DATA voltem à ser lidos de maneira que os mesmos possam ser novamente manipulados. Observe o exemplo abaixo:

```
>NEW
>10 DATA 1, 2, 3
>20 HOME
>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
>110 END
```

O resultado deste programa éé apresentado na figura a seguir:

```
1
1    2
1    2    3
```

Como se pode observar, cada vez que surge a instrução RESTORE, os dados de DATA voltam a ser lidos desde o início pela próxima instrução READ.

O programa a seguir apresenta as notas de 4 bimestres de uma determinada matéria da escola e 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.

```

>NEW
>10 REM NOTAS DOS BIMESTRES
>20 DATA 8, 7, 5, 9
>30 REM PESOS DOS BIMESTRES
>40 DATA 1, 2, 3, 4
>45 HOME
>50 PRINT "AS NOTAS SAO "
>60 READ A, B, C, D
>70 PRINT "PRIMEIRO BIMESTRE- "; A
>80 PRINT "SEGUNDO BIMESTRE- "; B
>90 PRINT "TERCEIRO BIMESTRE- "; C
>100 PRINT "QUARTO BIMESTRE- "; D
>110 PRINT : PRINT
>120 RESTORE
>130 READ B1, B2, B3, B4, X, Y, Z, W
>140 F = B1 * X + B2 * Y + B3 * Z + B4 * W
>150 G = X + Y + Z + W
>160 H = F / G
>170 PRINT " A MEDIA ANUAL E : "; H
>180 END

```

À figura abaixo indica o resultado deste programa:

```

AS NOTAS SAO :
PRIMEIRO BIMESTRE- 8,
SEGUNDO EIMESTRE- 7
TERCEIRO RIMESTRE- 5
QUARTO BIMESTRE- 9

A MEDIA ANUAL E 1: 7.5

```

Fig. IX.4

Como se pode observar, os dados da linha 20 foram utilizados duas vezes.

IX.3. GET

A forma geral da instrução GET é:

```
GET nome-de-variável
```

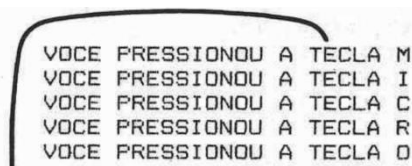
Esta instrução executa as seguintes operações:

1. Interrompe o programa
2. Aguarda a digitação pelo operador de uma e apenas uma tecla correspondente a um carácter.
3. Associa o carácter digitado ao nome-de-variável contido na instrução.
4. Prossegue o programa após a digitação do carácter pelo operador.

O programa a seguir fornece um exemplo do uso da instrução GET.

```
>NEW
>10 HOME
>20 PRINT "VOCE PRESSIONOU A TECLA ";
>30 GET A$
>40 PRINT A$
>50 GOTO 20
```

Se apos o comando RUN, pressionar as teclas M, I, C, R e O, a cada vez que surgir a mensagem "VOCE PRESSIONOU A TECLA ", será obtida uma tela semelhante à figura abaixo.



```
VOCE PRESSIONOU A TECLA M
VOCE PRESSIONOU A TECLA I
VOCE PRESSIONOU A TECLA C
VOCE PRESSIONOU A TECLA R
VOCE PRESSIONOU A TECLA O
```

Fig. IX.5

Esta instrução é muito útil quando usada com as instruções condicionais, conforme será visto no capítulo XI.

IX.4. DEF FN

A instrução DEF FN permite que se defina funções no TK-2000 COLOR, conforme demonstra o programa:

```
>NEW
>5 HOME
>10 PRINT "X", "FN A(X)"
>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
>60 END
```

O resultado do programa é do tipo.

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

Fig. IX.6

Ou seja, o programa aplicou os valores $X = 0$ a 5 a função $FN A(X) = X + 3X + 1$. Vamos conferir os resultados.

| | | |
|---------|---------------|----------------------|
| $X = 0$ | \rightarrow | $0 + 3 * 0 + 1 = 1$ |
| $X = 1$ | \rightarrow | $0 + 3 * 1 + 1 = 5$ |
| $X = 2$ | \rightarrow | $0 + 3 * 2 + 1 = 11$ |
| $X = 3$ | \rightarrow | $0 + 3 * 3 + 1 = 19$ |
| $X = 4$ | \rightarrow | $0 + 3 * 4 + 1 = 29$ |
| $X = 5$ | \rightarrow | $0 + 3 * 5 + 1 = 41$ |

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 $FN A(X)$.

```
>NEW
>10 PRINT "X", "FN A(X)"
>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
>60 PRINT
>70 PRINT "I", "FN A(I)"
>80 FOR I = 1 TO 3
>90 PRINT I, FN A(I) + 8
>100 NEXT I
>110 END
```

Após emitirmos o comando RUN teremos:

| | |
|------|---------|
| >RUN | |
| X | FN A(X) |
| 1 | 5 |
| 2 | 11 |
| 3 | 19 |
| 4 | 29 |
| 5 | 41 |
| I | FN A(I) |
| 1 | 13 |
| 2 | 19 |
| 3 | 27 |

FAÇA O SEU RESUMO

1. A instrução DATA permite definir um conjunto de , separados por e que serão lidos sequencialmente pela instrução
2. A instrução DATA pode ser incluída parte do programa.
3. Os dados a serem lidos pela instrução podem ser distribuídos em instruções DATA.
4. Os dados contidos na instrução DATA podem ser inteiros, ou , porém na instrução eles devem ser associados ao-.....-..... correto.
5. A não ser quando se usa a instrução RESTORE, o número de leituras executadas por instruções não pode ser superior ao número de da instrução DATA. Caso este fato ocorra, o programa será interrompido e o TK-2000 COLOR apresentará uma mensagem de
6. A instrução RESTORE faz com que os dados da instrução DATA a ser lidos, a partir do dado, pela próxima instrução
7. À forma geral da instrução GET é:
<GET> <.....-.....-.....>
8. A instrução GET executa as seguintes operações:
..... a execução do programa, aguarda a digitação de pelo operador; associa o digitado ao nome-de-variável contido na instrução; então o programa.
9. A instrução permite que se defina uma função em um programa.

Respostas.

1. dados, vírgulas, READ; 2. em qualquer; 3. READ, diversas; 4. reais, cadeias, READ, nome-de-variável; 5. READ, dados, erro; 6. voltem, primeiro, READ; 7. nome-de-variável; 8. interrompe, uma tecla, caracter, prossegue; 9. DEF FN;

CAPÍTULO 10

CAPITULO X- TRAÇANDO GRAFICOS E FIGURAS

Embora o conteúdo deste capítulo também faça parte das instruções de saída, não foi apresentado no capítulo anterior por se tratar de um grupo de instruções muito especiais, que permite a utilização da capacidade gráfica e definição de cores (se estiver ligado a uma TV colorida) do TK-2000 COLOR.

X.1. COLOR

A instrução COLOR permite a definição de cores no seu IV (se o seu TK-2000 COLOR estiver ligado a um televisor branco e preto, esta instrução será inútil).

Quando o microcomputador é usado para o traçado de gráficos coloridos, raramente é necessária a gama muito variada de cores. O TK-2000 COLOR permite que se usem até 6 diferentes cores: num mesmo gráfico.

A tabela X.1 indica os números que se relacionam as diferentes cores.

| | | |
|--------------|------------|---------------|
| 0 - preta | 6 - cyan | 11 - branca |
| 1 - azul | 7 - branca | 12 - branca |
| 2 - verde | 8 - verde | 13 - vermelho |
| 3 - branca | 9 - azul | 14 - cyan |
| 4 - azul | 10 - verde | 15 - branca |
| 5 - vermelho | | |

Tabela X.1

A instrução COLOR é usada necessariamente com a instrução PLOT, conforme descreve o próximo item.

X.2. PLOT, GR e TEXT

Usar o TK-2000 COLOR para "plotar" significa, definir as coordenadas horizontal e vertical e então colocar um ponto nesta posição. O TK-2000 COLOR divide a tela em 48 colunas (coordenada vertical) e 40 colunas (coordenada horizontal). As coordenadas tem então valores de 0 a 39 na horizontal e 0 a 47 na vertical.

A forma básica da instrução PLOT é:

PLOT X,Y

onde X corresponde a coordenada horizontal e Y a vertical.

Embora possa-se definir pontos isolados na tela, este procedimento não é usual pois, além de não formar figura nenhuma

a visualização se torna difícil.

O modo gráfico é iniciado pela instrução GR é desativado pelo comando TEXT.

Os programas a seguir ilustram o uso de GR, COLOR e PLOT, fornecendo respectivamente exemplos do traçado de uma reta vertical azul e uma horizontal azul verde (cyan).

Exemplo 1:

```
>NEW  
>10 HOME  
>20 GR  
>30 COLOR = 1  
>40 FOR I = 0 TO 39  
>50 PLOT 20, I  
>60 NEXT I  
>70 END
```

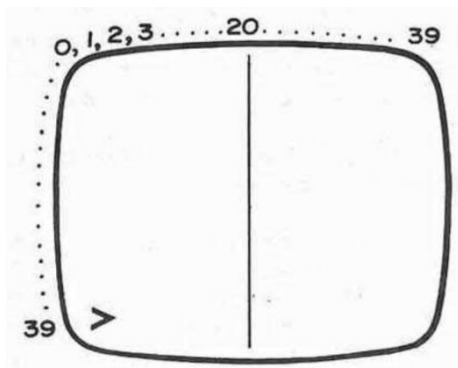


Fig. X.1 Resultado do exemplo 1

OBSERVAÇÃO: caso não surgirem as cores, procure ajustar a sintonia fina do canal de seu TV.

Exemplo 2:

```
>TEXT  
>NEW  
>10 HOME  
>20 GR  
>30 COLOR = 6  
>40 FOR I = 0 TO 39  
>50 PLOT I, 20  
>60 NEXT I  
>70 END
```

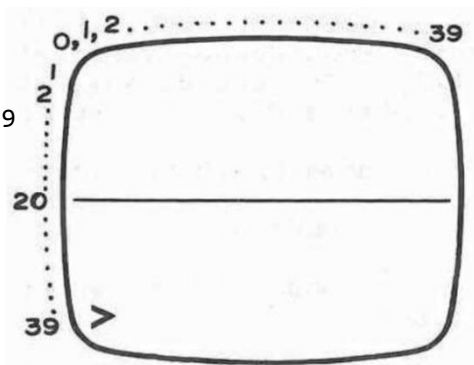


Fig. X.2 Resultado do exemplo 2

Há ainda duas instruções através das quais podemos desenhar linhas, como veremos a seguir.

X.3. HLIN e VLIN

HLIN e VLIN indicam, respectivamente linha horizontal e linha vertical.

Através de HLIN, obtemos com menos instruções, os mesmos resultados do 1 exemplo da seção anterior:

```
>TEXT
>NEW
>10 HOME
>20 GR
>30 COLOR = 1
>40 VLIN 0,39 AT 20
>50 END
```

De forma semelhante, o 2o exemplo da seção anterior pode ser refeito através das instruções.

```
>TEXT
>NEW
>10 HOME
>20 GR
>30 COLOR = 6
>40 HLIN 0,39 AT 20
>50 END
```

X.4. SCRN

A instrução SCRN associa um determinado ponto da tela ao número da cor ao qual esta corresponde. Vamos utilizar esta instrução no último exemplo da seção anterior e verificar o resultado.

```
>TEXT
>NEW
>10 HOME
>20 GR
>30 COLOR = 6
>40 HLIN 0,39 AT 20
>50 PRINT SCRN(20,20)
>60 PRINT SCRN(8,7)
>70 END
```

No caso do programa acima, após a ordem RUN, além do traçado da linha, a tela apresentará os números 6 e y, que são correspondentes a cor cyan e preta (ver tabela X.1) respectivamente, uma vez que o ponto (20,20) da tela foi definido como verde e o (8,7) como branco.

Após o comando TEXT, procure incluir a instrução INVERSE na linha 15 do programa acima e verificar os resultados.

X.5 TRAÇADOS DE ALTA RESOLUÇÃO

Alta Resolução refere-se a traçados com mais detalhes, e melhor definição. A capacidade de alta resolução no TK-2000 COLOR consiste na possibilidade da definição de 53.760 pontos, isto é, a tela serão formada por 53.760 pontos.

Na baixa resolução, usa-se o método de blocos, isto é, em cada coordenada definida é apresentado um bloco, enquanto em alta resolução, usa-se o método de pontos.

X.5.1. HCOLOR

HCOLOR é usada no modo alta resolução, exatamente da forma de COLOR no modo de baixa resolução (ver seção X.1)

X.5.2. HPLLOT, HGR e TEXT

De forma semelhante a PLOT, HPLLOT executa o traçado em alta resolução, porém com bem mais pontos de coordenadas. A figura abaixo indica as coordenadas da tela em modo normal e em alta resolução.

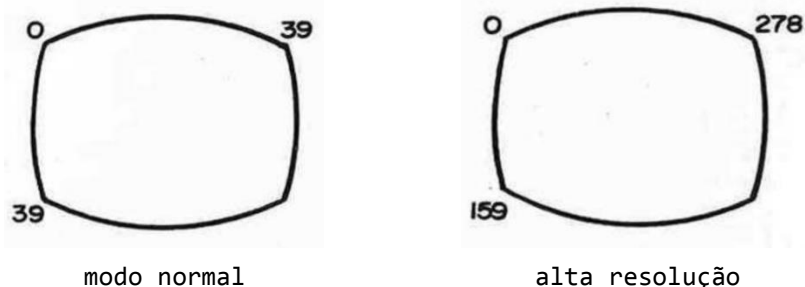


Fig. X.3

A instrução HGR permite que o TK-2000 COLOR entre no modo gráfico de alta resolução. Para voltar ao modo text deve-se digitar o comando TEXT.

Seguem exemplos do uso das instruções HCOLOR, HPLLOT e HGR, bem como as figuras indicando os resultados dos programas.

Exemplo 1: definição de quatro pontos, um em cada canto da tela.

```

>TEXT
>NEW
>10 HGR
>20 HOME
>30 HCOLOR = 5
>40 HPLOT 277,1
>50 HPLOT 1,1
>60 HPLOT 1,158
>70 HPLOT 277,158
>80 END

```

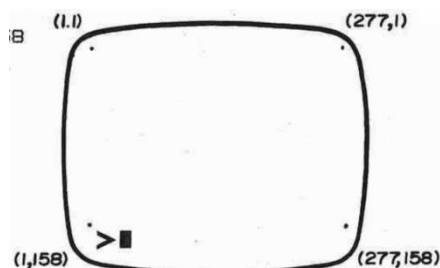


Fig. X.4

Note que os pontos apresentados são quase imperceptíveis.

Digite agora o comando TEXT e siga para o próximo exemplo:

Exemplo 2: traçado de uma reta horizontal no lado superior da tela.

```

>TEXT
>NEW
>10 HGR
>20 HOME
>30 HCOLOR = 5
>40 FOR I = 1 TO 279
>50 HPLOT I,1
>60 NEXT I
>70 END

```

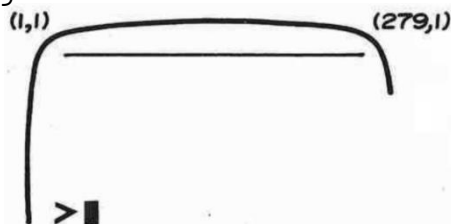


Fig. X.5

Exemplo 3: é executada uma linha vertical próxima ao lado esquerdo da tela (digite a ordem TEXT antes de iniciá-lo).

```

>TEXT
>NEW
>10 HGR
>20 HOME
>30 HCOLOR = 5
>40 FOR I = 1 TO 191
>50 HPLOT 1,I
>60 NEXT I
>70 END

```

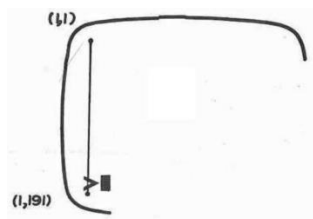


Fig. X.6

Exemplo 4: o programa abaixo desenha um retângulo de lados paralelos aos limites da tela. Neste programa, bem como na respectiva figura são feitas algumas observações.

```
>TEXT
>NEW
>10 HGR : HCOLOR = 3
>20 FOR I = 1 TO 278
>30 HPLLOT I,1
>40 NEXT I
>50 FOR I = 1 TO 190
>60 HPLLOT 278,I
>70 NEXT I
>80 FOR I = 278 TO 1 STEP -1
>90 HPLLOT I,190
>100 NEXT I
>110 FOR I = 190 TO 1 STEP -1
>120 HPLLOT 1,I
>130 NEXT I
>140 END
```

(1) lado superior

(2) lado direito

(3) lado inferior

(4) lado esquerdo

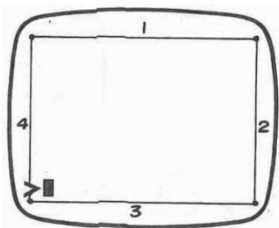


Fig. X.7

Exemplo 5: se os valores assinalados excederem os limites da tela, o TK-2000 COLOR emitira uma mensagem de erro:

```
>TEXT
>NEW
>10 HGR : HCOLOR = 3
>20 HOME
>30 FOR I = 0 TO 300
>40 HPLLOT I,0
```

```
>50 NEXT I
>60 END
```

Após digitar a ordem RUN, surgirá na tela:

```
>RUN
?VALOR ILEGAL #ERRO EM 40
>-
```

Isto porque o valor de I atingiu 280 e o máximo valor aceito pela instrução HPLLOT (máxima coordenada horizontal) é 279.

Exemplo 6: tente, você mesmo prever o resultado do programa a seguir:

```
>TEXT
>NEW
>10 HGR
>20 HOME
>30 HCOLOR = 5
>40 FOR I = 80 TO 110
>45 FOR J = 125 TO 155
>50 HPLLOT J,I
>55 NEXT J
>60 NEXT I
>70 END
```

X.5.3. HPLLOT X1,Y1 TO X2,Y2

Esta é a instrução mais prática para o traçado de retas. Ela faz com que seja feita uma reta entre os pontos de coordenadas X,Y e X,Y. Por exemplo, pode-se traçar uma diagonal na tela conforme mostra o programa:

```
>TEXT
>NEW
>10 HGR
>20 HCOLOR = 2
>30 HPLLOT 1,1 TO 279,150
>40 END
```

Note que, através de uma única instrução pode-se traçar várias retas. Desta forma, o exemplo do traçado de um retângulo com lados paralelos aos limites da tela, apresentado no sub-item anterior, pode também (e de maneira mais rápida) ser obtido através do programa:

```
>TEXT
>NEW
>10 HGR
>20 COLOR = 3
>30 HPLLOT 1,1 TO 258,1 TO 258,157 TO 1,157 TO 1,1
>40 END
```

Observe agora as figuras formadas pelos seguintes programas.

1.

```
>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
>60 END
```

2.

```
>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 1,1
>50 NEXT I
>60 END
```

3.

```
>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
>60 END
```

X.5.4. HGR2

Como se pode verificar, no modo de alta resolução gráfica acionado por HGR, as linhas finais da tela permanecem no modo texto e 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, permitindo a definição de 280 por 192 pontos na tela.

FAÇA O SEU RESUMO

1. A instrução:

COLOR =

permite a definição da cor correspondente ao número X, onde X varia de 0 a 7.

2. Sendo X a coordenada e Y a coordenada , a instrução:

..... X,Y define um ponto neste local da tela.

3. A instrução que permite o uso do modo gráfico é

4. Para se obter o número correspondente a cor do ponto (X,Y) da tela usa-se a instrução

..... (X,Y)

5. Traços horizontais e verticais podem ser obtidos através de instruções únicas, que são respectivamente

..... X,Y TO X

..... X,Y TO Y

6. No traçado normal de gráficos, a tela é dividida nas coordenadas verticais 0 a nas horizontais de 0 a , enquanto em alta resolução as coordenadas verticais vão de 0 a e as horizontais de 0 a

7. HCOLOR é uma instrução usada no modo de alta resolução da mesma forma que no modo de baixa resolução.

8. De forma semelhante a instrução , executa traçados em alta resolução.

9. À instrução:

..... X, Y X, YX, Y

executa o traçado a partir das coordenadas (X,Y) para (X,Y) e finalmente até as coordenadas (X,Y)

10. As instruções que permitem o uso do modo gráfico de alta resolução são ou

11. Para se voltar do modo gráfico ou do modo gráfico de alta resolução para o modo normal (ou modo texto), usa-se o comando

12. O modo de alta resolução gráfica, quando acionado por permite a definição de 280 x 160 pontos, e através de preenche toda a tela, possibilitando a definição de 280 x 192 pontos.

respostas:

1. X; 2. horizontal, vertical, PLOT; 3. GR; 4. SCRN; 5. HLIN, VLIN;

6. 47, 39, 191, 279; 7. COLOR; 8. PLOT, HPLOT; 9. HPLOT, TO, TO;

10. HGR, HGR2; 11. TEXT; 12. HGR, HGR2

CAPÍTULO 11

CAPITULO XI. LIDANDO COM DESVIOS

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

Uma instrução já apresentada que executa desvios é GOTO.

XI.1. Desvios Condicionais

Entende-se por desvios condicionais, desvios ou, de modo geral, operações do programa que só ocorrem em determinadas situações.

Seguem as instruções que provocam este tipo de "desvio".

XI.1.1. IFGOTO

Esta instrução faz com que, "se" (if) a condição apresentada após IF for verdadeira, o programa seja desviado para a linha indicada após GOTO: caso a condição após IF não seja verdadeira, O programa prossegue normalmente na próxima linha:

O programa seguinte mostra o número de vezes que cada face de um dado se apresenta quando atirado 600 vezes.

```
>NEW
>10 FOR I = 1 TO 600
>20 B% = RND(1) * 6 + 1
>30 IF B%=1 GOTO 100
>40 IF B%=2 GOTO 110
>50 IF B%=3 GOTO 120
>60 IF B%=4 GOTO 130
>70 IF B%=5 GOTO 140
>80 A(6) = A(6) + 1
>90 GOTO 200
>100 A(1) = A(1) + 1: GOTO 200
>110 A(2) = A(2) + 1: GOTO 200
>120 A(3) = A(3) + 1: GOTO 200
>130 A(4) = A(4) + 1: GOTO 200
>140 A(5) = A(5) + 1: GOTO 200
>200 NEXT I
>210 FOR I = 1 TO 6
>220 PRINT "A("; I; ") ="; A(I)
>230 NEXT I
>240 END
```

Antes de rodarmos o programa, vamos supor o primeiro ciclo de execução, entre as linhas 10 a 200.

10 FOR I = 1 TO 600 (sendo este o 1 ciclo do programa, é

| | |
|------------------------------|---|
| | associado a 1) |
| 20 B% = RND(1) * 6 + 1 | (Suponha que RND(1) tenha sido .23457, temos então: $9.23457 \times 6 = 1,40742$ $1.410742 + 1 = 2,40742$, B\$ = 2.40742 B% = 2) |
| 30 IF B% = 1 GOTO 100 | (A afirmação após IF é falsa, logo o programa não sofre desvio) |
| 40 IF B% = 2 GOTO 110 | (A afirmação após IF é verdadeira (B%=2), logo o programa é desviado para a linha indicada após GOTO (110), ocorrendo então o 1 ciclo). |
| 110 A(2) = A(2) + 1 GOTO 200 | (Inicialmente como A(2) não estava associado a nenhum valor ele corresponderia a 0. Após a execução desta instrução: $A(2) = 0 + 1$ ou seja $A(2) = 1$. Ocorre então o segundo desvio (este é incondicional) quando o programa é desviado para a linha 200). |
| 200 NEXT I | (E acrescentada uma unidade ao valor de 1 e o programa é desviado para a linha 20). |

Fig. XI.1

Após a ordem de execução do programa, aguarde algum tempo (cerca de 30 s) para que a resposta seja fornecida. Teremos então um resultado do tipo:

```

>RUN
A(1) = 87
A(2) = 96
A(3) = 102
A(4) = 91
A(5) = 127
A(6) = 97

```

XI.1.2. IF THEN

O uso do IF..... THEN é semelhante a IF.... GOTO, no caso da afirmação após IF ser verdadeira, podemos desviar o programa para uma determinada linha (exatamente da mesma forma que com IF.....GOTO) ou executarmos uma determinada operação. Desta forma, o programa apresentado no sub-item anterior pode também ser feito da seguinte maneira:

```

>NEW

```

```

>10 FOR I = 1 TO 600
>20 B% = RND(1) * 6 + 1
>30 IF B% = 1 THEN A(1) = A(1) + 1
>40 IF B% = 2 THEN A(1) = A(2) + 1
>50 IF B% = 3 THEN A(1) = A(3) + 1
>60 IF B% = 4 THEN A(1) = A(4) + 1
>70 IF B% = 5 THEN A(1) = A(5) + 1
>80 IF B% = 6 THEN A(1) = A(6) + 1
>90 NEXT I
>100 FOR I = 1 TO 6
>110 PRINT "A("; I; ") = "; A(I)
>120 NEXT I
>130 END

```

Aguarde alguns segundos para a apresentação da resposta.

X1.1.3. ON.....GOTO

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

ON <número inteiro> GOTO 1 parâmetro, 2 parâmetro, 3 parâmetro,

onde cada parâmetro corresponde a um número de linha do programa. Desta forma, se o número inteiro após ON, for 1 o programa ser à desviado para a linha de programa correspondente ao 1 parâmetro, se for 2, o programa será desviado para o número de linha correspondente ao 2o parâmetro e assim por diante.

O programa apresentado nos sub-itens anteriores, pode ser desenvolvido através da instrução ON....GOTO, conforme se vê abaixo.

```

>NEW
>10 FOR I = 1 TO 600
>20 B% = RND(I) * 6 + 1
>30 ON B% GOTO 100, 110, 120, 130, 140, 150
>100 A(1) = A(1) + 1 : GOTO 200
>110 A(2) = A(2) + 1 : GOTO 200
>120 A(3) = A(3) + 1 : GOTO 200
>130 A(4) = A(4) + 1 : GOTO 200
>140 A(5) = A(5) + 1 : GOTO 200
>150 A(6) = A(6) + 1
>200 NEXT I
>210 FOR I = 1 TO 6
>220 PRINT "A("; I; ") = "; A(I)
>230 NEXT I
>240 END

```

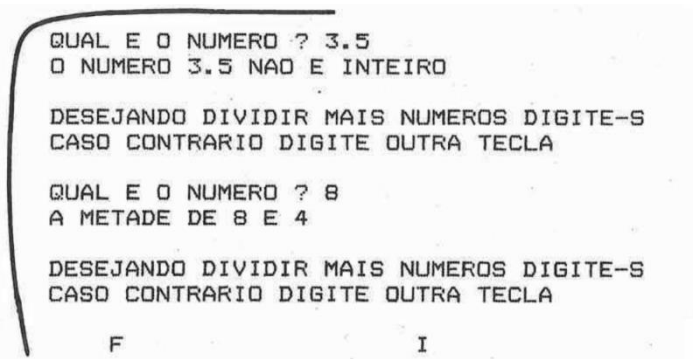
XI.1.4. IF THEN GOTO

Esta instrução 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
>45 REM VERIFICACÃO E DIVISAO
>50 B = INT(A)
>60 IF A = B THEN C = A / 2 : GOTO 100
>70 PRINT "O NUMERO "; A; " NAO E INTEIRO"
>80 GOTO 110
>100 PRINT "A METADE DE "; A; " E "; C
>110 PRINT : PRINT
>120 PRINT "DESEJANDO DIVIDIR MAIS NUMEROS DIGITE-S"
>130 PRINT "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
```

Como se pode ver, o programa utilizou duas instruções condicionais, uma para verificar se o número digitado é ou não inteiro e outra para saber se é ou não desejado que o programa prossiga. Note que em cada caso, de acordo com o dado introduzido nas instruções condicionais, o programa toma uma atitude diferente.

A tela da fig. XI.3 seria apresentada se, após a ordem RUN, sejam 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
CASO CONTRARIO DIGITE OUTRA TECLA

QUAL E O NUMERO ? 8
A METADE DE 8 E 4

DESEJANDO DIVIDIR MAIS NUMEROS DIGITE-S
CASO CONTRARIO DIGITE OUTRA TECLA

F I M
```

Fig. XI.3

XI.2. SUB-ROTINAS

O uso de sub-rotinas é um artifício bastante usado em programação.

Entende-se por sub-rotina, um programa secundário, que se pode usar diversas vezes dentro de um programa maior. Por exemplos, 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 quilo, hora para segundo, 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 programas as sub-rotinas economizam espaço na memória do computador.

XI.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 fez com que o programa volte para sua sequência normal, uma instrução após GOSUB. Observe o exemplo:

ATENÇÃO: Não confundir a instrução RETURN com a tecla RETURN

Exemplo: o programa a seguir pode ser subdividido em duas partes: o programa principal e a sub-rotina. As linhas de número 10 a 70 constituem o programa principal e as linhas 100 a 150 constituem 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          |      programa principal
>50 X = 4 : Y = 5      |
>60 GOSUB 100          |
>70 END                -
>100 REM SUBROTINA     -
>110 FOR I = 1 TO X    |
>120 PRINT Y;          |
>130 NEXT I            |      sub-rotina
>140 PRINT             |
>150 RETURN            -
```

Após a ordem RUN teremos:

```
>RUN
11111
3333333
5555
```

Examinemos agora a sequência de execução do programa:

- À linha 10 indica $X=5$ e $Y=1$
- Na linha 20 GOSUB desvia para a sub-rotina da linha 100
- Executa a sub-rotina e, na linha 150 RETURN faz com que o programa volte para a linha 30, que atribui a variável X o valor 7 e a Y o valor 8.
- As operações citadas são então repetidas para $X=7$, $Y=3$ e $X=4$ e $Y=5$ após o que o programa é finalizado.

XI.2.2. Sub-rotina dentro de uma sub-rotina

As vezes é necessário ter-se uma segunda sub-rotina dentro de uma sub-rotina. Execute o programa:

```
>NEW
>10 REM PROGRAMA PRINCIPAL
>20 DIM A(100)
>30 GOSUB 2000
>40 PRINT "FIM"
>50 END
>2000 REM PRIMEIRO NIVEL DE SUB-ROTINA
>2010 FOR I = 1 TO 5
>2020 A(I) = 5 * I
>2030 GOSUB 3000
>2040 NEXT I
>2050 RETURN
>3000 REM SEGUNDO NIVEL DE SUB-ROTINA
>3010 B(I) = A(I) * 2 + 1
>3020 PRINT "A(";I; " ) = ";A(I), "B(";I;") = ";B(I)
>3030 RETURN
```

Após o comando RUN teremos:

>RUN

| | |
|-----------|-----------|
| A(1) = 5 | B(1) = 11 |
| A(2) = 10 | B(2) = 21 |
| A(3) = 15 | B(3) = 31 |
| A(4) = 20 | B(4) = 41 |
| A(5) = 25 | B(5) = 51 |
| FIM | |

XI.2.3. ON.....GOSUB

O uso de ON.....GOSUB é bastante semelhante ao uso de ON.....GOTO (ver item XI.1.3) porém, ON.....GOSUB ao invés de desviar o programa para determinadas linhas, desvia-o para determinada sub-rotinas. É importante que não se esqueça de finalizar cada uma das sub-rotinas com a instrução RETURN. Verifique os dois exemplos a seguir: o primeiro utilizando a instrução ON.....GOTO e o segundo ON.....GOSUB.

Exemplo 1

```
>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
```

Exemplo 2

```
>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
```

O resultado dos programas é do tipo:

```
>RUN
A(1) = 27
A(2) = 35
A(3) = 38
```

No programa apresentado, determinamos randomicamente I=1,2 ou 3 por meio de RND. Se I=1, então GOSUB será 100; para I=2, GOSUB 200; e I=3, para GOSUB 300. Executamos um por um 10 vezes, e

então apresentamos a estatística dos números de 1, 2 e 3.

XI.2.4. ONERR.....GOTO

Se ocorrer um erro durante a execução do programa, o computador interromperá a sua execução e apresentará uma mensagem de erro. Por exemplo:

```
>NEW
>10 HOME
>20 FOR I = 0 TO 10
>30 PRINT I, 5/I
>40 NEXT I
```

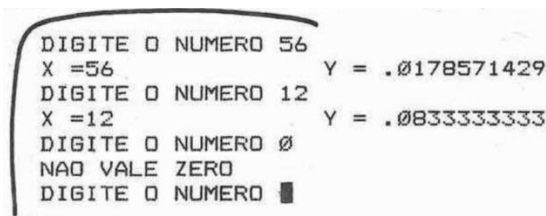
Quando o computador iniciar a execução do programa, ele o interromperá imediatamente, porque sendo $I=0$, o computador não terá possibilidade de executar $5/0$.

```
RUN
0
"DIVISÃO POR ZERO #ERRO EM 30
>■
```

Na linguagem BASIC, a instrução ONERR..... GOTO é a abreviação de "ON ERROR GOTO" (em caso de erro vá para...). Quando ocorre um erro, ONERR.....GOTO pode evitar que a execução do programa seja interrompida, ocasionando um desvio para uma rotina de tratamento adequado de erro, esta rotina deverá necessariamente finalizar 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 "NÃO VALE ZERO"
>110 RESUME
```

Se após darmos a ordem RUN, digitarmos os números 56, 12 e 0, teremos uma tela semelhante a figura XI.4:



```
DIGITE O NUMERO 56
X =56          Y = .0178571429
DIGITE O NUMERO 12
X =12          Y = .0833333333
DIGITE O NUMERO 0
NÃO VALE ZERO
DIGITE O NUMERO ■
```

Fig. XI.4.

Quando um programa acaba se tornando complexo, muitas vezes existe a possibilidade de vários erros diferentes terem a mesma chance de ocorrer. Assim sendo podemos, através da leitura de PEEK (222), determinar o código do erro cometido no programa. A tabela que apresenta os códigos de todos os erros está contida no APENDICE B. Assim sendo se na linha 105 tivéssemos digitado:

```
>110 A= PEEK (222) : PRINT A
```

o programa, além do resultado acima, na ocasião do erro ela também apresentaria o número 134 que corresponde ao erro DIVISAO POR ZERO.

A operação no modo ONERR deve ser indicada normalmente no início do programa, e sempre que acontecer um erro a rotina adequada será chamada, Esta rotina deve ser completada com o comando RESUME. Para desativar o modo ONERR deve ser executado o comando POKE 216,0.

X1.2.5. POP

Uma instrução POP tem efeito semelhante a RETURN, porém não executa o retorno ao programa principal. O próximo RETURN encontrado, ao invés de retornar ao programa uma instrução após o último GOSUB, irá pular para uma instrução após o penúltimo GOSUB.

No exemplo abaixo, inicialmente 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, teremos a apresentação da sequência A, B, C, D e E (uma letra em cada linha). A ordem de execução das linhas de comando é: 10 # 20 # 100 # 110 # 200 # 210 # 120 # 130 # 30 # 40.

Incluindo a linha

```
>205 POP
```

a letra D não mais será apresentada. A nova ordem de execução das linhas de comando passa a ser:

10 # 20 # 100 # 110 # 200 # 205 # 210 # 30 # 40.

Como podemos observar, após a inclusão de POP na linha 205, a instrução RETURN da linha 210 ao invés de desviar o programa para a linha 120, desviou-o para a linha 30, ou seja, o POP anula a última situação de retorno e a anterior passa a constar.

FAÇA O SEU RESUMO

1. Quando a sequência numérica da execução de um programa não é obedecida diz-se que ocorreu um
2. A instrução
<.....> <condição> <GOTO> <.....>
faz com que, se a condição indicada for verdadeira, a execução do programa seja desviada para a linha de número xyz. Caso a condição seja falsa, o programa prossegue na instrução..
3. Na instrução:
<.....> <condição> <THEN> <operação>
a operação indicada ocorrerá se a condição for
4. Se no item anterior, em vez de uma operação for indicado um número de linha do programa após THEN, esta instrução passará a ser idêntica a <.....> <condição> <GOTO> <.....-.....-.....>
5. Desejando-se desviar o programa para diferentes linhas, de acordo com o valor inteiro associado a BA, utiliza-se a instrução:
<.....> <B%> <.....> <aaa> <,> <bbb> <,> <ccc>
onde aaa, bbb, ccc equivalem a números de linhas do programa.
6. Na questão anterior, o programa será desviado para a linha bbb quando o valor de B% for
7. À instrução:
<.....> <condição> <THEN> <operação> <GOTO> <.....-.....-.....>
combina as ações de GOTO e THEN.
8. Uma sub-rotina é um programa, que se pode utilizar diversas vezes dentro de um programa principal.
9. O desvio de um programa para uma sub-rotina iniciada na linha xyz e feito através da instrução:
..... xyz
10. Suponha que a instrução de desvio para uma sub-rotina (.....) se localize na linha n do programa. A última instrução da sub-rotina deve ser, necessariamente, após a qual o programa voltará a ser executado normalmente, a partir da linha +
11. Pode haver diversos níveis de sub-rotinas dentro de um programa, ou seja, dentro de uma sub-rotina pode haver uma instrução que desvia a execução do programa para uma nova sub-rotina. Porém, nunca esqueça: cada uma das sub-rotinas devem ser sempre finalizada pela instrução

12. Pode-se desviar um programa para diversas sub-rotinas, dependendo do valor da variável inteira B% através da instrução: <.....> <B%> <.....> <aaa> <bbb> <ccc> onde aaa, bbb e ccc correspondem as linhas de inicio das sub-rotinas.

13. Na pergunta anterior, se B% for igual a 3, o programa será desviado para a sub-rotina que se inicia na linha

14. Caso ocorra um erro num programa, ao invés deste ser interrompido, ele pode ser desviado para a linha xyz através da instrução <.....> <GOTO> <XYZ>

15. A instrução faz com que a próxima instrução RETURN desvie o programa para a ordem seguinte a penúltima instrução GOSUB.

respostas:

1. desvio; 2. IF, xyz, próxima; 3. IF, verdadeira; 4. IF, número-de-linha; 5. ON, GOTO; 6. 2; 7. IF, número de linha, IF, IF; 8. Secundário; 9. GOSUB; 10. GOSUB, RETURN, n, 1; 11. RETURN; 12. ON, GOSUB; 13. ccc; 14. ONERR; 15. POP

CAPÍTULO 12

CAPITULO XII- FUNÇÕES MATEMATICAS

Além das operações básicas tais como! +, -, *, /. ^ podemos utilizar funções matemáticas. Neste capítulo estudaremos as funções existentes na linguagem BASIC do TK-2000 COLOR. Algumas funções já apresentadas são : INT(X), RND(1) e SQR(X).

XII.1. FUNÇÕES TRIGONOMETRICAS

SIN(X), COS(X) e TAN(X) são as principais funções trigonométricas.

XII.1.1 Função seno-SIN(X)

Na linguagem BASIC utilizada pelo TK-2000 COLOR, as operações com as funções trigonométricas são baseadas na unidade radiano. Isto & o valor entre parênteses deve ser expresso em radianos e não em graus. Lembre-se que 360 corresponde aproximadamente a 6.28 radianos e portanto, cada radiano aproximadamente vale 57.324804 graus.

Agora digite o seguinte programa que forma uma tabela do seno de X. Usamos primeiro a linha número 30 para obter o fator de conversão de graus em radiano, e então na linha 50 os graus são convertidos em radiano (I/R). A conversão é feita para cada 15 graus.

```
>NEW
>10 PRINT "GRAUS"; TAB(8); "RADIANOS"; TAB(24); "SEN(X)"
>20 PRINT
>30 R=360 / 6.28
>40 FOR I = 0 TO 350 STEP 30
>50 Y = SIN (I / R)
>60 PRINT I; TAB(8); I/R; TAB(24); Y
>70 NEXT I
>80 END
```

Após a ordem RUN, teremos a seguinte tabela:

| GRAUS | RADIANOS | SEN(X) |
|-------|------------|----------------|
| 0 | 0 | 0 |
| 30 | .523333333 | .499770103 |
| 60 | 1.04666667 | .8657539839 |
| 90 | 1.57 | .999999683 |
| 120 | 2.09333333 | .8665558 |
| 150 | 2.61666667 | .501148958 |
| 180 | 3.14 | 1.59265337E-03 |
| 210 | 3.66333333 | -.498389979 |
| 240 | 4.18666667 | -.864961682 |
| 270 | 4.71 | -.999997146 |
| 300 | 5.23333333 | -.867349563 |
| 330 | 5.75666667 | -.502526543 |
| 360 | 6.28 | -3.1853027E-03 |

XII.1.2. Função cosseno COS(X)

O programa seguinte executa uma tabela do cosseno de X, de forma semelhante a executada no sub-item anterior.

```
>NEW
>10 PRINT "GRAUS";TAB(8);"RADIANOS";TAB(24);"COS(X)"
>20 PRINT
>30 R=360 / 6.28
>40 FOR I = 0 TO 350 STEP 30
>50 Y = COS(I / R)
>60 PRINT I; TAB(8); I/R; TAB(24); Y
>70 NEXT I
>80 END
```

Teremos então:

| GRAUS | RADIANOS | COS(X) |
|-------|------------|-----------------|
| 0 | 0 | 1 |
| 30 | .523333333 | .866158094 |
| 60 | 1.04666667 | .500459689 |
| 90 | 1.57 | 7.96326206E-04 |
| 120 | 2.09333333 | -.4990802 |
| 150 | 2.61666667 | -.865361035 |
| 180 | 3.14 | -.999998732 |
| 210 | 3.66333333 | -.866952956 |
| 240 | 4.18666667 | -.501837909 |
| 270 | 4.71 | -2.28897806E-03 |
| 300 | 5.23333333 | .497699443 |
| 330 | 3.75666667 | .864561781 |
| 360 | 6.28 | .999994927 |

XII.1.3. Função Tangente- TAN(X)

O programa abaixo calcula os valores de tangente de X de maneira semelhante ao COS(X) e SIN(X).

```
>NEW
>10 PRINT "GRAUS";TAB(8);"RADIANOS";TAB(24);"TAN(X)"
>20 PRINT
>30 R = 360 / 6.28
>40 FOR I = 1 TO 360 STEP 30
>50 Y = TAN (I/R)
>60 PRINT I;TAB(8);I/R;TAB(24);Y
>70 NEXT I
>80 END
```

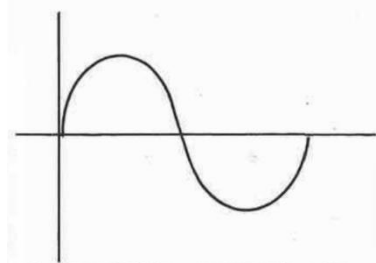
Após digitar RUN & apresentado:

| GRAUS | RADIANOS | TAN(X) |
|-------|------------|-----------------|
| 0 | 0 | 0 |
| 30 | .523333333 | .5769964 |
| 60 | 1.04666667 | 1.72992922 |
| 90 | 1.57 | 1255.76523 |
| 120 | 2.09333333 | -1.73630571 |
| 150 | 2.61666667 | -.579121243 |
| 180 | 3.14 | -1.59265539E-03 |
| 210 | 3.66333333 | .574875459 |
| 240 | 4.18666667 | 1.723958777 |
| 270 | 4.71 | 418.587575 |
| 300 | 5.23333333 | -1.74271757 |
| 330 | 5.75666667 | -.581250008 |
| 360 | 6.28 | -3.18531886E-03 |

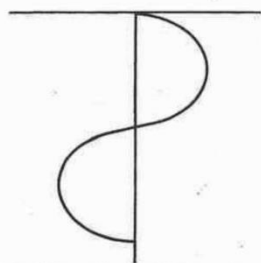
XII.2. TRAÇADOS GRAFICOS DE FUNÇÕES

XII.2.1. Diagramas Simples

O TK-2000 COLOR é capaz tanto de calcular valores numéricos como de traçar gráficos de funções. No traçado de funções trigonométricas, temos que girar o eixo das coordenadas em 90 no sentido horário, conforme a figura XII.1



a) sentido normal de execução do gráfico



b) rotação do eixo do gráfico em 90 no sentido horário, na tela do TK-2000 COLOR

Fig. XII.1

O número de caracteres é limitado pela tela.

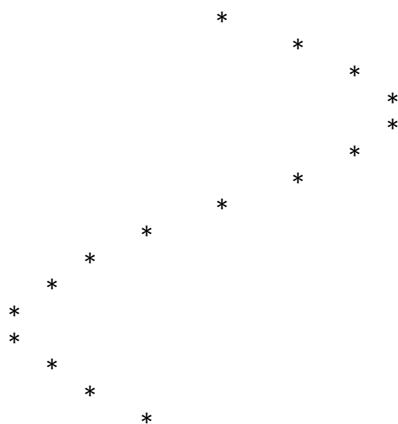
Sabemos que o valor da função seno varia entre +1 e -1. A tela do TK-2000 COLOR tem somente 40 caracteres para cada linha, de forma que devemos usar 20 como ponto médio. Faça o programa a seguir. O ponto chave deste programa se situa nas linhas 20 e 30.

```

>NEW
>10 FOR I = 0 TO 360 STEP 20
>20 Y = SIN(I/57.299578)
>30 K% = Y * 19 + 20
>40 PRINT TAB(K%);""
>50 NEXT I
>60 END

```


Na linha 20, Y é o valor de seno, e seus valores são entre -1 e +1; quando Y é multiplicado por 19, seus valores passam a variar entre -19 e +19, e aí 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 30 apresenta o símbolo "*" em TAB(K%), assim obteremos o diagrama da figura XII.2.



XII.2.2. Gráfico da Função Seno

Depois de observar o programa do sub-item anterior, compare as diferenças com o seguinte diagrama:

```
>NEW
>10 FOR I = 0 TO 39
>20 PRINT ".";
>30 NEXT I
>40 PRINT
>50 FOR I = 0 TO 360 STEP 20
>60 Y = SIN(I/57.29578)
>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
>140 END
```

A figura XII.3 ilustra o resultado deste programa.

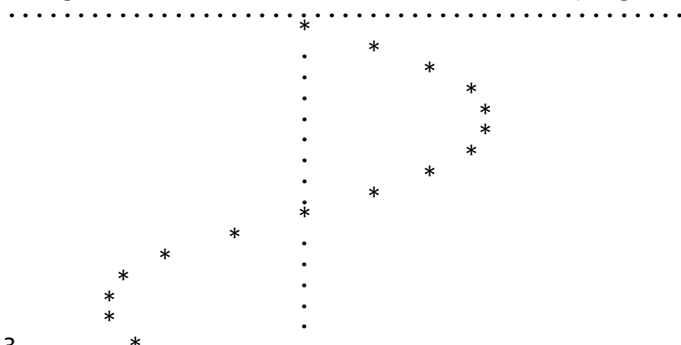


fig XII.3

XII.2.3. Gráfico da Função Cosseno

Verifique o programa abaixo e compare seus resultados com o programa apresentado no sub-item anterior. A figura XII.4 indica o resultado de sua execução.

ORSEVAÇÃO: Se você tiver digitado o programa anterior, não digite NEW pois, para executar este programa basta reentrar com a linha 60 alterada.

```
>10 FOR I = 0 TO 39
>20 PRINT ".";
>30 NEXT I
>40 PRINT
>50 FOR I = 0 TO 360 STEP 20
>60 Y = COS(I/57.29578)
>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
>140 END
```

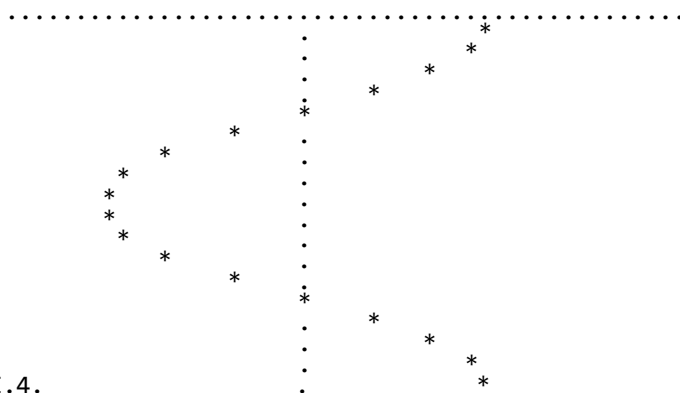


Fig. XII.4.

XII.2.4. OUTRAS FUNÇÕES

XII.2.4.a. Função Randômica - RND(X)

Há três tipos de funções randômicas no TK-2000 COLOR: 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 apresentará:

```

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

```

Dos resultados anteriores podemos observar que:

- RND(1) fornece um número entre 0 e 0.999999999
- RND(0) repete o número fornecido pela instrução RND anterior
- Cada RND(-X) seleciona um número diferente e de valor bastante pequeno

XII.2.4.b Função Valor Absoluto - ABS(X)

Em matemática às vezes é necessário obter-se valores absolutos. Fara tanto usa-se a instrução:

ABS(X)

Verifique o programa abaixo:

```

>NEW
>10 FOR I = -10 TO 1 STEP 5
>20 PRINT "ABS(";I;") = ";ABS(I)
>30 NEXT I
>40 END

```

Após a ordem RUN teremos:

```

>RUN
ABS(-10) = 10
ABS(-5) = 5
ABS(0) = 0
ABS(5) = 5
ABS(10) = 10

```

XII.2.4.c Função Trigonométrica Inversa – ATN(X)

Além das funções trigonométricas, o TK-2000 COLOR permite o uso da função inversa a tangente de X através da 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

```
>RUN
ATN(-10) = -1.47112768
ATN(-5) = -1.37340077
ATN(0) = 0
ATN(5) = 1.37340077
ATN(10) = 1.47112768
```

Outras funções trigonométricas devem ser obtidas pelas fórmulas adequadas como consta no Apêndice A.

XII.2.4.d. Reconhecimento do sinal - SGN(X)

O programa a seguir é um exemplo de SGN(X). Se o valor de X for negativo, o computador apresenta na tela -1; se o valor for 0 ele apresenta 0; se o valor for positivo, apresenta 1.

```
>NEW
>10 FOR I = -10 TO 10 STEP 5
>20 PRINT "SGN(";I;") = ";SGN(I)
>30 NEXT I
>40 END
```

Teremos então:

```
>RUN
SGN(-10) = -1
SGN(-5) = -1
SGN(0) = 0
SGN(5) = 1
SGN(10) = 1
```

XII.2.4.e Função Exponencial - EXP(X)

O uso da matemática em áreas como a engenharia e a física requer a utilização de expoentes naturais que tem e (aprox. 2.718) como base. Se e elevado a X é igual a Y, então podemos usar a seguinte fórmula para o cálculo de Y:

$$Y = e ^ X$$

Na linguagem BASIC do TK-2000 COLOR, podemos expressar isto por $Y = \text{EXP}(X)$. Execute o programa a seguir e confira seu resultado.

```
>NEW
>10 FOR I = 1 TO 5
>20 PRINT "EXP(";I;") = ";EXP(I)
>30 NEXT I
>40 END
```

Resultado

```
>RUN
EXP(1) = 2.71828183
EXP(2) = 7.3890561
EXP(3) = 20.0855369
EXP(4) = 54.5981501
EXP(5) = 148.413159
```

XII.2.4.f Função Logarítmica - LOG(X)

A função logarítmica pode ser obtida no TK-2000 COLOR através da instrução:

LOG(X)

Veja o exemplo:

```
>NEW
>10 FOR I = 1 TO 5
>20 PRINT "LOG(";I;") = ";LOG(I)
>30 NEXT I
>40 END
```

Após RUN teremos:

```
>RUN
LOG(1) = 4
LOG(2) = .693147181
LOG(3) = 1.09861229
LOG(4) = 1.38629436
LOG(5) = 1.604943791
```

XII.2.4.g. Raiz Quadrada - SQR(X)

Esta função já foi apresentada no capítulo III, e tem por finalidade extrair a raiz quadrada de um número. Execute o programa a seguir e confira você mesmo 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
```

FAÇA O SEU RESUMO

1. Nas operações trigonométricas o TK-2000 COLOR utiliza como unidade da medida de ângulos o

2. 360 equivalem aproximadamente a 6.28, desta forma, para sabermos o valor de um ângulo de A (A graus) em podemos utilizar a fórmula:

$$\text{Xrd} = 6.28 \times A / 360 \quad \text{ou} \quad \text{XKrd} = a / 57.29578$$

3. As instruções BASIC, para cálculo do seno, cosseno e tangente são respectivamente:

..... (X)

..... (X)

..... (X)

4. Para calcularmos o cosseno de 45 devemos emitir a instrução: COS(.....)

5. Os valores do seno e cosseno de qualquer ângulo estão compreendidos entre e

6. A instrução RND(.....) fornece valores aleatórios entre 0 e 0.999999999, enquanto RND(.....) repete o número pela instrução RND anterior e RND(.....) seleciona um número aleatório e de valor bastante pequeno.

7. O valor absoluto de um número X é obtido através da instrução (X)

8. A instrução inversa de (X) ou seja, o arco-tangente de um número é obtido através da instrução: (X)

9. Teremos o valor -1, 0 ou +1, dependendo respectivamente se X for negativo, 0 ou positivo, usando-se a função: (X)

10. Tendo o número natural e (cujo valor é 2.718) como base, a operação $Y = e^X$ é obtida no BASIC do TK-2000 COLOR na forma: Y = (X)

11. Para se obter o logaritmo de X utiliza-se a instrução (X)

12. A instrução que fornece a raiz quadrada de um número X é

respostas:

1. radiano; 2. radianos, radianos; 3. SIN, COS, TAN; 4. .785; 5. 1, -1; 6. 1, 0, -X; 7. ABS; 8. TAN, ATN; 9. SGN; 10. EXP; 11. LOG; 12. SQR(X)

CAPÍTULO 13

CAPITULO XIII. MANIPULAÇÃO DE FIGURAS EM ALTA RESOLUÇÃO

XIII.1. INTRODUÇÃO

As operações a seguir são um pouco mais trabalhosas e complexas, porém os resultados compensam tais dificuldades.

O TK-2000 COLOR tem cinco comandos especiais, que permitem manipular figuras no modo gráfico de alta resolução. São eles: DRAW, XDRAW, SCALE, ROT e SHLOAD. Antes que estes comandos possam ser usados, um traçado deve ser definido por uma "definição de figuras", que é constituída por uma sequência de vetores armazenados em uma série de bytes da memória do TK-2000 COLOR. Uma ou mais destas definições de figuras, cujo conjunto forma uma "tabela de figuras", podem ser criadas através do teclado e armazenadas na fita cassette para uso futuro.

Cada byte de definição de figura é dividido em três setores, e cada setor pode especificar um vetor, podendo ou não demarcar um ponto, e também uma direção de movimento (para cima, para 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 contém somente zeros é alcançado, a definição da figura é finalizada.

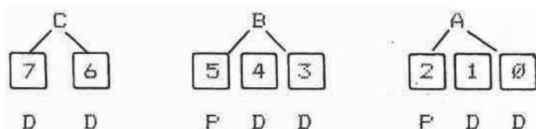
Abaixo pode-se ver como os três setores A, B e C são distribuídos num byte que forma uma definição de figura:

setor:

Número do bit no byte

Especificação

fig. XIII.1.



Cada par de bits DD especifica uma direção de movimento. e cada bit P especifica se o ponto deve ou não ser demarcado antes do movimento, conforme indica a tabela a seguir:

| Bits | Valor | ação |
|------|-------|----------------------|
| DD | 00 | move para cima |
| DD | 01 | move para a direita |
| DD | 10 | move para baixo |
| DD | 11 | move para a esquerda |
| P | 0 | não demarca |
| P | 1 | demarca |

Tabela XIII.1.

Note que o último setor C (os dois bits mais significativos), não tem bit do tipo P. Para este setor P é

considerado como igual a zero, de modo que o setor C pode especificar somente movimento sem pontos demarcados.

DRAW e XDRAW processam os setores da direita para a esquerda (A, B e então C). Qualquer setor do byte que contiver somente zeros & ignorado. Desta forma o byte não pode acabar com um movimento no setor C, que corresponda a 00 (para cima), pois este setor conterá apenas zeros: e neste caso, B não pode fazer o movimento 000, que também será ignorado e o movimento 000 no setor A que finalizará então 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 XIII.2

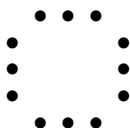


fig.XIII.2.

Primeiramente desenhe esta forma em papel quadriculado, um ponto por quadrado. Em seguida decida por onde começar o desenho da figura. Vamos começar esta pelo centro. Desenhe então uma linha passando por cada ponto do desenho, usando apenas ângulos de 90 para as mudanças de direção, como mostra figura abaixo:

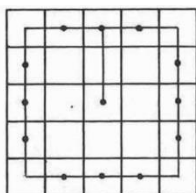


fig. XIII.3.

Redesenhe a figura com uma série de vetores, cada um com um movimento, para baixo, para cima, esquerda ou direita e destaque os vetores que, antes de se mover devem marcar um traço.

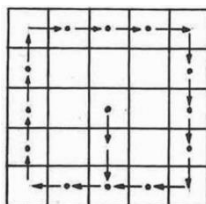
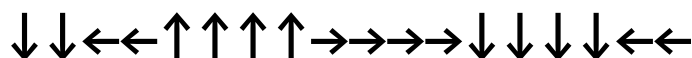


fig. XIII.4.

Agora junte os vetores numa sequência:



Execute como mostra a tabela XIII.2

| byte | setor | C | B | A | Vetores |
|------|-------|---|-----|-----|----------|
| 0 | 01 | | 010 | 010 | ↓↓ |
| 1 | | | 111 | 111 | ←← |
| 2 | | | 100 | 000 | ↑↑ |
| 3 | | | 100 | 100 | →↑↑ |
| 4 | | | 101 | 101 | →→ |
| 5 | | | 010 | 101 | ↓→ |
| 6 | | | 110 | 110 | ↓↓ |
| 7 | | | 011 | 110 | ←↓ |
| 8 | 00 | | | 111 | ← |
| 9 | | | 000 | 000 | FINALIZA |

TABELA XIII.2

este valor não pode
plotar ou mover para
cima

| VETOR | CODIGO | |
|-------|-----------|------------|
| ↑ | 000 | ┐ |
| → | 001 ou 01 | só |
| ↓ | 010 ou 10 | movimentam |
| ← | 011 ou 11 | └ |
| ↑ | 100 | ┐ |
| → | 101 | plotam e |
| ↓ | 110 | movimentam |
| ← | 111 | └ |

Para introduzir os códigos na tabela anterior basta que, de acordo com o vetor desejado, se consulte o respectivo código na tabela XIII.1. preenchendo os setores, da direita para a esquerda e, quando possível, preenchendo o setor C.

Agora forme uma nova tabela como a tabela XIII.3

| seção | C | B | A | Hexadecimal |
|--------|----|-----|-----|-------------|
| byte 0 | 00 | 010 | 010 | 12 |
| 1 | 00 | 111 | 111 | 3F |
| 2 | 00 | 100 | 000 | 20 |
| 3 | 01 | 100 | 100 | 64 |
| 4 | 00 | 101 | 101 | 2D |
| 5 | 01 | 010 | 101 | 15 |
| 6 | 00 | 110 | 110 | 36 |
| 7 | 00 | 011 | 110 | 1E |
| 8 | 00 | 000 | 111 | 07 |
| 9 | 00 | 000 | 000 | 00 |

Tabela XIII.3.

Os valores hexadecimais, são obtidos aplicando cada grupo de 4 bits de um byte na tabela XIII.4.

| CODIGO BINARIO | CODIGO HEX |
|----------------|------------|
| 0000 | 0 |
| 0001 | 1 |
| 0010 | 2 |
| 0011 | 3 |
| 0100 | 4 |
| 0101 | 5 |
| 0110 | 6 |
| 0111 | 7 |

| CODIGO BINARIO | CODIGO HEX |
|----------------|------------|
| 1000 | 8 |
| 1001 | 9 |
| 1010 | A |
| 1011 | B |
| 1100 | C |
| 1101 | D |
| 1110 | E |
| 1111 | F |

Tabela XIII.4.

O conjunto de bytes a que chegamos, apresentado na tabela XIII.Z. é a definição da figura. Há ainda alguns parâmetros a mais que é necessário que se forneça para que a tabela de figuras fique completa. A localização da tabela, cujo endereço genérico será denominado S, deve conter o número de definições de figuras (de 0 a 255), em hexadecimal. No nosso caso o número é um. Em seguida devemos incluir os índices (em hexadecimal) que exprimem quantos bytes após o inicial começa a definição de cada figura. Nos colocaremos a nossa definição de figura imediatamente abaixo de seu índice (uma vez que só há uma definição de figura). Isto significa que neste caso, a definição será iniciada em S + 4. à figura a seguir conterá o formato geral da tabela é o seu formato no caso do exemplo apresentado.

Endereço inicial = S

| | | |
|----------------------|-------|--------------------------|
| Endereço inicial = S | | |
| Bytes | S+0 | n (0 a FF) |
| | +1 | Não usado |
| | +2 | 2 dígitos de menor valor |
| | +3 | 2 dígitos de maior valor |
| | +4 | 2 dígitos de menor valor |
| | +5 | 2 dígitos de maior valor |
| | | |
| | +2n | 2 dígitos de menor valor |
| | +2n+1 | 2 dígitos de maior valor |

← Número total de definições de figuras

D1: índice do 1 byte de definição da figura 1, relativamente a S

Dn: índice do 1 byte de definição da figura n, relativamente a S

| | | | |
|------|------------------|---|------------------------|
| S+D1 | 1 Byte | } | Definição da figura #1 |
| | : | | |
| | Último byte = 00 | | |
| | : | | |
| S+D2 | 1 Byte | } | Definição da figura #2 |
| | : | | |
| | Último byte = 00 | | |
| | : | | |
| S+Dn | 1 Byte | } | Definição da figura #n |
| | : | | |
| | Último byte = 00 | | |
| | : | | |

fig. XIII.5. Formato Geral da Tabela de Definições de Figuras

| | | | |
|------|---|----|---|
| Byte | 0 | 01 | → Número de Figuras |
| | 1 | 00 | → Não usado |
| | 2 | 04 | } Índice da definição de figura #1 |
| | 3 | 00 | |
| | 4 | 12 | |
| | 5 | 3F | |
| | 6 | 20 | } Definição da Figura #1 |
| | 7 | 64 | |
| | 8 | 2D | |
| | 9 | 15 | |
| | A | 36 | } Byte de finalização de definição de figura. |
| | B | 1E | |
| | C | 07 | |
| | D | 00 | |

fig. XIII.6. Tabela de finalização da definição da figura de exemplo.

XIII.1.1. MODO MONITOR

Para registrarmos dados apresentados na figura XIII.6. devemos entrar no MODO MONITOR do TK-2000 COLOR. Dentro deste modo, o TK-2000 COLOR deixa de estar sob controle do interpretador BASIC e passa a ser controlado pelo programa monitor propriamente dito.

A mudança para o MODO MONITOR é obtido através da instrução:

```
>CALL -159          (RETURN)
ou
>LM                 (RETURN)
```

sabemos que o TK-2000 COLOR entrou no modo monitor quando o sinal ">" no início de uma linha de comando é substituído por @.

XIII.1.2. Carga da Tabela de Figuras.

Agora você está pronto para digitar sua tabela de figuras na memória do TK-2000 COLOR. Primeiro escolha um endereço inicial. Para o prosseguimento de nosso exemplo vamos selecionar 1DFC.

OBSERVAÇÃO: 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 1ª página de memória utilizada para definição de gráficos de alta resolução.

Digite agora o endereço inicial da tabela de figuras.

```
@1DFC
```

Se você pressionar a tecla RETURN, o TK-2000 COLOR irá mostrar o conteúdo do endereço especificado. Agora execute o seguinte.

```
@1DFC: 01 RETURN
@1DFC   RETURN
1DFC- 01
```

O TK-2000 COLOR diz que o valor 01 (hexadecimal) está armazenado na localização cujo endereço é 1DFC. Para armazenar mais do que dois dígitos de números hexadecimais em bytes sucessivos de memória, basta entrar com o primeiro endereço (1DFC) e então com os valores desejados, separados por espaço. Entremos agora a nossa tabela.

```
@1DFC: 01 00 04 00 12 3F 20 64 2D 15 36
1E 07 00 RETURN
```

Para conferir as informações da sua tabela de figuras, você pode examinar cada byte separadamente ou simplesmente pressionar

repetidas vezes a tecla RETURN até que todos os bytes que interessam (e provavelmente alguns a mais) sejam apresentados.

```
@1DFC RETURN
1DFC- 01
@      RETURN
00 04 00
@      RETURN
1E00- 12 3F 20 64 2D 15 36 1E
@      RETURN
1E08- 07 00 FF FF 78 78 FF FF
```

estes bytes não interessam

XIII.1.3. Indicando ao BASIC o local da Tabela de Figuras

Se a sua tabela de figuras está correta, tudo 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 cassette pelo comando SHLOAD, conforme será 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
```

XIII.1.4. Proteção da tabela de Figuras

A proteção da sua tabela de figuras de uma alteração accidental pelo programa BASIC, é conveniente. Com este propósito, alteraremos HIMEM: (nas posições 73 e 74 hexa), digitando:

```
@73: FC 1D
```

Isto também é feito automaticamente pelo uso de SHLOAD, indicando no próximo sub-item.

XIII.1.5. Armazenando uma Tabela de Figuras e SHLOAD

Para armazenar sua Tabela de Figura, você precisa conhecer três coisas:

1. O endereço inicial da Tabela (1DFC, no nosso exemplo)
2. O último endereço da Tabela (1E09, no nosso exemplo)
3. A diferença entre os endereços final e inicial (000D, no nosso exemplo)

Estes dados devem ser fornecidos em hexadecimal e, no caso do nosso exemplo, equivalem a 1DFC, 1E09 e 000D.

O terceiro dado (diferença entre os endereços final e inicial), deve ser armazenado nos endereços 0 (os dois dígitos menos significativos) e 1 v(os dois dígitos mais significativos):

```
@0: 0D 00 RETURN
```

Agora voce pode "escrever" (armazenar no cassette), o tamanho e o endereço inicial e final da Tabela:

```
@0.1WA 1DFC.1E09WA RETURN
```

Não pressione RETURN antes que você tenha colocado a fita cassette no gravador, rebobinando-a e pressionado os controles de gravação (provavelmente as teclas PLAY e RECORD do gravador, simultaneamente). Pressione então a tecla RETURN do TK-2000 COLOR.

Para ler a fita, rebobine-a, pressione a tecla PLAY com o TK-2000 COLOR sob o controle do BASIC digite:

```
>SHLOAD RETURN
```

Você deverá ouvir um sinal sonoro (beep) quando o número de bytes da tabela for lido com sucesso e um segundo sinal quando for concluída a leitura da Tabela.

XIII.1.6. Saindo do Modo Monitor

Para sair do modo monitor basta manter a tecla CONTROL pressionada, digitar a tecla C e em seguida (após soltar ambas as teclas) pressionar RETURN.

XIII.2 USANDO A TABELA DE FIGURAS

Podemos agora escrever um programa em BASIC usando a tabela de figuras e os comandos apropriados, que serão vistos a seguir.

O programa abaixo & bastante simples e define uma figura na tela, executa uma rotação de 16 graus e então repete a figura apresentada. Cada nova figura terá tamanho maior que a anterior. Execute-o com RUN.

```
>100 HGR
>110 HCOLOR = 3
>120 FOR R = 1 TO 140
>130 ROT = R
>140 SCALE = R
>150 DRAW 1 AT 139,90
>160 NEXT R
```


Para ver, inicialmente, um pequeno quadrado, adicione a linha:

```
>155 END
```

Pode-se então adicionar uma pausa e apagar cada quadrado após sua apresentação adicionando-se:

```
>153 FOR 1 = 0 TO 1000 : NEXT I  
>155 XDRAW 1 AT 139,90
```

Execute o comando RUN e observe agora o desenvolvimento do programa.

XIII.2.1. DRAW

```
DRAW A AT X,Y
```

E desenhada uma figura no modo gráfico de alta resolução, começando no ponto cujas coordenadas são: X,Y ;. A figura é a A-ésima da tabela de figuras, carregada previamente pelo comando SHLOAD ou digitada no modo monitor do TK-2000 COLOR.

O valor A deve estar entre Y em, onde n é o número (entre y e 255) da definição de figura indicada no byte e da tabela de figuras. X deve ter valores entre b e 278 e Y entre 0 e 191. Qualquer valor que exceda os limites apresentados causará a mensagem de erro.

```
?VALOR ILEGAL #ERRO
```

A cor, rotação e escala da figura a ser apresentada devem ser indicadas antes que DRAW seja executado.

Quando o comando DRAW é emitido sem a existência de uma tabela de figuras, pode acontecer que o sistema não seja recuperado. Deve-se então pressionar as teclas RESET ou CONTROL C RETURN. Pode também acontecer que sejam criadas figuras aleatórias em toda área de memória para gráficos de alta resolução, possivelmente destruindo o seu programa.

XIII.2.2. XDRAW

Este comando é similar a DRAW, exceto que a cor usada para desenhar a figura & a complementar daquela já existente nos pontos determinados. Desta forma, através de XDRAW pode-se suprimir da tela um traço já existente, sem alterar a cor de fundo da tela. Seu formato é o mesmo que o da instrução DRAW:

```
XDRAW A AT X,Y
```

XIII.2.3. ROT e SCALE

O formato da primeira instrução é:

ROT = X

Ela 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 y a 255.

ROT=0, irá especificar uma rotação nulas ROT=16 faz com que a figura definida por DRAW execute uma rotação de 90 graus no sentido do relógio (horário); ROT = 32 causará uma rotação de 180 graus no sentido do relógio, e assim por diante. À repetição do processo ocorre com ROT=64.

Para SCALE=1, apenas 4 valores de rotação são recomendados (0, 16, 32 e 48); Para SCALE=2, são recomendados 8 rotações, e assim por diante. Os valores de rotação não reconhecidos não causarão (geralmente) a definição da figura orientada de acordo com o primeiro valor de rotação definido.

A instrução SCALE determina o tamanho da figura a ser traçada por DRAW ou XDRAW, a partir do valor 1 (reproduzindo, ponto a ponto, a definição da figura) até 255 (quando cada vetor é ampliado 255 vezes).

Importante: SCALE=0 é à máximo tamanho e não apenas um ponto.

A forma geral desta instrução é:

SCALE = X

onde X determina a escala da figura.

Para encerrar o assunto, por enquanto, é apresentado abaixo um outro programa com a mesma tabela de figura usada no exemplo anterior:

```
>NEW
>10 HGR
>20 J = INT( RND(I) * 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
```

Note que agora surgem duas figuras iguais e que a cor varia randomicamente de uma execução para outra.

XIII.2.4. SHLOAD

Como já vimos o comando SHLOAD carrega uma tabela da fita cassette. Esta tabela é carregada exatamente abaixo de HIMEM:, e HIMEM: é fixado depois, abaixo da tabela para protege-la. Na hora da leitura o endereço inicial da tabela é fixado automaticamente. Se uma segunda tabela é carregada, substituindo a primeira tabela, HIMEM: deve ser resetado antes do SHLOAD para evitar perda de memória.

Só o comando RESET pode deter o SHLOAD.

FAÇA O SEU RESUMO

1. A definição de figuras é constituída por um conjunto de Por sua vez, um conjunto de definições de figuras forma uma
2. O primeiro endereço da deve conter o de definições de figuras. Em seguida deve-se incluir os que determinam quantos bytes após o byte inicial começa a definição de cada uma das figura.
3. Para entrar no modo monitor do TK-2000 COLOR devemos emitir o comando -159 ou
4. No modo monitor, para sabermos o conteúdo de determinado endereço basta digitarmos o endereço desejado e a tecla RETURN. Se no entanto quisermos carregar um valor hexadecimal num determinado endereço, deveremos digitar:
<endereço><.....><valor hexadecimal> RETURN
5. Para a carga de vários valores em endereços consecutivos, basta dar o endereço inicial, conforme se vê na questão anterior, em seguida digitar cada valor separados por um e só no final da carga pressionar RETURN.
6. O endereço inicial da deve ser carregado através da ordem :
<.....><.....><2 byte do end.><.....><1 byte do end.>
Esta ordem é executada automaticamente através do comando SHLOAD.
7. Para se proteger a de uma alteração acidental, carrega-se seu endereço inicial em 73H, através da ordem:
<.....><.....><2 byte do end.><.....><1 byte do end.>
Esta ordem também é executada automaticamente por SHLDAD.
8. Para se armazenar a em fita cassette, deve-se conhecer seus endereços , é a diferença entre eles.
9. À instrução
..... A AT X,Y
permite traçar qualquer uma das figuras definidas pela
..... , sendo que (X,Y) indica o ponto do traçado. Se for desejado o traçado da segunda definição de figura da
..... o valor de A deverá ser
10. A instrução é similar a porém utiliza a cor para o traçado de uma figura. Ela pode ser usada para um traçado, sem alterar outras áreas da tela.
11. = 16, define uma rotação de 90 graus na figura a per

traçada por ou

12. Determina-se o tamanho da figura a ser traçada por
ou usa através da instrução = 1 reproduz o
traçado ponto a ponto e = 255 amplia 255 vezes os
vetores que compõe a figura.

13. Para carregar no TK-2000 COLOR uma
contida em fita cassette, após a correta preparação do gráfico,
deve-se executar o comando

respostas: 1. vetores, tabela de figuras; 2. tabela de figuras,
número, índices; 3. CALL, LM; 4. : ; 5. espaço; 6. tabela de
figuras, E8, : , espaço; 7. tabela de figuras, 73, : , espaço; 8.
tabela de figuras, inicial, final; 9. DRAW, tabela de figuras,
inicial, tabela de figuras, 2; 10. XDRAW, DRAW, complementar,
suprimir; 11. ROT, DRAW, XDRAW; 12. DRAW, XDRAW, SCALE, SCALE,
SCALE; 13. tabela de figuras, SHLOAD

CAPÍTULO 14

CAPITULO XIV - USO DE ROTINAS EM LINGUAGEM DE MAQUINA

Seria pretencioso querer esgotar em uma única publicação um assunto tão vasto e profundo como é a programação deste computador. Desta forma, devemos traçar nossos próprios limites. No capítulo anterior já tivemos alguns conceitos um pouco mais complexos, cuja assimilação se tornou um pouco mais difícil para pessoas inexperientes. O uso da linguagem de máquina permite uma maior flexibilidade, melhor aproveitamento da memória e facilidades e programação no TK-2000 COLOR. Este assunto porém é bastante extenso e apresenta uma certa complexidade, de forma que deverá ser abordado por uma nova publicação. Neste capítulo, descreveremos apenas, de forma sintética as instruções EASIC relacionadas ao uso da linguagem de máquina, permitindo que programadores mais experientes possam utilizá-las.

Uma vez que este capítulo não tem um teor prático propriamente dito, ele não será acompanhado do item "FAÇA O SEU RESUMO".

As instruções PEEK e POKE pertencem a este grupo, e já foram apresentadas.

XIV.1. CALL

Esta instrução causa 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 após a execução da rotina, o programa volte automaticamente para a próxima ordem após CALL, a última instrução da rotina deverá ser RETURN.

No capítulo anterior, já utilizamos uma vez esta instrução para entrar no modo monitor (CALL. -159). Outro exemplo da utilização desta instrução é CALL-936, cujo efeito é exatamente o mesmo do comando HOME.

XIV.2. WAIT

A forma geral desta instrução é:

WAIT X,Y,Z

Ela é usada para inserir uma pausa condicional no programa. X é um endereço de memória e deve estar entre os limites -65535 a 65535, fora dos quais provocará uma mensagem de erro do tipo:

?VALOR ILEGAL #ERRO

Y e Z devem estar entre os limites 0 a 255 decimal. Quando WAIT é executado, estes valores são convertidos para binário entre 0 e 11111111.

Se apenas Y for especificado, 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 forem oito zeros, o teste é repetido. Se algum dos resultados for diferente de zero, isto é, se houver pelo menos um bit igual a 1 nos dois bytes, o WAIT é completado e o programa prossegue na próxima instrução.

Se os três parâmetros forem especificados, WAIT executará 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, é executado um AND entre cada bit resultante da operação citada e do byte Y. Se o resultado deste conjunto de operações for oito zeros, o teste é repetido. Caso contrário o programa prossegue na próxima ordem.

XIV.3. USR(X)

Esta instrução permite a transferência do valor X para o acumulador ponto flutuante (endereços \$9D a \$A3) e efetuado um desvio ao endereço \$0A. Ai deve existir uma rotina ou JMP adequados. O valor de retorno para a função e o conteúdo do acumulador de ponto flutuante.

XIV.4. HIMEM:

Permite determinar a máxima posição de memória disponível para o programa BASIC, incluindo as variáveis. E usada para proteger a área de memória acima do valor determinado. À 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 -65535 a 65535, fora do que o TK-2000 COLOR emitirá uma mensagem de erro do tipo:

?VALOR ILEGAL #ERRO

Veja APENDICE F

XIV.5. LOMEM:

Semelhante á instrução anterior, define o' menor

endereço disponível para variáveis. Normalmente, o TK-2000 COLOR define automaticamente LOMEM como o endereço após a área de programa.

Forma geral da instrução:

LOMEM: X

onde X é o endereço desejado.

Veja APENDICE F

CAPÍTULO 15

CAPITULO XV- O COMANDO SOUND

Uma característica de destaque no seu TK-2000 COLOR, é a possibilidade que ele tem de, através do alto-falante da televisão, ao qual & conectado, produzir sons. Estes sons permitem, por exemplo, tornar os programas de jogos especialmente interessantes, e próximos a realidade, pela imitação dos barulhos de bombas, metralhadoras laser etc..

Pode-se também programar músicas no TK-2000 COLOR, uma vez que ele permite a utilização de até quatro oitavas com tons e semitons, 5 tempos diferentes para cada nota.

Todos estes recursos são obtidos através do comando SOUND, descrito a seguir.

NOTA: Lembre-se que o volume com que o som será emitido dependerá do controle de volume do seu TV.

XV.1. Comando SOUND

O comando SOUND é o que especificamente controla a tonalidade e a duração da nota gerada do alto falante do televisor. O formato deste comando é:

SOUND X,Y

onde X representa a nota a ser tocada e Y a duração desta. Para testar a utilização deste comando pode-se digitar a seguinte instrução abaixo em modo imediato:

>SOUND 192,240

após pressionar RETURN o TK-2000 COLOR emitirá uma nota DO com uma duração inteira, como demonstra o pentagrama abaixo:



Caso você queira emitir um MI com duração meia nota basta digitar:

>SOUND 154,120

XV.1.1. SOUND X1,Y1 TO X2,Y2

Esta modalidade de SOUND permite tocar em uma mesma linha varias notas com suas respectivas durações. Este comando agora oferece um maior desempenho que o anterior na composição de músicas ou acordes num pequeno número de linhas. Por exemplo, digite o seguinte comando:

>SOUND 96,120 TO 85,120 TO 76,240

o som emitido neste caso é a sequência das notas DO RE MI em uma escala mais aguda.



XV.1.2. Pausa

Como frequentemente nas músicas existe a pausa, no TK-2000 COLOR há um dispositivo que provoca um intervalo na continuidade da música.

Neste caso usa-se o comando SOUND sendo que no espaço do tom coloca-se o número 1.

Por exemplo:

>SOUND 96,120 TO 1,240 TO 76,240

XV.2. EXECUTANDO MUSICAS

Uma vez que possuímos em mãos as tabelas XV.1 e XV.2 podemos escrever ou transportar de um pentagrama diversas músicas. Para isso teremos que saber interpretar uma linha musical.

Para representar os sons são usados sinais de forma oval que, pelas posições tomadas no pentagrama indicam os sons mais graves ou mais agudos, designando-se o tempo de prolongação pela variedade de sua forma.



Quando a nota vier precedida de uma letra "b" isto significa que a nota deve ser tocada em bemol (nota intermediária com a anterior) e ao encontrar o sinal de "#" sustenido, ou seja a nota deverá ser tocada no tom intermediário com a nota seguinte. Na tabela XV.1 estes tons também foram representados com um "b" ou va respectivamente.

Para representar os tempos de duração da nota já a tabela XV.2 permite identificar cada símbolo pelo seu tempo de duração bastando somente explicar que as notas unidas irão ser representadas no TK-2000 COLOR como notas separadas.

Já temos agora condições de interpretar uma linha de música e após você rodar o exemplo já estará em condições de executar programas musicais no TK-2000 COLOR.

A seguir são apresentadas as tabelas comparativas de tonalidade e duração das notas:

| T O N A L I D A D E | | | | | | | | | |
|---------------------|-----------|----------|----------|----------|--|--|--|--|--|
| SOL - 255 | SOL - 128 | SOL - 64 | SOL - 31 | SOL - 15 | | | | | |
| LAB - 243 | LAB - 121 | LAB - 60 | LAB - 29 | LAB - 14 | | | | | |
| LA - 231 | LA - 114 | LA - 56 | LA - 28 | LA - 13 | | | | | |
| SIb - 217 | SIb - 108 | SIb - 53 | SIb - 26 | SIb - 42 | | | | | |
| SI - 203 | SI - 102 | SI - 50 | SI - 25 | | | | | | |
| DO - 192 | DO - 96 | DO - 47 | DO - 23 | | | | | | |
| DO# - 182 | DO# - 90 | DO# - 45 | DO# - 22 | | | | | | |
| RE - 172 | RE - 85 | RE - 42 | RE - 21 | | | | | | |
| MIb - 162 | MIb - 80 | MIb - 40 | MIb - 20 | | | | | | |
| MI - 154 | MI - 76 | MI - 37 | MI - 18 | MI - 11 | | | | | |
| FA - 146 | FA - 72 | FA - 35 | FA - 17 | FA - 10 | | | | | |
| FA# - 137 | FA# - 67 | FA# - 33 | FA# - 16 | FA# - 9 | | | | | |

Tabela XV.1

A faixa compreendida entre o 98 e 47 corresponde a escala 4 do piano.







| D U R A Ç Ã O | | |
|---------------|-----|---|
| Minima | 240 |  |
| Seminima | 120 |  |
| Colcheia | 60 |  |
| Semicolcheia | 30 |  |
| Fusa | 15 |  |
| Semifusa | 8 |  |

Tabela XV.2

São apresentados três músicas já incluídas e com suas traduções para o comando SOUND notando inicialmente que todas as notas e durações foram diretamente introduzidas em comandos DATA que o programa se encarregara de transportar para o comando SOUND, sendo que esta operação torna o programa BASIC mais simples e mais fácil de ser digitado. Os tempos de duração também foram reduzidos e divididos, já no comando DATA pelo seu máximo divisor comum sendo

esta constante, então, multiplicada na mesma linha do comando SOUND. Caso você queira tornar mais rápida a música, deve ser diminuído o fator de multiplicação de B(I) no comando SOUND. Para melhor entender estes detalhes observe os exemplos:

EXEMPLO 1

PARABENS P'RA VOCE

```
>NEW
>10 DIM A(28), B(28) : REM SENDO A MATRIZ A AS NOTAS E B AS
    DURACOES.
>20 DATA 96, 96, 85, 96, 72, 76, 1, 96, 96, 85, 96, 64, 72, 72, 56,
    56, 47, 56, 72, 76, 85, 53, 53, 56, 72, 64, 72, 72
>30 DATA 4, 4, 8, 8, 8, 8, 4, 4, 4, 8, 8, 8, 8, 8, 4, 4, 8, 8, 8,
    8, 8, 4, 4, 8, 8, 8, 8, 8
>40 FOR I = 1 TO 28 : REM LEITURA DA MATRIZ DAS TONALIDADES
>50 READ A(I)
>60 NEXT I
>70 FOR I = 1 TO 28 : REM LEITURA DA MATRIZ DAS DURACOES
>80 READ B(I)
>90 NEXT I
>100 FOR I = 1 TO 28 : REM EXECUCAO DA MUSICA
>110 SOUND A(I),B(I)*30
>120 NEXT I
>130 END
```

Exemplo 2 : ATIREI UM PAU NO GATO

```
>NEW
>10 DIM A(40), B(40)
>20 DATA 64, 72, 76, 85, 76, 72, 64, 64, 64, 56, 64, 72, 72, 72,
    64, 72, 76, 76, 76, 96, 96, 56, 56, 56, 50, 56, 64, 64,
    64, 76, 70, 64, 76, 72, 64, 72, 76, 85, 96, 47
>30 DATA 4, 4, 4, 4, 4, 4, 8, 8, 8, 4, 4, 8, 8, 8, 4, 4, 8, 8, 8,
    4, 4, 8, 8, 8, 4, 4, 8, 8, 8, 4, 4, 8, 4, 4, 4, 4, 4, 4,
    8, 8
>40 FOR I = 1 TO 40
>50 READ A(I)
>60 NEXT I
>70 FOR I = 1 TO 40
>80 READ B(I)
>90 NEXT I
>100 FOR I = 1 TO 40
>110 SOUND A(I),B(I)*30
>120 NEXT I
>130 END
```

Exemplo 3 A TIME FOR US

```
>NEW
>10 DIM A(100),B(100)
>20 DATA 64, 53, 56, 85, 85, 72, 85, 64, 64, 72, 80, 72, 72, 80,
85, 96, 85, 96, 108, 96, 85, 64, 53, 56, 85, 85, 72, 85,
64, 64, 47, 56, 85, 53, 56, 64, 72, 53, 56, 64, 72, 64, 53
>24 DATA 42, 53, 47, 42, 40, 47, 42, 53, 47, 60, 53, 64, 56, 72,
64, 64, 53, 56, 85, 85, 72, 85, 64, 64, 72, 80, 72, 72,
80, 85, 96, 85, 96, 108, 96, 85, 64, 53, 56, 85, 85, 72,
85, 64, 64, 47, 56, 85, 53, 56, 64, 72, 53, 56, 64, 72, 64
>30 DATA 1, 1, 1, 3, 1, 1, 1, 3, 1, 1, 1, 3, 1, 1, 1, 3, 0.5, 0.5,
1, 1, 3, 1, 1, 1, 3, 1, 1, 1, 5, 1, 6, 5, 1, 3, 1, 1, 1,
1, 1, 2, 2, 5, 1, 5, 1, 5, 1, 5, 1, 5, 1, 5, 1, 5, 1
>35 DATA 5, 1, 3, 1, 1, 1, 3, 1, 1, 1, 3, 1, 1, 1, 3, 1, 1, 1, 3,
0.5, 0.5, 1, 1, 1, 3, 1, 1, 1, 3, 1, 1, 1, 5, 1, 6, 5, 1, 3,
1, 1, 1, 1, 1, 2, 2, 6
>40 FOR I = 1 TO 100
>50 READ A(I)
>60 NEXT I
>70 FOR I = 1 TO 100
>80 READ B(I)
>90 NEXT I
>100 FOR I = 1 TO 100
>110 SOUND A(I),B(I)*42.5
>120 NEXT I
>130 END
```

XV.3. EFEITOS SONOROS EM LINGUAGEM DE MAQUINA

Além da possibilidade do comando SOUND em BASIC para gerar um tom, é possível acessar o canal de som a nível de linguagem de máquina. O acionamento do canal de som é produzido ao acessar o endereço \$C030 no modo monitor ou PEEK(-16336) ou PEEK (49200) em BASIC.

Cada acesso a este endereço efetua uma transição de sinal. Repetido isto a certa frequência, é gerado um som cuja frequência corresponde a metade da frequência de acesso.

Desta forma é possível maior versatilidade nos tipos de sons a serem gerados, especialmente quando em rotina em linguagem de máquina.

Em continuação são apresentados uma série de efeitos sonoros que você pode utilizar em programas.

Para entrar este programa você deve entrar em modo monitor e inserir nos endereços indicados as informações correspondentes de acordo as instruções apresentadas na seção XIII.1.

Caso você queira inseri-los em outros endereços deverão ser realizadas algumas alterações no conteúdo deste programas.

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

Para executar este comando no modo monitor basta digitar 08B6G ou senão, no Modo BASIC - CALL 2230.

EXPLOÇÃO

1560- A9 07 85 06 A0 00 A9 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 C8 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

Para executar este efeito basta. em modo monitor, digitar 1560G ou, em modo BASIC, CALL 5472.

HELICOPTERO

15AC- A0 23 20 B4
15B0- 15 88 D0 FA A9 01 85 FF
15B8- A9 45 85 FE A9 00 BD 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 BD 30 C0 EE 30 C0 CE
15E8- 30 C0 C6 06 D0 02 C6 07
15F0- D0 F8 C6 08 D0 E3 60

Para executar este efeito basta, em modo Monitor digitar 15ACG 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 F0 03 64 CA D0
14F8- FD 88 F0 03 4C EF 14 56
1500- 09 A9 00 BD 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 00 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 efeito basta, em modo monitor digitar 14D6G e, em modo BASIC, CALL 5334.

Uma vez que os efeitos sonoros apresentados acima apresentam endereços complementares, poderia-se incluir-los todos de uma só vez na memória RAM. Deste modo seria possível, através do programa abaixo, a combinação de todos os efeitos de uma só vez.

```
>10 CALL 5548
>20 CALL 2230
>30 CALL 5334
>40 CALL 5472
>RUN
```

APENDICES
A-B-C-D-
E-F-G-H-I

APENDICE A

FUNÇÕES TRIGONOMETRICAS

As funções trigonométricas apresentadas a seguir, como não fazem parte do BASIC TK-2000 COLOR, podem ser obtidas através das seguintes fórmulas:

SECANTE

$$\text{SEC}(X) = 1/\text{COS}(X)$$

COSECANTE

$$\text{CSC}(X) = 1/\text{SIN}(X)$$

COTANGENTE

$$\text{COT}(X) = 1/\text{TAN}(X)$$

SENO INVERSO

$$\text{ARCSIN}(X) = \text{ATN}(X / \text{SQR}(-X * X + 1))$$

COSSENO INVERSO

$$\text{ARCCOS}(X) = -\text{ATN}(X / \text{SQR}(-X * X + 1)) + 1.5708$$

SECANTE INVERSA

$$\text{ARCSEC}(X) = \text{ATN}(\text{SQR}(X * X - 1)) + (\text{SGN}(X) - 1) * 1.5708$$

COSSECANTE INVERSA

$$\text{ARCCSC}(X) = \text{ATN}(1/\text{SQR}(X * X - 1)) + (\text{SGN}(X) - 1) * 1.5708$$

COTANGENTE INVERSA

$$\text{ARCCOT}(X) = -\text{ATN}(X) + 1.5708$$

SENO HIPERBOLICO

$$\text{SINH}(X) = (\text{EXP}(X) - \text{EXP}(-X)) / 2$$

COSSENO HIPERBOLICO

$$\text{COSH}(X) = (\text{EXP}(X) + \text{EXP}(-X)) / 2$$

TANGENTE HIPERBOLICA

$$\text{TANH}(X) = -\text{EXP}(-X) / (\text{EXP}(X) + \text{EXP}(-X)) * 2 + 1$$

SECANTE HIPERBOLICA

$$\text{SECH}(X) = 2 / (\text{EXP}(X) + \text{EXP}(-X))$$

COSSECANTE HIPERBOLICA

$$\text{CSCH}(X) = 2 / (\text{EXP}(X) - \text{EXP}(-X))$$

COTANGENTE HIPERBOLICA

$$\text{COTH}(X) = \text{EXP}(-X) / (\text{EXP}(X) - \text{EXP}(-X)) * 2 + 1$$

SENO HIPERBOLICO INVERSO

$$\text{ARCSINH}(X) = \text{LOG}(X + \text{SQR}(X * X + 1))$$

COSSENO HIPERBOLICO INVERSO

$\text{ARGCOSH}(X) = \text{LOG}(X + \text{SQR}(X * X - 1))$

TANGENTE HIPERBOLICA INVERSA

$\text{ARGTANH}(X) = \text{LOG}((1 + X) / (1 - X)) / 2$

SECANTE HIPERBOLICA INVERSA

$\text{ARGSECH}(X) = \text{LOG}((\text{SQR}(-X * X + 1) + 1) / X)$

COSSECANTE HIPERBOLICA INVERSA

$\text{ARGCSCH}(X) = \text{LOG}(\text{SGN}(X) * \text{SQR}(X * X + 1) + 1) / X$

COTANGENTE HIPERBOLICA INVERSA

$\text{ARGCOTH}(X) = \text{LOG}((X + 1) / (X - 1)) / 2$

A MOD B

$\text{MOD}(A) = \text{INT}((A / B - \text{INT}(A / B)) * B + .05) * \text{SGN}(A / B)$

APENDICE B

MENSAGENS DE ERRO

Quando ocorre um erro o BASIC interrompe o programa em execução e não é possível executar CONT. Fara se precaver contra isto pode ser usada uma subrotina apropriada através de um ONERR GOTO.

A seguir uma breve explicação acerca dos erros e 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.

SINTAXE

Falta de parênteses, pontuação incorreta, comando digitado errado ou inexistente, caracter ilegal, variável não compatível etc.

RETURN SEM GOSUB

Existe algum comando RETURN sem que houvesse previamente definido um comando GOSUB.

NAO HA MAIS DATA

Um comando READ foi encontrado porém não há mais DATA disponível (todas as variáveis já foram lidas ou inexistente o comando DATA)

VALOR ILEGAL

Poderá ocorrer numa das seguintes condições:

- a) Valor negativo na definição de um array (Ex:LET A(-3) = 0)
- b) LOG com argumento nulo ou negativo
- c) SQR com argumento negativo
- d) Uso de argumento impróprio nos comandos MID\$, LEFT&, RIGHT\$, WAIT, PEEK, POKE, TAB, SPC, ON....GOTO
- e) $A \wedge B$ sendo A negativo e B não inteiro.

ESTOURO

O resultado de uma operação matemática foi maior que a capacidade de BASIC de representá-lo.

EXCEDE MEMORIA

Poderá ocorrer numa das seguintes condições:

- a) Programa muito extenso
- b) Excesso de variáveis
- c) Mais do que 10 loops de FOR.....NEXT
- d) Mais do que 24 loops de GOSUB..... RETURN
- e) Mais do que 36 níveis de cálculo (parênteses).
- f) LOMEM maior que HIMEM

COMANDO NÃO DEFINIDO
Ocorre quando um GOTO ou GOSUB referencia uma linha de programa inexistente.

INDICE ILEGAL
Ocorre quando o índice de um array for maior que DIM especificado. Ex: LET A(1,1,1)=5 ou LET A(1,3)=5, porém foi dimensionado DIM A(2,2).

RE'DIM' DE ARRAY
Ocorre quando um array foi dimensionado mais do que uma vez dentro de um programa.

DIVISÃO FOR ZERO
Qual quer tentativa de divisão por zero acusará erros.

COMANDO ILEGAL.
Os comandos INPUT, DEF FIN, DATA não podem ser usados no modo imediato.

INCOMPATIVEL
Ocorre quando o comando for alfanumérico e a variável associada for numérica ou vice-versa. Ex: LET A\$ = 5

EXCEDE STRING
Ocorre quando for criado um string maior do que 255 caracteres.

FORMULA MUITO COMPLEXA
Ocorre quando houver mais do que 2 comandos seguidos de IF "xx"
Ex: IF "xx" THEN IF "xx" THEN IF "yy" THEN

IMPOSSIVEL
Ocorre quando se deseja continuar um programa inexistente ou que foi interrompido por outro erro.

FUNÇÃO NAO DEFINIDA
Ocorre quando se referencia a uma função não definida anteriormente.

CODIGOS DOS ERROS OBTIDOS ATRAVES DE PEEK(222)

| CODIGO | ERRO |
|---------------|------------------------|
| 0 | NEXT SEM FOR |
| 12 | SINTAXE |
| 20 | RETURN SEM GOSUB |
| 36 | NAO HA MAIS DATA |
| 52 | VALOR ILEGAL |
| 64 | ESTOURO |
| 71 | EXCEDE MEMORIA |
| 85 | COMANDO NAO DEFINIDO |
| 105 | INDICE TLEGAL |
| 118 | RE'DIM' DE ARRAY |
| 134 | DIVISAO FOR ZERO |
| 150 | COMANDO ILEGAL |
| 164 | INCOMPATIVEL |
| 176 | EXCEDE STRING |
| 189 | FORMULA MUITO COMPLEXA |
| 211 | IMPOSSIVEL |
| 221 | FUNCAÃO NAO DEFINIDA |

APENDICE C

ECONOMIA DE ESPAÇO E TEMPO

A seguir serão apresentadas algumas "dicas" referentes a procedimentos que o usuário pode adotar com a finalidade de compactar o uso da memória pelo seu programa.

- 1) Use linhas de múltiplos comandos, de no máximo 239 caracteres. Este procedimento se recomenda após o programa ter sido testado já que a edição pode resultar difícil se for encontrado algum erro.
Ex: 10 FOR 1 = 1 TO 10 : PRINT "TK2000 COLOR" : NEXT
- 2) Apague todos os comandos REM do programa. Recomendase guardar uma versão do programa com os REM para facilitar a documentação.
- 3) Use matrizes de números inteiros se possível, no lugar de números reais.
- 4) Use variáveis no lugar de constantes.
- 5) Não use no fim de um programa o comando END. A utilização deste comando nos exemplos utilizados neste manual tinham caráter puramente didático.
- 6) Reuse, se possível, as mesmas variáveis dentro de um programa, desde que não incorra em conflitos de lógica.
- 7) Procure criar subrotinas para áreas de programa que executarem tarefas similares.
- 8) Use os elementos zero das matrizes, como por exemplo A(0), B(0,Y).
- 9) Quando A\$ = "COL" e redefinido em A\$ = "AP" o antigo string "COL" não é apagado na memória. Usando periodicamente uma instrução do tipo X = FRE(0) dentro do seu programa o BASIC "limpará" antigos conteúdos do topo da memória.

Agora algumas "dicas" para aumentar a velocidade de execução do BASIC:

- 1) Use variáveis no lugar de constantes
- 2) Defina as variáveis mais usadas no seu programa no início do mesmo, já que assim diminui o tempo de procura.
- 3) Use NEXT sem a variável índice, com isso o processador não efetuará a procura.

- 4) Coloque as linhas mais referenciadas do seu programa no início do mesmo.

APENDICE D

CODIGOS DOS COMANDOS

| | | | | | |
|-----|---------|-----|---------|-----|--------|
| 128 | END | 129 | FOR | 139 | NEXT |
| 131 | DATA | 132 | INPUT | 133 | DEL |
| 134 | DIM | 135 | READ | 136 | GR |
| 137 | TEXT | 138 | DSK | 139 | ASS |
| 140 | CALL | 141 | PLOT | 142 | HLIN |
| 143 | VLIN | 144 | HGR2 | 145 | HGR |
| 146 | HCOLOR | 147 | HPLLOT | 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\$ | 235 | LM |
| 236 | MOTOR | 237 | TK2000 | 238 | MP |
| 239 | MA | | | | |

APENDICE E

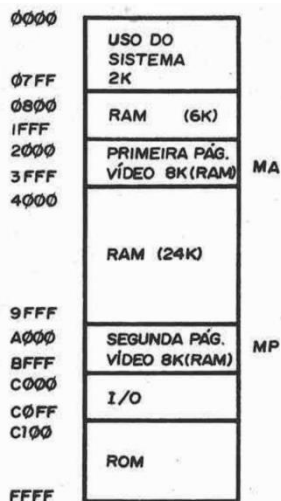
PALAVRAS RESERVADAS

Estas palavras reservadas são palavras que o BASIC reconhece para uso exclusivo.

| | | | | | |
|--------|---------|--------|---------|---------|--------|
| ABS | AND | ASC | ASS | AT | ATN |
| CALL | CHR\$ | CLEAR | COLOR | CONT | COS |
| DATA | DEF | DEL | DIM | DRAW | DSK |
| & | END | EXP | | | |
| FN | FOR | FRE(| | | |
| GET | GOTO | GOSUB | GR | | |
| HCOLOR | HGR | HGRZ | HIMEM: | HLIN | HOME |
| | HPLOT | HTAB | | | |
| IF | INPUT | INT | INVERSE | | |
| LEFT\$ | LEN | LET | LIST | LN | LOADA |
| | LOADT | LOG | LOMEM: | | |
| MA | MID& | MOTOR | MP | | |
| NEW | NEXT | NORMAL | NOT | NOTRACE | |
| ON | ONERR | OR | | | |
| PDL(| PEEK | PLOT | POKE | POP | POS |
| | PRINT | | | | |
| READ | RECALL | REM | RESTORE | RESUME | RETURN |
| | RIGHT\$ | RND | ROT= | RUN | |
| SAVEA | SAVET | SCALE= | SCRN(| SGN | SHLOAD |
| | SIN | SOUND | SPC(| SPEED= | SQR |
| | STEP | STOP | STORE | STR\$ | |
| TAB(| TAN | TEXT | THEN | TK2000 | TO |
| | TRACE | | | | |
| USR | | | | | |
| VAL | VLIN | VTAB | | | |
| WAIT | | | | | |
| XDRAW | | | | | |

APENDICE F

MAPA DA MEMORIA



O mapa de memória mostra as diferentes funções a que foram dedicadas as várias áreas disponíveis.

Os primeiros 2k de RAM são de uso do sistema operacional e do BASIC do TK-2000 COLOR, assim como os últimos 16k são dedicados as portas de I/O (entrada e saída) e a ROM,

Podemos observar que sobraram entres os endereços 0800H e BFFFH duas áreas dedicadas a páginas de vídeo. A área restante é livre e disponível. As páginas de vídeo contêm as informações que aparecem na imagem da sua TV ou monitor. Como só uma aparece por vez, pode-se selecionar qual delas se deseja no momento, pelo uso dos comandos MA e MP. MA seleciona a página compreendida entre 2000H e 3FFFH, e MP seleciona as página compreendida entre A000H e BFFFH. Ao ligar seu computador é selecionada automaticamente a primeira página. Se desejar usar só uma página de vídeo, a área da outra fica livre e disponível para o usuário.

O programa em BASIC começa automaticamente a partir do endereço 0800H. Se o programa foi maior que 6k, existira um conflito com a primeira página de vídeo. Neste caso existem duas opções: a) digite o comando MP e a área normalmente dedicada a primeira página de vídeo poderá ser usada para o programa, permitindo assim acesso a 38k de RAM. b) Defina a posição de início do seu programa a partir do endereço 4000. Isto é obtido digitando os seguintes comandos diretos:

```
>LOMEM: 16384
>POKE 103,1
>POKE 104,64
>POKE 16384,0
>NEW
```

O comando LOMEM: é usado para definir o limite inferior da área de variáveis. O comando HIMEM: define o limite superior. For exemplo, após ligar o TK-2000 COLOR digite o seguinte comando direto:

```
HIMEM: 2060
```

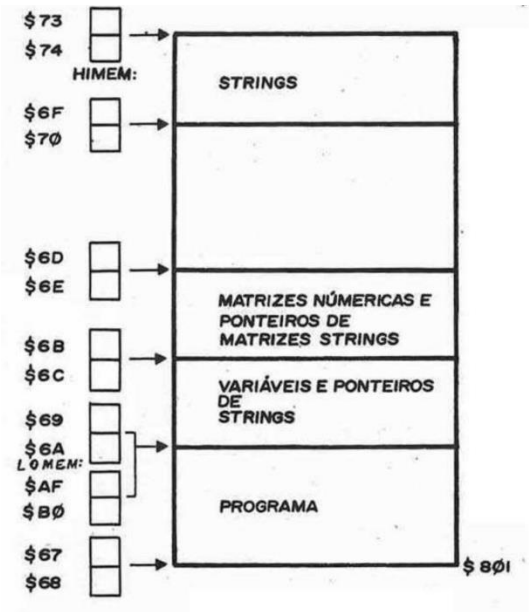
Agora tente entrar vários comandos do tipo:

```
10 REM
20 REM
```

e aparecerá o comentário de erro: EXCEDE MEMORIA indicando que não há mais espaço na área de programa BASIC.

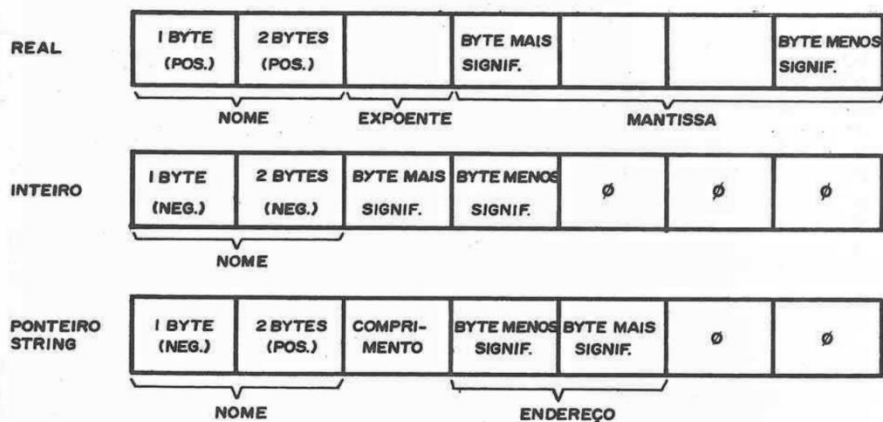
Mapa de memória de utilização do BASIC

A memória disponível pelo usuário & dividida em várias áreas pelo BASIC, e cada uma dessas áreas é endereçada por ponteiros que se situam nos endereços iniciais da memória, na área de variáveis do sistema. À figura abaixo descreve as áreas e seus ponteiros.



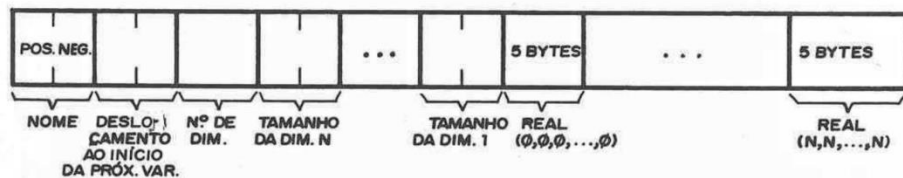
CONFIGURAÇÃO DAS VARIÁVEIS

Variáveis Simples

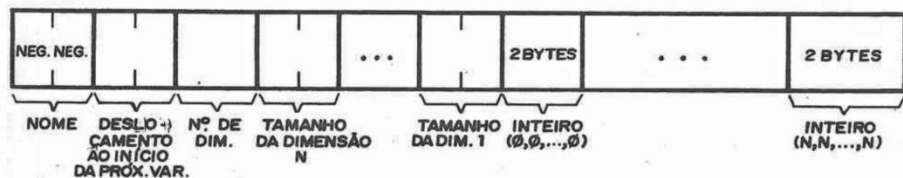


Matrizes

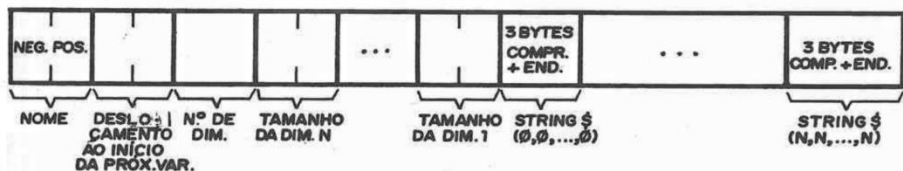
Real



Inteiro



Ponteiro String



APENDICE G

TABELA DE CODIGOS ASCII

| Decimal Hex | 00 \$00 | 16 \$10 | 32 \$20 | 48 \$30 | 64 \$40 | 80 \$50 |
|----------------|------------|------------|------------|------------|------------|------------|
| 0 \$0 | nul | dle | | 0 | @ | P |
| 1 \$1 | soh | dc1 | ! | 1 | A | Q |
| 2 \$2 | stx | dc2 | " | 2 | B | R |
| 3 \$3 | etx | dc3 | # | 3 | C | S |
| 4 \$4 | eot | dc4 | \$ | 4 | D | T |
| 5 \$5 | enq | nak | % | 5 | E | U |
| 6 \$6 | ack | syn | & | 6 | F | V |
| 7 \$7 | bel | etb | ' | 7 | G | W |
| 8 \$8 | bs | can | (| 8 | H | X |
| 9 \$9 | ht | em |) | 9 | I | Y |
| 10 \$A | lf | sub | * | : | J | Z |
| 11 \$B | vt | esc | + | ; | K | [|
| 12 \$C | ff | fs | , | < | L | \ |
| 13 \$D | cr | gs | - | = | M |] |
| 14 \$E | so | rs | . | > | N | ^ |
| 15 \$F | si | us | / | ? | O | _ |

APENDICE H

OPERAÇÃO EM LINGUAGEM DE MÁQUINA

Este apêndice foi escrito para aqueles que entendem programação em linguagem de máquina.

Os comandos específicos para esta utilização são mencionados a seguir:

LM Acessa o modo MONITOR

ASS Acessa o modo MINI-ASSEMBLER

DSK Com uma unidade de diskete conectada ao seu TK-2000
COLOR, este comando efetuará o "boot" do DOS do
diskete.

TK2000 Executa uma rotina em linguagem de máquina no
endereço \$3F8.

Para maiores detalhes consulte ao Manual Técnico do TK-2000
COLOR

APENDICE I

COMPARAÇÃO COM APPLE II PLUS

Existe um certo grau de compatibilidade entre o TK-2000 COLOR e o computador APPLE II PLUS e similares nacionais. Neste apêndice serão destacadas as diferenças mais importantes.

a) Diferença nos comandos BASIC

| | TK-2000 | APPLE II | Comandos no TK-2000 com o mesmo código do APPLE II |
|--------|------------------------|-------------------|--|
| FLASH | - | ✓ | SOUND |
| IN# | - | ✓ | ASS |
| PR# | - | ✓ | DSK |
| MA | ✓ | - | |
| MP | ✓ | - | |
| ASS | ✓ | - | |
| LM | ✓ | - | |
| TK2000 | ✓ | - | |
| MOTOR | ✓ | - | |
| SOUND | ✓ | - | |
| GR,HGR | (limpa só a janela) | (limpa a tela) | |

b) Carregando fita de APPLE II no TK-2000

Para carregar as fitas em APPLESOFT BASIC no TK-2000, digite o comando MP e a tela mostrará um padrão de barras similar aquele que aparece ao ligar o aparelho.

Nesta posição pode-se também digitar o comando HOME para limpar a tela tornando agora possível carregar um programa gerado num APPLE II ou compatível no TK-2000 COLOR através do comando LOADA.

Se o programa não executar corretamente, deve-se procurar a solução para o erro observando o mapa da memória, ou conferindo no programa se não existem chamadas de subrotinas da ROM que possuem endereços diferentes no TK-2000 e no APPLE II ou similar.

Caso ao executar o programa, este seja interrompido com a mensagem de ERRO DE SINTAXE, provavelmente este ocorreu devido a um comando (FLASH, IN# ou PR#) que existia no APPLE (incompatível ao TK-2000), e deverá ser removido ou substituído apropriadamente de acordo ao contexto.

c) Diferenças no uso da RAM ocupado pelo video

| | TK2000 | APPLE II |
|-----------------|-------------|-------------|
| TEXTO página 1 | 2000-3FFFFH | 400-7FFH |
| TEXTO página 2 | A000-BFFFFH | 800-BFFH |
| BAIXA RES pág 1 | 2000-3FFFFH | 400-7FFH |
| BAIXA RES pág 2 | A000-BFFFFH | 800-BFFH |
| ALTA RES pág 1 | 2000-3FFFFH | 2000-3FFFFH |
| ALTA RES pág 2 | A000-BFFFFH | 4000-5FFFFH |

INDICE ALFABETICO

ABS, 154
APPLE II, 205
Arredondamento, 53
ASC, 106-107
ASCII
 códigos, 195-106,291
ASS, 203
ATN, 155
BASIC, 17
Cadeia (String), 51,77
CALL, 163-173 Veja também USR
Caracteres Gráficos, 24
Cassettes, 27
 regulando o volume, 29
 guardando programas no, 84-85
 lendo memória a partir do, 84-85
 manuseio, 27
 proteção contra gravação, 28
CHR\$, 147
CLEAR, 92
Códigos de comandos, 193
COLOR, 127
Condicionais, 137
CONT, 85-87
CONTROL, 25
 C, 58-59, 87-88
 Q, 44
 S, 43
 X, 72, 93
CONTROL-SHIFT, 25
COS, 150
Cursor, 22
 determinação da posição horizontal, 90
 determinação da posição vertical, 90
 movimentação, 26
DATA, 117-122
Declaração de atribuição, 55
Declaração PRINT, 41
 abreviada, 43
 pontos e virgulas em, 42-49
 SPC em, 91
 TAB em, 90
 virgulas em, 42
DEF FN, 123
DEL, 72
Depuração de programas, 88
DIM, 110
Dois pontos (:), 49
DRAW, 167
Economia de espaço e tempo, 191
Edição, 72
 eliminando linhas, 72
 trocando caracteres, 73

- eliminando caracteres, 74
 - eliminando linhas, 74
- END, 43
- Erros
 - códigos dos, 189
 - corrigindo, 144
 - mensagens de, 187-188
- EXP, 155
- FN, 123
- FOR, 65-72
- FRE, 93
- Funções Trigonométricas, 191
- GET, 122
- GOSUB, 141
- GOTO, 57
- GR, 127
- Gráficos, 151
 - alta resolução, 159-163
 - baixa resolução, 127-129
 - caracteres, 24, 151
- HCOLOR, 130
- HGR, 130
- HGR2, 134
- HIMEM:, 174
- HLIN, 129
- HOME, 43
- HPLLOT, 130-133
- HTAB, 90
- IF-THEN, 137-138
- INPUT, 59-63
- INVERSE, 93
- INT, 83
- LEFT\$, 78
- LEN, 77
- LET, 55
- LIST, 45
- LM, 163
- LOADA, 84-85
- LOADT, 30, 84-85
- LOG, 156
- LOMEM:, 174
- Matrizes, 198
 - Unidimensionais, 111
 - Bidimensionais, 112
 - Tridimensionais, 113
- Memória, 13
 - acesso direto via BASIC, 173
 - apenas leitura (ROM), 164
 - endereçamento, 164
 - mapas, 197-198
- MID\$, 80
- MOD, 186
- Modo imediato, 35
- Modo programado, 44
- Monitor, 163

- acessando, 163
 - guardando, 164
 - lendo diretamente do, 165
 - saindo, 166
- NEXT, 65
- NEW, 41
- NOTRACE, 88
- NORMAL, 93
- Notação científica, 52, 99
- Números
 - em notação científica, 52, 99
 - inteiros, 52
 - reais, 52
- Número aleatório, 82
- Número de linha, 44
 - como endereço, 164
- Números hexadecimais, 162
- ON-GOSUB, 143
- ON-GOTO, 139
- ONERR GOTO, 144
 - rotina de manipulação de erros, 144
- Operações lógicas, 101
- Operações comparativas, 99
- Palavras reservadas, 195
- PEEK, 89
- PLOT, 127
- POKE, 89
- Ponto e vírgula, 42-49
- POP, 145
- POS, 9, 11
- PRINT, 41
- RAM, 14
- READ, 117-122
- RECALL, 113-114
- REM, 63
- REPEAT, 27
- RESET, 26, 87-88
- RESTORE, 121
- RETURN, 141
- RETURN (TECLA), 25
- RIGHT\$, 79
- ROM, 13
- ROT, 168
- RND, 82, 153
- RUN, 44
- Saídas
 - alinhando valores numéricos, 41
 - cadeias, 103
- SAVEA, 84-85
- SAVET, 84-85
- SCALE, 168
- SCRN, 129
- SGN, 155
- SHIFT, 22
- SHLOAD, 165, 169

SIN, 149
SOUND, 177-181
SPC, 91
SPEED, 93
SQR, 38, 156
STOP, 85-87
STORE, 113,114
STR\$, 103
Sub-rotinas, 141
TAB, 90
Tabelas de figuras, 164
TAN, 150
Teclado, 22-27
TEXT, 127-130
TK2000, 203
TRACE, 88
USR, 174
VAL, 105
Variáveis, 54, 97
 inteiras, 54, 97
 reais, 54, 88
Video-mapas, 206
Virgulas, 42
VLIN, 129
VTAB, 90
WAIT, 173
XDRAW, 167

