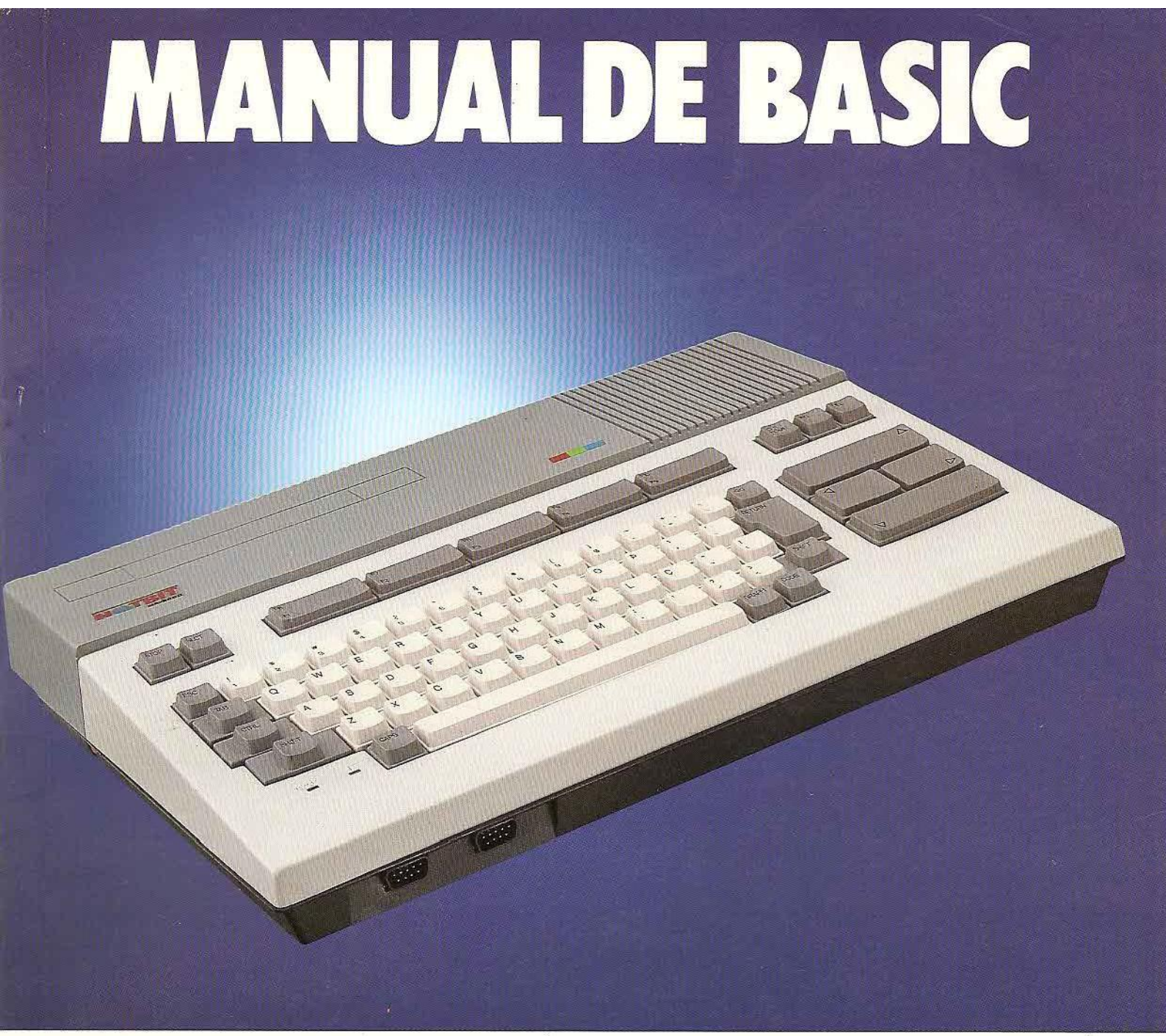
MANUAL DE BASIC



HOME COMPUTER

HOTBIT

HB-8000

HOTBIT MANUAL DE BASIC

Este manual contém as descrições de todos os comandos, instruções e funções do BASIC, dispostas por comodidade em ordem alfabética.
As descrições estão estruturadas da seguinte forma:

TIPO: Indica a que categoria pertence: comando, instrução ou função.

FORMATO: Enuncia os parâmetros utilizados pelos comandos, instruções ou funções (vide nesta página: Notações de formato)

EXEMPLO: É mostrado um exemplo para ilustrar a aplicação do comando, instrução ou função.

USO: Neste item será dada uma breve explicação sobre o comando, instrução ou função, isto é, o motivo pelo qual é utilizado.

Cada um dos comandos, instruções e funções traz, após a explicação, um programinha como exemplo do uso do mesmo.

NOTAÇÕES DE FORMATO:

1. Todas as letras que dentro do "formato" vêm escritas em letras maiúsculas, deverão ser digitadas através do teclado exatamente na mesma forma em que aparecem no manual.
2. Os elementos entre parênteses angulares ⟨ ⟩ deverão ser definidos pelo usuário.
3. Os elementos entre parênteses quadrados [ ] são optativos.
4. Todos os sinais de pontuação, à exceção dos parênteses quadrados e angulares, deverão ser digitados exatamente da mesma forma em que está indicado no formato. Isso se aplica aos pontos, vírgulas, parênteses, dois pontos, ponto e vírgula, etc.
5. "X", "Y" e "Z" ou "expressão" representam expressões numéricas, variáveis ou constantes.
6. "X$", "Y$" e "Z$" ou "string" representam expressões alfanuméricas.
7. O apóstrofe (') que fica na mesma tecla do trema (¨ ·) é usado nos exemplos, como instrução rem. (vide página 88).

ABS

TIPO — Função — Numérica
FORMATO — **ABS** (⟨expressão⟩)
EXEMPLO — B = **ABS** (-2)
USO — Indica o valor absoluto do número entre parênteses, ou seja, seu valor sem sinal. O valor absoluto de um número negativo é o mesmo número multiplicado por -1.

PROGRAMA EXEMPLO

```
10 CLS
20 PRINT"Adivinhar um numero entre 0 e 99":PRINT
30 A=INT(RND(-TIME)*100)
40 INPUT"Seu palpite e";B:K=K+1
50 C=ABS(A-B)
60 IFC=0THEN100
70 IFC<10THENPRINT"Voce esta perto!":GOTO40
80 IFC<20THEN PRINT"Voce esta longe!":GOTO40
90 PRINT"Completamente errado!":GOTO40
100 PRINT"Acertou em ";K;"jogadas"
```

ASC

TIPO — Função — Numérica
FORMATO — **ASC** (⟨string⟩)
EXEMPLO — A = **ASC** ("A")
USO — **ASC** fornecerá um valor numérico entre 0 e 255, correspondente ao código ASCII do primeiro caractere do "string". No caso do "string" ser nulo (i.e.: não ter nenhum caractere), o uso do **ASC** originará uma mensagem de erro: FUNÇÃO ILEGAL.
REFERÊNCIAS — CHR$, Tabela de Código de Caracteres.

PROGRAMA EXEMPLO 1 (Código decimal)

```
10 CLS:A$="ABCdef1230"
20 FORI=1TO10
30 B$=MID$(A$,I,1)
40 PRINT"O codigo ASCII de ";B$;" e ";ASC(B$)
50 NEXT:END
```

PROGRAMA EXEMPLO 2 (Código hexadecimal)

```
10 CLS:A$="ABCd@f1230"
20 FOR I=1 TO 10
30 B$=MID$(A$,I,1)
40 PRINT "O codigo ASCII DE ";B$;" é ";HEX$(ASC(B$))
50 NEXT:END
```

ATN

TIPO: Função — Numérica
FORMATO: **ATN** ((expressão))
EXEMPLO: A = **ATN** (1)
USO: Essa função matemática indica o arco-tangente da expressão. O resultado é o ângulo (em radians) cuja tangente é a expressão dada. O resultado estará sempre entre $-\pi/2$ e $\pi/2$

PROGRAMA EXEMPLO

```
10 CLS:PI=3.1415926536#
20 PRINT" x(graus)   atn(x)"
30 FORI=0TOPISTEPPI/20
40 PRINTUSING"####";INT(I/PI*180);
50 PRINTATN(I)
60 NEXT
70 END
```

AUTO

TIPO: Comando
FORMATO: **AUTO** [⟨No. de linha⟩] [,⟨incremento⟩]
EXEMPLO: **AUTO** 100,5
USO: Geração automática de um número de linha de programa após cada retorno de carro.

O comando **AUTO** começará a numeração automática a partir do número de linha indicado por ⟨No. de linha⟩, e incrementará cada número de linha subseqüente no valor indicado por ⟨incremento⟩. O "default" para esses dois valores é 10. Se o ⟨No. de linha⟩ vier seguido de uma vírgula mas o valor ⟨incremento⟩ não estiver especificado, continuará sendo respeitado o último valor de incremento dado por um comando **AUTO.**

Se o comando **AUTO** gerar um No. de linha já existente na memória, aparecerá um asterisco impresso após o número, a fim de advertir ao usuário que o que for introduzido nessa linha substituirá o conteúdo já existente. Todavia, se for pressionada a tecla ENTER imediatamente após o aparecimento do asterisco, a linha primitiva será mantida e a linha seguinte será gerada.

Para sair do modo **AUTO,** deve-se pressionar simultaneamente CONTROL-C ou CONTROL-STOP. A linha na qual CONTROL-C foi pressionado não será salvada. Após a digitação de CONTROL-C, o BASIC volta ao nível de comandos.

BASE

(Trata-se de uma variável de sistema que pode ser avaliada ou atribuída como qualquer variável ordinária. Será de utilidade somente para programadores experientes. Aconselha-se não fazer uso da mesma no caso de não conhecer o seu significado).

TIPO: Variável de Sistema.
FORMATO: BASE (⟨n⟩)
EXEMPLO: BASE (10)
USO: O processador de vídeo (VDP) possui 8 registros WRITE-ONLY. Os registros do VDP definem os endereços base para diversos sub-blocos dentro da VRAM. Esses sub-blocos formam tabelas que são usadas para produzir a imagem desejada na tela do TV.
O valor de **BASE** (⟨ n ⟩) determina o endereço inicial de cada tabela do registro de VDP, relacionado com a saída na tela.
Para maiores detalhes vide manual do usuário (Apêndice VDP).
Nessa variável de sistema não poderá ser colocado nenhum valor.

A descrição de ⟨n⟩ vem a seguir:

0. Modo texto. Base da tabela de nomes (40*24 letras)
1. Sem significado.
2. Modo texto. Base da tabela do gerador de padrões (40*24 letras)
3. Sem significado.
4. Sem significado.

5. Modo texto. Base da tabela de nomes (32*24 letras)
6. Modo texto. Base da tabela de cor (32*24 letras)
7. Modo texto. Base da tabela do gerador de padrões 32*24 letras)
8. Modo texto. Base da tabela de atributos de sprites (32*24 letras).
9. Modo texto. Base da tabela de padrão de sprites (32*24 letras).

10. Modo Alta Resolução Base da tabela de nomes.
11. Modo Alta Resolução. Base da tabela de cor
12. Modo Alta Resolução. Base da tabela do gerador de padrões
13. Modo Alta Resolução. Base da tabela de atributos de sprites.
14. Modo Alta Resolução. Base da tabela de padrão de sprites.

15. Modo Multi-Cor Base da tabela de nomes.
16. Sem significado.
17. Modo Multi-Cor. Base da tabela do gerador de padrões.
18. Modo Multi-Cor. Base da tabela de atributos de sprites.
19. Modo Multi-Cor. Base da tabela de padrão de sprites.

BEEP

TIPO Comando
FORMATO **BEEP**
EXEMPLO **BEEP**
USO Esse comando origina um som de "beep" no alto-falante de aproximadamente 0,04 seg. de duração.
O comando BEEP gera exatamente a mesma saída (som) que a função CHR$ (7)
REFERÊNCIAS Consultar tabela de códigos de controle.

PROGRAMA EXEMPLO

```
10 BEEP
20 FORI=0TO100
30 NEXT
40 PRINTCHR$(7)
50 END
```

BIN$

TIPO Função – string
FORMATO **BIN$** (⟨expressão⟩)
EXEMPLO L PRINT **BIN$** (A)
USO Transformar números de base decimal para binário.
⟨EXPRESSÃO⟩ é um valor numérico compreendido entre -32768 e 65535.
Se a ⟨EXPRESSÃO⟩ for negativa, será usada a forma "complemento de dois". Isto é: **BIN$** (-1) é igual a **BIN$** (65536-1)
REFERÊNCIAS VAL, OCT$, HEX$
PROGRAMA EXEMPLO

```
10 A=123:B=234
20 PRINT"a=";BIN$(A)
30 PRINT"b=";BIN$(B)
40 C=AANDB
50 PRINT"a and b=";C;BIN$(C)
60 C=AORB
70 PRINT"a or b=";C;BIN$(C)
80 C=AXORB
90 PRINT"a xor b=";C;BIN$(C)
100 C=AEQUB
110 PRINT"a equ b=";C;BIN$(C)
120 C=AIMPB
130 PRINT"a imp b=";C;BIN$(C)
140 END
```

BLOAD

TIPO — Comando

FORMATO — **BLOAD** "⟨dispositivo⟩ : ⟨nome do arquivo⟩" [,R] [, ⟨ offset ⟩]

EXEMPLO — **BLOAD** "CAS : TESTE", R

USO — Para carregar um programa em linguagem de máquina, a partir do dispositivo especificado em ⟨dispositivo⟩

O dispositivo pode ser

CAS = gravador
A = acionador de disco 1
B = acionador de disco 2

No caso da opção R estar especificada, após o programa ter sido carregado, começará a execução automaticamente a partir do endereço indicado por BSAVE.

O programa em linguagem de máquina já carregado, será armazenado na localização de memória indicada por BSAVE.

No caso de aparecer ⟨offset⟩ no comando, todos os endereços especificados por BSAVE serão deslocados no valor do "offset".

No caso do ⟨nome do arquivo⟩ ter sido omitido, o primeiro programa em linguagem de máquina encontrado será carregado na memóri a.

REFERÊNCIAS — BSAVE

NOTA — "BLOAD CAS" SÓ PODERÁ SER UTILIZADO QUANDO O COMPUTADOR ESTIVER CONECTADO A UM GRAVADOR.

"BLOAD A" OU BLOAD B" SÓ PODERÃO SER UTILIZADOS QUANDO O COMPUTADOR ESTIVER CARREGADO COM O HB-DOS E OS ACIONADORES DE DISCO 1 OU 2

BSAVE

TIPO — Comando

FORMATO — **BSAVE** "⟨dispositivo⟩ : ⟨nome do arquivo⟩", ⟨endereço inicial⟩, ⟨end. final⟩ [, ⟨endereço de execução⟩]

EXEMPLO 1 — **BSAVE** "CAS : TESTE", & HA1ØØ, & HA2FF

EXEMPLO 2 — **BSAVE** "A: TESTE", & HEØØØ, & HEØFF, & HEØ2Ø

USO — Para salvar um programa em linguagem de máquina, localizado na memória entre os endereços inicial e final, através do dispositivo indicado por ⟨dispositivo⟩

• Os dispositivos podem ser:

CAS = gravador
A = acionador de disco 1
B = acionador de disco 2

⟨endereço inicial⟩ e ⟨endereço final⟩ são os endereços de início e fim da área a ser salvada.

No caso de ⟨endereço de execução⟩ ter sido omitido, o ⟨endereço inicial⟩ será considerado como tal.

REFERÊNCIAS — BLOAD.

NOTA — "BSAVE CAS" SÓ PODERÁ SER UTILIZADO QUANDO O COMPUTADOR ESTIVER CONECTADO A UM GRAVADOR

"BSAVE A" E "BSAVE B" SÓ PODERÃO SER UTILIZADOS QUANDO O COMPUTADOR ESTIVER CARREGADO COM O HB-DOS E OS ACIONADORES DE DISCO 1 OU 2.

CALL

TIPO: Instrução
FORMATO: **CALL** ⟨nome da instrução expandida⟩ [(⟨lista de argumentos⟩)]
EXEMPLO: **CALL** TALK ("A", "BC", "DEF")
_TALK ("A", "BC", "DEF")
USO: Para chamar uma instrução expandida fornecida pelo cartucho ROM. (Vide MEM. SLOT para mais detalhes). "_" é uma abreviação para "CALL", de forma que os dois exemplos acima são equivalentes.

CDBL

TIPO: Função
FORMATO: **CDBL** (⟨expressão⟩)
EXEMPLO: A = **CDBL** (B!/2)
USO: Transforma a ⟨expressão⟩ em um número de dupla precisão.
Porém o número de dígitos efetivos não altera apenas com a transformação de forma. A precisão do valor obtido será igual à forma anterior à transformação (se for uma constante numérica, será de 6 dígitos efetivos).

PROGRAMA EXEMPLO

```
10 A#=CDBL(9/7)
20 PRINTA#
30 END
```

CHR$

TIPO: Função — String
FORMATO: **CHR$** (⟨ expressão ⟩)
EXEMPLO: A$ = **CHR$** (65)
USO: Fornece um "STRING" cujo único elemento é o caractere correspondente ao código ASCII igual a ⟨ expressão ⟩
CHR$ é comumente usado para transferir um caractere especial para a tela.
A ⟨ expressão ⟩ deverá estar compreendida entre 0 e 255 e se for superior a 32 poderá ser visualizado na tela.
Se a ⟨ expressão ⟩ estiver definida como código de controle, a função que esse código define será executada e a letra correspondente ao código de caracteres não será visualizada na tela.
Se a ⟨ expressão ⟩ não estiver compreendida entre 0 e 255 aparecerá uma mensagem de erro "FUNÇÃO ILEGAL".
REFERÊNCIAS: ASC, Tabela Código de Caracteres

```
10 CLS:PRINT"Visualizacão de CHR$(32) até CHR$(255)"
20 PRINT
30 FORI=32TO255
40 PRINTCHR$(I);
50 NEXT:PRINT:PRINT
60 PRINT"Visualizacão de CHR$(1)+CHR$(65)"
70 PRINT"até CHR$(1)+CHR$(95)"
80 PRINT
90 FORI=65TO95
100 PRINTCHR$(1)+CHR$(I);
110 NEXT
```

CINT

TIPO: Função — numérica
FORMATO: **CINT** (⟨ expressão ⟩)
EXEMPLO: A% = **CINT** (B# * 2)
USO: Transforma a ⟨ expressão ⟩ em um número inteiro, truncando a parte decimal.
Se a ⟨ expressão ⟩ não estiver compreendida entre -32768 e 32767, aparecerá uma mensagem de erro de "OVERFLOW".
REFERÊNCIAS: CSNG, CDBL
PROGRAMA EXEMPLO:

```
10 A%=CINT(9/7)
20 PRINTA%
30 END
```

CIRCLE

TIPO: Instrução

FORMATO: **CIRCLE** | (x, y) / STEP (x, y) | , ⟨raio⟩ [, ⟨cor⟩] [, ⟨ ângulo início ⟩] [, ⟨ ângulo fim ⟩] [, ⟨relação de raios⟩] .

EXEMPLO: **CIRCLE** (80, 80), 40, 14, 0, 6, 2

USO: Para traçar uma elipse de centro e raios especificados nos dois primeiros argumentos.
(x, y) determinam as coordenadas do centro da elipse (ou círculo).
A forma STEP (offset X, offset Y) define a localização de um ponto relativo ao último ponto de referência (Vide instrução PUT SPRITE para maiores detalhes).
A ⟨cor⟩ refere-se à cor do círculo. (Se nada for especificado o "default" é a cor do texto). Os parâmetros ⟨ângulo início⟩ e ⟨âng. fim⟩ são argumentos expressados em radianos, com valores entre 0 e 2*PI e que permitem especificar os pontos nos quais começará e terminará o traçado da elipse. Se o ângulo de início ou o de fim for negativo, a elipse será conectada ao ponto central com uma linha e os ângulos serão tratados como se fossem positivos. (Note-se que é diferente de somar 2*PI).
A ⟨relação de raios⟩ indica a relação entre os raios vertical e horizontal da elipse.
Se os ângulos de início ou fim não estiverem compreendidos entre -2π e 2π, aparecerá uma mensagem de erro "FUNÇÃO ILEGAL"

REFERÊNCIAS: COLOR, SCREEN.

PROGRAMA EXEMPLO 1

```
10 '***laranja***
20 COLOR10,15:SCREEN2
30 CIRCLE(128,96),76,10,,,1.15
40 CIRCLE(128,96),60,10,,,1.15
50 PAINT(128,96),10
60 FORI=-6.28#TO-0STEP.3
70 CIRCLE(128,96),60,15,I,I+.3,1.15
80 NEXT
90 FORI=0TO2000:NEXT
100 COLOR15,4,7
```

PROGRAMA EXEMPLO 2

```
10 SCREEN2:CIRCLE(50,50),60,10,,,1.15
20 FORI=0TO2:IFI<=2THENCIRCLESTEP(50,50),60,15,,,1.15:NEXT
30 FORI=0TO2000:NEXT
```

CLEAR

TIPO — Instrução

FORMATO — **CLEAR** [⟨ espaço string⟩ [, ⟨ localização máxima⟩]]

EXEMPLO — **CLEAR** 100, &HE000

USO — Para zerar todas as variáveis numéricas, anular todas as variáveis string, fechar todos os arquivos abertos, e, opcionalmente, fixar o fim da memória. Inutiliza também todas as funções definidas por DEFFN, DEFSNG, DEFDBL, DEFSTR, DEFUSR, DEFINT.

⟨ espaço string⟩ define o espaço destinado ao armazenamento de variáveis string. O tamanho "default" é de 200 bytes.

⟨ localização máxima⟩ define a localização mais alta de memória disponível para uso do BASIC. Por isso, os dados e o programa em linguagem de máquina, colocados após esta área, até &HF380, não serão modificados pelo BASIC. O valor de ⟨ localização máxima⟩ deve estar compreendido entre &H831F e &HF380.

No caso de definir a ⟨ localização máxima⟩ , deverá também definir-se o ⟨ espaço string⟩ .

PROGRAMA EXEMPLO

```
10 A=10:B$="teste"
20 PRINTA,B$
30 CLEAR
40 PRINTA,B$
50 END
```

CLOAD/ CLOAD?

TIPO — Comando

FORMATO — **CLOAD** ["nome do arquivo"]
CLOAD? ["nome do arquivo"]

EXEMPLO — **CLOAD** "TESTE"
CLOAD? "TESTE"

USO — O comando **CLOAD** utiliza-se para carregar na memória um programa BASIC arquivado em fita cassette.

O comando **CLOAD?** utiliza-se para verificar ou comparar um programa BASIC da fita cassette com um programa na memória.

O comando **CLOAD** fecha todos os arquivos abertos e apaga o programa que encontrar na memória. No caso do ⟨ nome do arquivo ⟩ ser omitido, o primeiro programa arquivado na fita encontrado, será carregado na memória. Em todas as operações de leitura de fita cassette, a velocidade "baud rate" será determinada automaticamente.

Enquanto o arquivo de nome determinado no comando estiver sendo procurado, qualquer outro nome achado, provocará a visualização na tela da mensagem PULEI ⟨ nome ⟩ . Ao achar o programa desejado, aparecerá ACHEI ⟨ nome do arquivo ⟩ .

O ⟨ nome do arquivo ⟩ deverá ter 6 letras. Se tiver mais de 7, somente 6 serão consideradas, e o resto ignoradas.

O comando CLOAD? é usado normalmente após um CSAVE, a fim de verificar se o programa da memória foi corretamente carregado na fita cassette. Quando ambos programas forem iguais, aparecerá na tela: OK. Se não forem iguais, aparecerá uma mensagem de erro "ERRO/VERIF.".

REFERÊNCIAS — CSAVE.

NOTA — "CLOAD" e "CLOAD?" só poderão ser utilizados quando o computador estiver conectado a um gravador.

CLOSE

TIPO Comando.
FORMATO **CLOSE** [[#] ⟨ Nº do arquivo⟩ [, [#] ⟨ Nº do arquivo] ...]
EXEMPLO **CLOSE # 1**
USO Para fechar o canal e liberar o "buffer" a ele associado.
No caso de não haver ⟨Nº de arquivo⟩ especificado, todos os canais serão fechados.
O uso de Nºs de arquivo sucessivos possibilita fechar vários arquivos ao mesmo tempo. Se o comando não tiver Nº de arquivo, todos os arquivos abertos fechar-se-ão.
Em relação ao arquivo fechado, não se poderá efetuar nenhuma entrada ou saída, até abri-lo novamente com o comando OPEN.
No caso de ter dado um comando OPEN para saída do arquivo, o comando **CLOSE** provocará a expulsão dos dados que restaram no "buffer". Por isso, para encerrar corretamente o processamento de saída para o arquivo, o CLOSE deverá ser efetuado indispensavelmente.
Os comandos END e NEW também fecham todos os arquivos, o comando STOP não os fecha.
REFERÊNCIAS OPEN, PRINT#, INPUT#, END, NEW.
PROGRAMA EXEMPLO

```
10 MAXFILES=1
20 OPEN"cas:teste"FOR OUTPUT AS #1
30 A$="ARQUIVO"
40 PRINT#1,A$
50 CLOSE#1
60 END
```

CLS

TIPO Instrução
FORMATO **CLS**
EXEMPLO **CLS**
USO Para limpar a tela. É válido em todos os modos de SCREEN. Somente não será apagada a visualização das teclas de função. Ao apagar a tela, o cursor move-se para o ponto de coordenadas (0,0).
REFERÊNCIAS Color
PROGRAMA EXEMPLO

```
10 PRINTCHR$(12)
20 WIDTH25
30 FORI=0TO40
40 X=INT(RND(1)*27):Y=INT(RND(1)*22)
50 LOCATEX,Y:IF(X>5)AND(Y>5)THENPRINT"lixo"
60 NEXT
70 LOCATE3,2:PRINT"DAR CLS"
80 FORI=0TO1000:NEXT
90 CLS
```

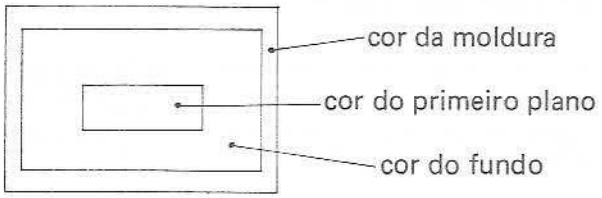
COLOR

TIPO Instrução

FORMATO **COLOR** [⟨cor do primeiro plano⟩] [, ⟨cor do fundo⟩] [, ⟨cor da moldura⟩] .

EXEMPLO **COLOR** 15, 7, 7.

USO Para definir as cores a serem utilizadas nas telas. O valor "default" é 15, 4, 4 . O argumento deve estar compreendido entre 0 e 15.



As cores correspondentes a cada valor são:

0 : transparente
1 : preto
2 : verde médio
3 : verde claro
4 : azul escuro
5 : azul claro
6 : vermelho escuro
7 : ciano
8 : vermelho médio
9 : vermelho claro
10 : amarelo escuro
11 : amarelo claro
12 : verde escuro
13 : magenta
14 : cinza
15 : branco

A cor do primeiro plano é utilizada para escrever letras no modo TEXTO ou pontos e traços no modo gráfico.

No SCREEN Ø não é possível mudar a cor da moldura. Por isso é que a necessidade de especificar a cor da moldura existe principalmente no uso de SCREEN2 ou SCREEN3.

REFERÊNCIAS CLS

PROGRAMA EXEMPLO

```
10 CLS
20 FORI=0TO15
30 FORJ=0TO15
40 PRINT"COLOR";I;",";J
50 COLORI,J
60 FORK=0TO300:NEXT
70 NEXTJ,I
80 FORI=0TO15
90 COLOR,,I
100 FORJ=0TO100:NEXT
110 NEXTI
120 COLOR15,0,0
```

NOTA: Pressionando SHIFT junto à tecla de função F1/F6, a tela volta às cores "default", (COLOR 15, 4, 4).

CONT

TIPO Comando.
FORMATO CONT
EXEMPLO CONT
USO Para continuar com a execução do programa após um BREAK ou um STOP durante a execução.
No caso da execução ser interrompida ao aparecer o ponto de interrogação devido à frase INPUT, só poder-se-á continuar através de uma nova frase INPUT.
Enquanto a execução estiver parada, poder-se-á executar LIST ou PRINT, porém, no caso de efetuar alguma alteração de conteúdo do programa, a execução não poderá ser continuada através do comando CONT.

PROGRAMA EXEMPLO

```
10 CLS:PRINT"PROGRAMA TESTE.DIGITE CONT"
20 END
30 PRINT"Oi!Tudo bem!"
```

NOTA: Enquanto o programa estiver parado, o programador poderá examinar ou mudar valores de variáveis. Assim sendo, é interessante colocar a instrução STOP em lugares estratégicos do programa a fim de poder "checar" o andamento do programa. O comando CONT fará com que o programa continue sendo rodado a partir do ponto de parada.

COPY

TIPO Instrução
FORMATO **COPY** "⟨dispositivo⟩: ⟨nome do arquivo-1⟩ "[TO "⟨dispositivo⟩: ⟨nome do arquivo-2⟩"]
EXEMPLO **COPY** "A : TESTE" TO "B : TESTE2"
USO Copiar um arquivo em disco.
Os "dispositivos" podem ser
A = acionador de disco1
B = acionador de disco2
⟨nome do arquivo-1⟩ é o nome do arquivo que deve ser copiado.
⟨nome do arquivo-2⟩ é o nome do arquivo cópia.
Se não vier indicado "TO" o arquivo será transferido ao outro acionador de disco sob o mesmo nome.

NOTA: "COPY" só poderá ser utilizado se o computador estiver carregado com o HB-DOS e os acionadores de disco 1 ou 2.

COS

TIPO Função.
FORMATO **COS** (⟨ expressão ⟩)
EXEMPLO **COS** (3.1415926535898/2)
USO Fornece o cosseno da ⟨expressão⟩ em radianos. O **COS**(X) é calculado com dupla precisão.
REFERÊNCIAS SIN, TAN.
PROGRAMA EXEMPLO

```
10 '***COSSENO E CIRCULO***
20 SCREEN2
30 PI=3.1415926536#
40 LINE(0,96)-(255,96)
50 LINE(191,0)-(191,191)
60 FORI=0TOPI*2STEPPI/100
70 X=I/PI*63
80 Y=96-63*COS(PI*X/64)
90 PSET(X,Y)
100 X=192+63*SIN(I)
110 Y=96-63*COS(I)
120 PSET(X,Y)
130 NEXT
140 GOTO140
```

CSAVE

TIPO Comando.
FORMATO **CSAVE** "⟨ nome do arquivo ⟩" [, ⟨ relação "baud" ⟩]
EXEMPLO **CSAVE** "TESTE1"
USO Para salvar um arquivo de programa BASIC em fita cassette.
O BASIC salva o arquivo em formato binário comprimido (simbólico). Os arquivos ASCII ocupam mais espaço, porém alguns tipos de acesso exigem que os arquivos estejam em formato ASCII. Por exemplo, um arquivo que pretenda ser usado com o comando MERGE, deverá ser salvado em formato ASCII. Os programas salvados em ASCII deverão ser lidos como arquivos BASIC, de dados e de texto. Nesses casos, usa-se o comando SAVE.
⟨ relação "baud" ⟩ é um parâmetro entre 1 e 2, que determina a relação de bauds default para cada operação de escrita em fita cassette. 1 determina 1200 bauds, e 2 determina 2400 bauds. A relação de bauds default pode também ser fixada pela instrução SCREEN.
REFERÊNCIAS CLOAD, BSAVE.

NOTA: **"CSAVE"** só poderá ser utilizado se o computador estiver conectado a um gravador.

CSNG

TIPO Função.
FORMATO **CSNG** (⟨ expressão⟩)
EXEMPLO A! = CSNG (B#)
USO Transforma a ⟨ expressão⟩ em número de simples precisão com 6 dígitos de algarismos efetivos.
No caso da ⟨ expressão⟩ não estar compreendida entre –9.99999E+62 e 9.99999E+62, dará erro de "overflow".
REFERÊNCIAS CINT, CDBL.
PROGRAMA EXEMPLO

```
10 A!=CSNG(9/7)
20 PRINTA!
30 END
```

CSRLIN

TIPO Variável de sistema.
FORMATO **CSRLIN**
EXEMPLO Y = **CSRLIN**
USO Fornece a coordenada vertical do cursor. A variável assume valores entre 0 e 23. A linha superior da tela, corresponde ao valor 0, e seu valor aumenta conforme o cursor desce.
Para saber a posição de cursor, usa-se a variável POS.
REFERÊNCIAS POS, LOCATE.
PROGRAMA EXEMPLO

```
10 SCREEN0
20 LOCATE10,20
30 PRINTCSRLIN
40 END
```

CVI/CVS CVD

TIPO Função.

FORMATO **CVI** (⟨ X$⟩)
CVS (⟨ Y$⟩)
CVD (⟨ Z$⟩)

EXEMPLO A% = **CVI** (N1$)

USO Converte um valor alfanumérico no seu correspondente valor numérico. Os valores numéricos que devam ser passados do buffer para um arquivo aleatório no disco, deverão ser convertidos de alfanuméricos em numéricos.

"CVI" converte uma variável alfanumérica de 2 bytes em um número inteiro.

"CVS" converte uma variável alfanumérica de 4 bytes em um número de precisão simples.

"CVD" converte uma variável alfanumérica de 8 bytes em um número de precisão dupla.

REFERÊNCIAS "FIELD".

NOTA: "CVI", "CVS" e "CVD" só poderão ser utilizadas quando o computador estiver carregado com o HB-DOS e os acionadores de disco 1 ou 2.

DATA

TIPO Instrução.

FORMATO DATA ⟨lista de constantes⟩

EXEMPLO DATA 123, ABC, 456.

USO A instrução **DATA** armazena constantes numéricas e "strings" dentro de um programa. O programa utilizará essas constantes através da instrução READ.

As instruções **DATA** não são executadas pelo programa e podem ser colocadas em qualquer lugar do mesmo. Normalmente elas são colocadas no fim do programa. A instrução **DATA** poderá conter tantas constantes quanto couber em uma linha (255 caracteres) (separadas por vírgulas), e dentro do programa poderá haver qualquer número de instruções **DATA**.

As instruções READ acessarão as instruções **DATA** pela ordem (pelo número de linha), e os dados nelas contidos poderão ser considerados como uma lista contínua de itens, não importando quantos itens houver em cada linha nem onde as linhas estiverem localizadas dentro do programa. Os dados serão "lidos" da esquerda para a direita, a partir da linha de menor número.

A ⟨lista de constantes⟩ poderá conter constantes numéricas com qualquer formato, i.e.: ponto fixo, ponto flutuante ou inteiro (Expressões numéricas não são permitidas nessa lista). As constantes "string" dentro da instrução DATA deverão ser colocadas entre aspas somente se elas contiverem vírgulas, dois pontos ou espaços em branco no início ou no fim. Em caso contrário as aspas não serão necessárias. O tipo de variável (numérica ou string) dada na instrução READ deverá concordar com a constante correspondente da instrução **DATA** caso contrário aparecerá mensagem de erro.

As instruções **DATA** poderão ser lidas novamente a partir do início ou de uma lista específica através da instrução RESTORE.

PROGRAMA EXEMPLO

```
10 READ A$,B$:PRINTA$;B$
20 FORI=1TO4
30 READ A$:PRINT A$
40 NEXT
50 READ A$,B$,A,B,C,D
60 PRINT A$;B$:PRINT A;B;C;D
70 DATA Para ler dados de
80 DATA " letras:"
90 DATA "   quando houver"
100 DATA espaços vazios na
110 DATA "frente, ou virgulas"
120 DATA "no meio,o dado"
130 DATA deverá estar entre
140 DATA " aspas"
150 DATA 3,&h20,&b1010,3.14
```

DEF FN

TIPO — Instrução.

FORMATO — **DEF FN** ⟨nome⟩ [(⟨lista de parâmetros⟩)] = ⟨definição da função⟩

EXEMPLO — **DEF FNA** (X, Y) = X*2 + Y*3

USO — Para definir e dar nome a uma função escrita pelo usuário.

⟨nome⟩ deverá ser um nome de variável permitida. Esse nome, precedido de FN, torna-se o nome da função.

⟨lista de parâmetros⟩ estará composta por aqueles nomes de variáveis utilizados na definição da função que deverão ser substituídos quando a função for chamada. Os itens da lista deverão estar separados por vírgulas. ⟨definição da função⟩, é uma expressão que realiza a operação da função. Seu comprimento está limitado a uma linha. Os nomes de variáveis que aparecem nessa expressão servem somente para definir a função, elas não afetarão as variáveis do programa que tenham o mesmo nome. Um nome de variável utilizado na ⟨definição da função⟩ poderá aparecer ou não, na ⟨lista de parâmetros⟩. Se aparecer, o valor do parâmetro será fornecido quando a função for chamada. Caso contrário, será utilizado o valor presente da variável.

As variáveis da ⟨lista de parâmetros⟩ representam as variáveis argumento ou valores que serão dados ao ser chamada a função.

Se o tipo for especificado no ⟨nome da função⟩, o valor da expressão deverá também ser forçado ao mesmo tipo, antes de ser devolvida à instrução de chamada. Se o tipo especificado no nome da função e o tipo do argumento não coincidirem, aparecerá uma mensagem de erro: "TIPO DESIGUAL".

A instrução DEF FN deverá ser executada antes que a função que ela define seja chamada. Se a função for chamada antes de ser definida, acontecerá um erro do tipo: "FUNÇÃO NÃO DEFINIDA".

A instrução DEF FN é ilegal no modo direto.

PROGRAMA EXEMPLO

```
10 DEFFNLN(X)=LOG(X)/LOG(10)
20 PRINT"    x     log(x)     log10(x)"
30 FORI=.1TO3.5STEP.1
40 PRINTUSING" #.#   ##.#####   ##.#####";I;LOG(I);FNLN(I)
50 NEXT
```

TIPO Instrução.
FORMATO **DEFINT** ⟨ lista de letras ⟩
DEFSNG ⟨ lista de letras ⟩
DEFDBL ⟨ lista de letras ⟩
DEFSTR ⟨ lista de letras ⟩
EXEMPLO **DEFINT** A , I–K
USO Essas instruções são usadas para declarar o tipo de uma variável como sendo inteira, simples precisão, dupla precisão ou "string".
As instruções **DEFINT/SNG/DBL/STR** declaram que as variáveis com nomes começando com as letras especificadas na ⟨ lista de letras ⟩, serão de um desses tipos de variáveis.
Todavia, os caracteres de declaração de tipo de variável (%, !, #, vide seção "VARIÁVEIS" para maiores detalhes sobre esses caracteres), tem precedência sobre as instruções **DEF xxx**, na impressão de uma variável.
O uso do comando CLEAR, cancela as instruções DEF xxx anteriores.

PROGRAMA EXEMPLO

```
10 DEFINT I
20 DEFSNG J
30 DEFDBL K
40 DEFSTR L
50 I=1.6:PRINTI
60 J=1!:PRINTJ
70 K=1/3:PRINTK
80 L="ABCDEF":PRINTL
90 CLEAR
100 PRINTI;J;K;L
```

DEFUSR

TIPO Instrução.
FORMATO **DEFUSR** [⟨ dígito ⟩] = ⟨ expressão inteira ⟩
EXEMPLO **DEFUSR3** = &HF000
USO Esta instrução especifica o endereço inicial de uma subrotina em linguagem assembly.
⟨ dígito ⟩ poderá ser qualquer dígito entre 0 e 9. Esse dígito corresponde ao número da rotina USR cujo endereço está sendo especificado.
Se o ⟨ dígito ⟩ for omitido, o computador assumirá o valor **DEFUSR0**.
O valor da ⟨ expressão inteira ⟩, é o endereço inicial da rotina USR.
Qualquer número de instruções **DEFUSR** poderão aparecer no programa, a fim de redefinir os endereços de início de subrotinas, permitindo dessa forma acessar a quantas subrotinas for necessário.

PROGRAMA EXEMPLO

```
10 CLEAR 200,&HEFFF
20 AA=&HF000:DEFUSR3=AA
30 PRINT"O endereço inicial da sub-rotina é=";
40 PRINT AA
50 END
```

DELETE

TIPO — Comando.

FORMATO — **DELETE** [⟨ número de linha inicial ⟩ — ⟨ número de linha ⟩]
⟨ número de linha ⟩
— ⟨ número de linha final ⟩

EXEMPLO — **DELETE 10-230**

USO — Apagar linhas de programa. Dependendo da especificação de número de linhas dado após o comando **DELETE** o comando apagará um trecho do programa compreendido entre duas linhas do mesmo, ou só uma linha do programa ou ainda o trecho compreendido entre o início do programa e uma linha determinada. Se colocar o comando **DELETE** seguido de um ponto (.), a execução do mesmo apagará a última linha do programa.

O BASIC sempre retorna ao nível de comando após a execução do **DELETE**.

No caso de não aparecer ⟨ número de linha ⟩ após o comando **DELETE**, aparecerá a mensagem de erro ⟨ FUNÇÃO ILEGAL ⟩.

EXEMPLOS — **DELETE 10**

(Apagará a linha 10 do programa).

DELETE 50-100

(Apagará o trecho do programa compreendido entre as linhas 50 e 100).

DELETE-100

(Apagará o trecho do programa compreendido entre o início e a linha 100).

DIM

TIPO — Instrução.

FORMATO — **DIM** ⟨ variável ⟩ (⟨ índices ⟩) [, ⟨ variável ⟩ (⟨ índices ⟩) ...]

EXEMPLO — **DIM** A (15, 20)

USO — Essa instrução define um "arranjo" ou matriz de variáveis. Especifica os valores máximos dos índices das variáveis de um "arranjo" e localiza o armazenamento conformemente.

Se uma variável de "arranjo" for utilizada sem a instrução **DIM**, o valor máximo de seus índices poderá ser de 10. No caso de utilizar um índice maior que o máximo especificado, aparecerá a mensagem de erro "ÍNDICE FORA DO LIMITE". O valor mínimo de um índice é sempre 0.

A variável de matriz que tornar-se inútil, poderá ser apagada mediante o comando ERASE. A variável definida através de DIM poderá ser redefinida somente após a execução do CLEAR ou do ERASE.

PROGRAMA EXEMPLO

```
10 CLS:PRINT"Usando o arranjo sem DIM"
20 FOR I=0TO10
30 PRINTA(I)
40 NEXT:PRINT
50 PRINT"Os indices poderão variar entre (0)-(10)"
60 FORI=0TO2500:NEXT
70 N=15:PRINT
80 PRINT"Se quisermos que o arranjo tenha 16 elementos"
90 PRINT"devemos fazer DIM A(15)"
100 ERASE A:DIM A(N)
110 FOR I=0TON:PRINT A(I),:NEXT
120 PRINT"Se colocarmos novamente DIM sem ERASE "
130 PRINT"Obteremos o seguinte erro:"
140 DIM A(16)
```

DRAW

TIPO Instrução.

FORMATO DRAW ⟨ expressão string ⟩

EXEMPLO DRAW "BM10, 10F 100U 100 G100 U100"

USO Para desenhar uma figura de acordo com a linguagem macro gráfica. Os comandos em linguagem macro gráfica estão contidos na ⟨ expressão string ⟩. O "string" define um objeto, que é desenhado quando o BASIC executa a instrução **DRAW**.

Durante a execução o BASIC examina o valor do string e interpreta os comandos (de uma letra) contidos no string.

Esses comandos são detalhados a seguir:

Os seguintes comandos do movimento começam a partir do último ponto de referência. Após cada comando, o último ponto referenciado será o último ponto desenhado pelo comando.

U(n) Movimenta para cima.
D(n) Movimenta para baixo.
L(n) Movimenta para esquerda.
R(n) Movimenta para direita.
E(n) Movimenta diagonalmente para cima e direita.
F(n) Movimenta diagonalmente para baixo e direita.
G(n) Movimenta diagonalmente para baixo e esquerda.
H(n) Movimenta diagonalmente para cima e esquerda.

O(n) em cada um dos comandos precedentes indica a distância do movimento. O número de pontos movido é n vezes o fator de escala (fixado pelo comando S).

Mx, y provoca movimentos absolutos ou relativos. Se x tiver na frente um sinal mais (+) ou um sinal menos (—), o movimento será relativo ao último ponto de referência. Se não tiver, será absoluto, referente à tela do monitor.

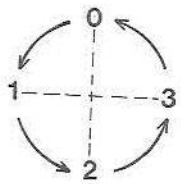
A proporção de aparência da tela é de 1. Ou seja, 8 pontos horizontais são iguais a 8 pontos verticais.

Os 2 seguintes comandos prefixos podem preceder qualquer dos comandos de movimento indicados acima:

B Movimenta, mas não traça pontos.
N Movimenta, mas volta à posição de origem ao terminar o desenho.

Existem também os seguintes comandos:

A(n) Fixa o ângulo n, n poderá variar entre 0 e 3, onde 0 equivale a 0 graus, 1 equivale a 90 graus, 2 a 180 e 3 a 270.



C(n) Fixa a cor n — n poderá variar entre 0 e 15.
S(n) Fixa o fator de escala. n poderá variar entre 0 e 255. n dividido por 4 é o fator de escala. Por exemplo, se n = 1, o fator de escala será 1/4. O fator de escala multiplicado pela distância dada nos comandos U, D, L, R, E, F, G, H e o relativo M, dará a distância real movida. O valor default é 0, que significa "sem escala" (isto é: equivale a S4).
X⟨ variável string ⟩ Executa os comandos de movimento através de uma variável alfanumérica.

EXEMPLO

```
A$ = "U80R80D80L80": DRAW" XA$;"
```

(Desenhará um quadrado).

Em todos esses comandos, os argumentos n, x ou y poderão ser constantes do tipo 123 ou "= ⟨ variável ⟩" onde ⟨ variável ⟩ será o nome de

(continua)

DRAW

(continuação)

uma variável numérica. Nesse último caso, se faz necessário a colocação de um ponto e vírgula após a ⟨variável⟩, assim como também no caso do uso do comando X. Nos outros casos, o ponto e vírgula é opcional. Os espaços dentro do string são ignorados.
Um exemplo do uso de variáveis dentro de um dos comandos de movimento seria:

```
10 SCREEN2
20 X1=40:X2=50
30 DRAW"M+=x1;,-=x2;"
```

O comando X poderá ser muito útil na instrução DRAW já que permitirá definir uma parte do objeto a ser desenhado, separado do objeto total, como assim também pode ser usado para desenhar um string de comandos com mais de 255 caracteres.

PROGRAMAS EXEMPLO

```
10 SCREEN2
20 L=50:PSET(220,191),7
30 DRAW"U190"
40 FORI=189TO1STEP-4
50 A$="L"+STR$(I)+"D"+STR$(I-1)+"R"+STR$(I-2)+"U"+STR$(I-3)
60 DRAW"XA$;"
70 NEXTI
80 GOTO80
```

```
10 SCREEN2
20 A$="U40G20F20"
30 DRAW"BM100,170M120,90S4"
40 DRAW"A0"+A$
50 DRAW"A1"+A$
60 DRAW"A2"+A$
70 DRAW"A3"+A$
80 DRAW"A0"
90 GOTO90
```

DSKF

TIPO — Função.
FORMATO — **DSKF** (⟨Nº de acionador de disco⟩)
EXEMPLO — PRINT **DSKF** (1)
USO — Fornece a quantidade de espaço livre em um disco, no acionador de disco especificado.

PROGRAMA EXEMPLO

```
10 PRINTDSKF(1)
20 END
```

NOTA: "DSKF" só poderá ser utilizado se o computador estiver carregado com HB-DOS e os acionadores de disco 1 ou 2.

END

TIPO Instrução.
FORMATO **END**
EXEMPLO **END**
USO Para terminar a execução de um programa, fechar todos os arquivos e retornar ao nível de comandos.
A instrução **END** pode ser colocada em qualquer lugar do programa a fim de terminar a execução do mesmo. A diferença da instrução STOP, **END** não provoca a impressão da mensagem PARE!.
A instrução **END** no fim de um programa é opcional.

PROGRAMA EXEMPLO

```
10 PRINT"**VAMOS RODAR O PROGRAMA?**"
20 END
30 PRINT"ESSA LINHA NÃO SERA EXECUTADA"
```

EOF

TIPO Função.
FORMATO **EOF** (⟨número do arquivo⟩)
EXEMPLO IF**EOF**(1) THEN CLOSE1 : END
USO Assume o valor —1 (verdadeiro) quando o fim de um arquivo seqüencial é alcançado. Em caso contrário volta ao valor 0.
Usa-se a função **EOF** como teste de fim de arquivo, durante a introdução (INPUT) de valores no mesmo, a fim de não provocar um erro do tipo "FIM DO ARQUIVO".

PROGRAMA EXEMPLO

```
10 MAXFILES=1
20 OPEN"CAS:DEMO"FOR OUTPUT AS #1
30 IF EOF(1)=-1 GOTO70
40 INPUT#1,A$
50 PRINTA$
60 GOTO30
70 CLOSE#1
80 END
```

ERL/ERR

TIPO — Variável do sistema.

FORMATO
ERL
ERR

EXEMPLO
L = **ERL**
E = **ERR**

USO
Ao ser introduzida no programa uma subrotina de tratamentos de erro, indicada por ON ERROR GO TO:
A variável **ERR** assume o código do erro que aparecer durante a execução do programa, e a variável **ERL** assume o valor do número da linha na qual o erro foi detectado. As variáveis **ERR** e **ERL** são normalmente utilizadas em instruções IF...THEN a fim de orientar o caminho a seguir pelo programa no caso de ocorrência de um erro. Quando a instrução que motiva o erro for uma instrução do modo direto, ERL assumirá o valor 65535. Para testar se um erro acontece no modo direto, usa-se a instrução direta

IF 65535 = ERL THEN ...

Caso contrário usa-se:

IF ERL = ⟨ número de linha ⟩ THEN ...
IF ERR = ⟨ código de erro ⟩ THEN ...

Por serem ERL e ERR variáveis reservadas, nenhuma das duas poderá aparecer à esquerda do sinal de igual (=) em uma instrução LET.

REFERÊNCIAS ON ERROR GO TO

PROGRAMA EXEMPLO

```
10 ON ERROR GOTO 50
20 A=25:PRINTA
30 B=A/0
40 END
50 PRINT"ERRO NA LINHA="; ERL
60 PRINT"CODIGO DE ERRO=";ERR
70 RESUME NEXT
```

ERASE

TIPO — Instrução.

FORMATO
ERASE ⟨ lista de variáveis arranjo ⟩

EXEMPLO
ERASE C, D

USO
Para eliminar arranjos de um programa.
Os arranjos poderão ser redimensionados após o uso do **ERASE**. O espaço de memória previamente destinado à localização do arranjo poderá ser usado para outros propósitos.
No caso de intentar redimensionar um "arranjo" sem o uso prévio do **ERASE**, aparecerá a mensagem de erro: 'DIM' REDEFINIDO.

PROGRAMA EXEMPLO

```
10 DIMA(15)
20 DIMB(13)
30 ERASEA,B
40 DIMA(80)
50 DIMB(20)
60 END
```

ERROR

TIPO Instrução.

FORMATO **ERROR** ⟨ código do erro ⟩

EXEMPLO ERROR 200.

USO Para simular a ocorrência de um erro ou para permitir ao usuário definir códigos de erro.

O valor do ⟨ código do erro ⟩ deverá estar entre 0 e 255. Se o valor do ⟨ código de erro ⟩ for igual a um código de erro já utilizado pelo BASIC, a instrução **ERROR** simulará a ocorrência desse erro e a mensagem de erro correspondente aparecerá no monitor.

Para definir seu próprio código de erro, utilize um valor maior que qualquer um dos utilizados pela BASIC nos seus códigos de erro. Vide lista de códigos de erro e mensagens de erro. (É preferível escolher os maiores valores disponíveis, de forma que a compatibilidade seja mantida o dia que forem anexados mais códigos de erro ao BASIC.)

Esse código de erro definido pelo usuário poderá ser convenientemente manipulado por uma rotina de tratamento de erros.

EXEMPLO

```
10 ON ERROR GO TO 1000
   :
120 IF A$ = "Y" THEN ERROR 250
   :
1000 IF ERR = 250 THEN PRINT "TEM CERTEZA?"
   :
```

No caso de uma instrução **ERROR** especificar um código para o qual não existe mensagem definida, o BASIC responderá com a mensagem "ERRO INDEFINIDO". A execução de uma instrução **ERROR** para a qual não existe rotina de tratamento provocará a aparição da mensagem "ERRO INDEFINIDO", após o qual a execução deter-se-á.

PROGRAMA EXEMPLO

```
10 INPUT"Nº DE ERRO=";A
20 ERROR A
```

EXP

TIPO — Função.
FORMATO — **EXP** (⟨ expressão ⟩)
EXEMPLO — **E = EXP**(1)
USO — Fornece o valor de e elevado à potência dada pela ⟨ expressão ⟩
⟨ expressão ⟩ deverá ser ⟨ = 145.06286085862
Se EXP passar do limite aparecerá a mensagem de erro "OVERFLOW".

PROGRAMA EXEMPLO

```
10 PRINT"    X    EXP(X)    LOG(EXP(X))"
20 FORI=.1TO5.5 STEP .1
30 PRINTUSING"#.# ###.###";I;EXP(I);
40 PRINTLOG(EXP(I))
50 NEXT
```

FIX

TIPO — Função.
FORMATO — **FIX** (⟨ expressão ⟩)
EXEMPLO — **F = FIX** (B/3)
USO — Fornece a parte inteira da ⟨ expressão ⟩ (fração truncada)
FIX(X) é equivalente a SGN(X)*INT(ABS(X)). A principal diferença entre **FIX** e INT é que no caso da ⟨ expressão ⟩ ser < 0 (negativo), **FIX** fornece o valor do número imediato superior, e INT fornece o imediato inferior.
EX. — fix (–9.5) = – 9
int (–9.5) = –10

PROGRAMA EXEMPLO

```
10 FORA=-1.2345TO2.2345STEP.5
20 PRINTUSING "##.####";A;
30 B=FIX(A)
40 PRINTUSING "##";B;
50 B=INT(A)
60 PRINTUSING "##";B
70 NEXT
```

FIELD

TIPO Instrução.

FORMATO **FIELD** [#]⟨X⟩,⟨Y⟩ AS ⟨Y$⟩[,⟨Z⟩ AS ⟨A$⟩...]

EXEMPLO **FIELD** # 1, 2 AS N1$

USO Dividir o buffer para um arquivo aleatório no disco.

Antes de poder utilizar uma instrução "GET" ou "PUT" em um programa deve-se executar uma instrução "**FIELD**" ...

Antes de poder utilizar uma instrução "**FIELD**", o arquivo deverá ter sido aberto.

⟨X⟩ é o número sob o qual o arquivo tem sido aberto na instrução OPEN.

⟨Y⟩ e ⟨Z⟩ indicam o número de caracteres de cada elemento no buffer.

⟨Y$⟩ e ⟨Z$⟩ correspondem aos nomes dos elementos no buffer.

O número total de caracteres no buffer não pode passar de 256.

A instrução **FIELD** não introduz informação nenhuma no buffer. Para esse fim serão utilizadas as instruções "LSET" e "RSET".

Por cada arquivo aleatório podem ser utilizadas diversas instruções **FIELD** a fim de redividir o buffer.

PROGRAMA EXEMPLO

```
10 MAXFILES=1
20 OPEN"A:TESTE"AS#1
30 FIELD #1,2 AS N1$,4 AS N2$,8 AS N3$,20 AS N4$
40 INPUT"A%";A%
50 INPUT"B!";B!
60 INPUT"C#";C#
70 INPUT"D$";D$
80 RSETN1$=MKI$(A%):RSETN2$=MKS$(B!):RSETN3$=MKD$(C#)
90 LSETN4$=D$
100 PUT#1,1
110 A%=0:B!=0:C#=0:D$=""
120 PRINTA%;B!;C#;D$
130 GET#1,1
140 A%=CVI(N1$):B!=CVS(N2$):C#=CVD(N3$):D$=N4$
150 PRINTA%;B!;C#;D$
160 CLOSE#1
170 END
```

NOTA "FIELD" só poderá ser utilizado quando o computador estiver carregado com o HB-DOS e os acionadores de disco.

FILES

TIPO — Comando.

FORMATO — **FILES** ["⟨ dispositivo ⟩ : ⟨ nome do arquivo ⟩"]

EXEMPLO — FILES"A : DEMO".

USO — Fornecer o diretório de um disco.

O dispositivo pode ser:

A = acionador do disco 1

B = acionador do disco 2

Se for omitido o ⟨ dispositivo ⟩ e o ⟨ nome do arquivo ⟩, todos os arquivos do disco colocado no acionador atualmente selecionado serão projetados na tela do televisor.

Se vier indicado o ⟨ dispositivo ⟩ e o ⟨ nome do arquivo ⟩, o nome do arquivo será projetado na tela do televisor, se ele for achado no disco. Caso contrário, aparecerá a mensagem "ARQ. NÃO EXISTE".

NOTA: O comando "FILES" só poderá ser utilizado quando o computador estiver carregado com o HB-DOS e os acionadores de disco 1 ou 2.

FOR-NEXT

TIPO Instrução.

FORMATO **FOR** ⟨ variável ⟩ = ⟨ X ⟩ **TO** ⟨ Y ⟩ [STEP Z]
NEXT [[⟨ variável ⟩] [, ⟨ variável ⟩]]

EXEMPLO
```
FOR J = 0 TO 100 STEP 2
NEXT J
```

NOTA: ⟨ variável ⟩ poderá ser inteira, simples precisão ou dupla precisão. x, y e z são expressões numéricas.

USO

As instruções **FOR** e **NEXT** permitem que uma série de instruções sejam realizadas dentro de um "loop" um determinado número de vezes. A ⟨ variável ⟩ é utilizada como um contador. A primeira expressão numérica (x) será o valor inicial do contador. A segunda expressão numérica (y) será o valor final do contador. As linhas do programa que seguem à instrução **FOR** são executadas até que a instrução **NEXT** seja encontrada. Nesse ponto o contador é incrementado no valor dado por STEP. O sistema faz um teste para verificar se o valor do contador é maior que o valor final (y). Se não for o BASIC desvia o programa novamente até a linha seguinte à instrução **FOR** e o processo todo é repetido. Se for maior, a execução do programa continua com a linha seguinte à instrução **NEXT**. Isto é o que se chama de "loop". No caso do STEP não estar especificado, o BASIC assume um incremento igual a 1. Quando o STEP é negativo, o valor final do contador é fixado num valor menor que o inicial. O contador é decrementado em cada passagem do "loop", e o "loop" é executado até que o contador assuma uma valor inferior ao valor final. O roteiro do "loop" é executado só uma vez, se o valor inicial do mesmo multiplicado pelo sinal do STEP é maior que o valor final multiplicado pelo sinal do STEP. Os "loops" podem estar aninhados, isto é, um "loop" **FOR-NEXT** estar contido dentro de um outro "loop". Quando os "loops" estão aninhados, cada "loop" deverá ter um único nome de variável como seu contador. A instrução **NEXT** do "loop" interno, deverá aparecer antes que a do externo.

Exemplo dos "loops" aninhados.

```
10 FOR I = 0 TO ...
20 FOR J = 1 TO ...
      ⋮
60 NEXT J
70 NEXT I.
```

No caso de "loops" aninhados terem o mesmo ponto final, uma única instrução NEXT poderá ser usada para todos eles.
Exemplo:

```
10 FOR I = 0 TO ...
20 FOR J = 1 TO ...
      ⋮
60 NEXT J, I
```

Esse aninhamento de "loops" **FOR-NEXT** só está limitado pela memória disponível.
As variáveis podem ser omitidas na instrução **NEXT**, em cujo caso cada **NEXT** se corresponderá com a instrução **FOR** mais próxima.
Exemplo:

```
10 FOR I = 0 TO...
20 FOR J = 1 TO...
      ⋮
60 NEXT
70 NEXT
```

(continua)

(continuação)

Se uma instrução **NEXT** for encontrada antes da sua correspondente instrução **FOR**, aparecerá a mensagem de erro **"NEXT" SEM "FOR"** e a execução será detida.

PROGRAMA EXEMPLO

```
10 FORI=0TO100
20 PRINTI;
30 NEXT
40 FORI=0TO10STEP5
50 FORJ=ITOI+4
60 PRINTUSING"###";J;
70 NEXT:PRINT
80 NEXT
90 END
```

FRE

TIPO Função.
FORMATO FRE (⟨ argumento ⟩)
EXEMPLO FRE (0)
FRE (" ")

NOTA: Os argumentos da função **FRE** são pseudo-argumentos que podem ter qualquer valor.

USO **FRE** fornece o número de bytes da memória que não estão sendo utilizados pelo BASIC.
FRE (0) fornece o número de bytes na memória que estão disponíveis para serem usados pelo programa BASIC, arquivo de texto, arquivo de programa em linguagem de máquina etc. **FRE** (" ") fornece o número de bytes na memória disponíveis para espaços de "string".

PROGRAMA EXEMPLO

```
10 WIDTH31
20 PRINT"Vamos reduzir o numero de bytes";
22 PRINT"disponivel atualmente,que é de:"
24 PRINTFRE(0):PRINT
30 FORI=200TO1000STEP200
40 DIMA(I)
50 PRINT"Com DIM A(";I;") ficam ";FRE(0);"bytes livres"
60 ERASE A:NEXT
70 PRINT"O limite de espaços livres para";
72 PRINT" strings é achado com FRE(string)"
74 PRINT: PRINT"PRESSIONE UMA TECLA QUALQUER!"
80 I$=INPUT$(1):I$=""
90 FORJ=1TO5
100 A$=A$+"ABCD":PRINTA$;"..";
110 PRINT FRE("");"BYTES LIVRES"
120 NEXT
```

GET

TIPO Instrução.
FORMATO **GET** [#]⟨X⟩[,⟨Y⟩]
EXEMPLO **GET** # 1, 1
USO Ler no buffer um dado de um arquivo aleatório gravado em um disco.
⟨X⟩ é o número sob o qual tem sido aberto o arquivo através da instrução "OPEN".
⟨Y⟩ é o número do dado que deverá ser lido. Deverá ser um inteiro compreendido entre Ø e 32767. Se ⟨Y⟩ não vier indicado, será lido o dado sucessivo, ou seja o dado gravado imediatamente a seguir do último dado lido ou escrito.
O buffer na memória deverá ter sido reservado com a instrução "FIELD".
Para escrever sobre um disco um dado de um arquivo aleatório, existente no buffer, usa-se a instrução "PUT"
REFERÊNCIAS "FIELD".

NOTA: "GET" só poderá ser utilizado quando o computador estiver carregado com o HB-DOS e os acionadores de disco 1 ou 2.

GOSUB - RETURN

TIPO — Instrução.

FORMATO — **GOSUB** ⟨ número de linha ⟩
RETURN [⟨ número de linha ⟩]

EXEMPLO
```
10 GOSUB 100
100 RETURN.
```

USO — O **GOSUB** provoca o desvio do programa para uma sub-rotina que começa na linha indicada no ⟨ número de linha ⟩. O **RETURN** indica a saída da sub-rotina e retorno à instrução seguinte ao **GOSUB** mais recente.
A sub-rotina poderá ser chamada inúmeras vezes dentro do programa, e também pode ser chamada de dentro de outra sub-rotina. Esse aninhamento de sub-rotinas só está limitado pela memória disponível.
Uma sub-rotina poderá conter mais do que uma instrução **RETURN**, se a lógica do programa indicar que devem existir saídas da sub-rotina em diversos pontos da mesma.
As sub-rotinas poderão aparecer em qualquer lugar do programa, mas é conveniente que elas sejam facilmente identificáveis.
Para evitar entradas inadvertidas nas sub-rotinas, elas podem vir precedidas por uma instrução **STOP** ou **END** ou **GOTO**. Caso contrário, a entrada na sub-rotina provocará um erro do tipo "**RETURN**" **SEM** "**GOSUB**" e a execução do programa será detida.

PROGRAMA EXEMPLO

```
10 GOSUB90
20 PRINT"LINHA Nº 20"
30 GOSUB90
40 PRINT"LINHA Nº 40"
50 I=1:GOSUB90
60 PRINT"LINHA Nº 60"
70 PRINT"END"
80 END
90 PRINT"I=";I
100 PRINT"SUB-ROTINA"
110 IFI=1THENRETURN70
120 RETURN
```

GOTO

TIPO Instrução.
FORMATO 1) **GOTO** ⟨ número de linha ⟩
2) **GOTO** ⟨ número de linha ⟩
EXEMPLO **GOTO 10Ø**
USO Provocar um desvio incondicional, fora da seqüência normal do programa, para uma linha indicada em ⟨ número de linha ⟩. . No caso de ser a linha indicada por ⟨ número de linha ⟩, uma instrução executável, essa instrução e as seguintes serão executadas. Se for uma instrução não-executável, a execução procede até achar a primeira instrução executável.

PROGRAMA EXEMPLO

```
10 GOTO40
20 PRINT"deste tipo.":END
30 PRINT"Você não ";:GOTO60
40 PRINT"Isto é ";:GOTO70
50 PRINT"exemplo.":GOTO80
60 PRINT"pode fazer ";:GOTO90
70 PRINT"um mau ";:GOTO50
80 PRINT:GOTO30
90 PRINT"programas ";:GOTO20
```

HEX$

TIPO Função.
FORMATO **HEX$** (⟨ expressão ⟩)
EXEMPLO PRINT **HEX$** (13)
USO Fornece um "string" que representa o valor hexadecimal do argumento decimal.
⟨ expressão ⟩ é um valor numérico compreendido entre –32768 e 65535. No caso da ⟨ expressão ⟩ ser negativa, será utilizada a forma complemento de dois. Isto é: **HEX$** (–n) é igual a **HEX$** (65535–n).

PROGRAMA EXEMPLO

```
10 PRINT"Vamos contar em base hexadecimal"
20 PRINT"BASE 16    BASE10"
30 FORI=0TO300
40 PRINTHEX$(I);"          ";I
50 FORT=0TO300-I:NEXT
60 NEXT
70 FORI=0TO10
80 PRINT" .           ."
90 NEXT
100 PRINT"FFFF        65535"
```

TIPO Instrução.

FORMATO 1) **IF** ⟨ expressão ⟩ **THEN** ⟨ instrução ⟩ | ⟨ número de linha ⟩
[**ELSE** ⟨ instrução ⟩ | ⟨ número de linha ⟩]
2) **IF** ⟨ expressão ⟩ **GOTO** ⟨ número de linha ⟩
[**ELSE** ⟨ instrução ⟩ | ⟨ número de linha ⟩]

EXEMPLO **IF** A$ = "Y" **THEN** 200 **ELSE** 120

USO Tomar uma decisão no que se refere ao fluxo do programa, baseando-se no resultado fornecido por uma expressão.

Se o resultado da expressão não for zero, a cláusula **THEN** ou a **GOTO** são executadas. **THEN** poderá estar seguido por um número de linha a fim de realizar um desvio ou por uma ou mais instruções a serem executadas. **GOTO** é sempre seguido por um número de linha.

Se o resultado da expressão for zero, as cláusulas **THEN** e **GOTO** são ignoradas, e a cláusula **ELSE** (se ela existir) será executada.

A execução continua na primeira instrução executável.

Exemplo:

A = 1 : B = 2 → A = B é zero (FALSO)
A = 2 : B = 2 → A = B não é zero (VERDADEIRO).

As instruções **IF... THEN... ELSE** podem estar aninhadas. O aninhamento só está limitado pelo comprimento da linha. Se a instrução não contém o mesmo número de cláusulas **ELSE** e **THEN**, cada **ELSE** se corresponderá com o THEN desacompanhado mais próximo.

Exemplo

IF A = B **THEN** IF B = C **THEN** PRINT "A = C"
ELSE PRINT "A ⟨ ⟩ C"

Não imprimirá "A ⟨ ⟩ C" sendo A ⟨⟩ B — Imprimirá "A ⟨⟩ C" quando A = B e B ⟨⟩ C.

Se uma instrução **IF...THEN** estiver seguida por um número de linha no modo direto, provocará um erro do tipo "número de linha indefinido", a menos que a instrução com o número de linha especificado tenha sido introduzido anteriormente no modo indireto.

PROGRAMA EXEMPLO

```
10 CLS:PRINT"***ACERTE NA MOSCA!***"
20 PRINT:DEFINTI,N,S
30 M=10000
40 PRINT"VOCÊ CONTA COM=";M
50 PRINT" DE $ ";M;"QUANTO VOCÊ APOSTA?";:INPUTK
60 IFK<0THEN PRINT"O QUE!":GOTO50
70 IFK>MTHEN PRINT"UEH!NÃO TEM TUDO ISSO!":GOTO50
80 IFK=MTHEN PRINT"NOSSA!"ELSEIFK>M/2THENPRINT"SERÁ QUE
   PODE TUDO ISSO?"ELSEIFK<
M/100THEN PRINT"MESQUINHO!"
90 INPUT"ACERTE NA MOSCA ENTRE (1-6)";N
100 IFN<1ORN>6THENPRINT"NÃO PODE SER!":GOTO90
110 S=RND(-TIME)*6+1
120 PRINT"A MOSCA ERA=";S
130 IF N=STHENM=M+K*3:PRINT"ACERTOU!!"ELSE PRINT"ERROU!
   ":M=M-K
140 IFM<1THENPRINT"VOCÊ FALIU!SINTO MUITO!"ELSEIFM>10000
   00!THENPRINT"VOCÊ FICOU
RICO!PARABÉNS!"ELSE50
```

INKEY$

TIPO Função.

FORMATO INKEY$

EXEMPLO A$ = INKEY$

USO Fornece um string de um caractere só. Esse caractere será introduzido pelo usuário através do teclado. No caso de não pressionar tecla alguma, a função assumirá o valor de string nulo.
Nenhum caractere será repetido, e qualquer caractere passará pelo programa, com exceção de CONTROL-C, que provocam a saída do programa.

Exemplo: 100 A$ = INKEY$: IF A$ = " " THEN 1ØØ.
Nesse exemplo a variável A$ assumirá o valor da tecla introduzida pelo teclado. No caso de não pressionar tecla alguma, o programa continuará esperando a introdução de um caractere sem sair da linha 1ØØ.

PROGRAMA EXEMPLO

(Para fazer desenhos na tela, movimentando o cursor com as teclas: ←, ↑, ↓, →).

```
10 SCREEN2:X=100:Y=100
20 IFINKEY$<>""THEN20'key buffer clear'
30 I$=INKEY$
40 IFI$=""THEN30
50 A=ASC(I$)
60 IFA<28ORA>31THEN30
70 ONA-27GOTO80,90,100,110
80 X=X+1:IFX>255THENX=1:GOTO 120ELSE120
90 X=X-1:IFX<1THENX=255:GOTO120ELSE120
100 Y=Y-1:IFY<1THENY=191:GOTO120ELSE120
110 Y=Y+1:IFY>191THENY=1
120 PSET(X,Y)
130 GOTO20
```

INP

TIPO Função.
FORMATO **INP** (⟨ número de "port" ⟩)
EXEMPLO A = **INP** (15).
USO Fornece o byte lido através do "port" dado pelo ⟨ número de port ⟩. O ⟨ número de port ⟩ deverá estar compreendido entre 0 e 255.
INP é a função complementária da instrução OUT.
OBSERVAÇÃO: Essa função acessa de forma direta os "port" I/O da máquina. Por isso, não pode-se garantir que os programas que a utilizam, sejam compatíveis com os sistemas futuros.

PROGRAMA EXEMPLO

```
10 CLS
20 PRINT"PRESSIONAR AS TECLAS ESPAÇO,HOME,INS,DELETE E CURSOR"
30 OUT170,(INP(170)AND&HF0)OR8
40 LOCATE10,10:PRINTRIGHT$("00000000"+BIN$(INP(169)),8):GOTO30
```

INPUT

TIPO Instrução.

FORMATO **INPUT** ["⟨ string mensagem ⟩ ";] ⟨ lista de variáveis ⟩

EXEMPLO **INPUT** "DATA";A$

USO Permite introduzir dados através do teclado durante a execução do teclado.

Ao encontrar uma instrução INPUT, a execução do programa para e aparece um ponto de interrogação na tela a fim de indicar que o programa está esperando a introdução de dados. Quando a instrução inclui um "string mensagem", ela virá impressa antes do ponto de interrogação. O dado requisitado será então introduzido através do teclado.

Os dados introduzidos são atribuídos às variáveis listadas na ⟨ lista de variáveis ⟩. O número de itens de dados fornecidos deverá ser igual ao número de variáveis da lista. Os itens de dados são separados por vírgulas.

Os nomes da ⟨ lista de variáveis ⟩ poderão ser de variáveis numéricas ou de "strings" (inclusive variáveis indexadas). O tipo de cada item de dado introduzido deverá concordar com o tipo especificado pelo nome da variável. (Os "strings" introduzidos como entrada para a instrução INPUT não precisarão estar entre aspas.)

No caso de responder à instrução **INPUT** com o tipo errado de valor (por exemplo: string no lugar de valor numérico etc.), aparecerá a mensagem de erro "? DIGITE DE NOVO". A atribuição do valor introduzido não será realizada até que uma resposta aceitável seja dada.

Exemplo:

```
LIST
  10 INPUT "A + B"; A, B
  20 PRINT A + B
  OK
RUN
  A e B? 1Ø, mØ (O 1Ø e o mØ foram digitados pelo usuário).
  ? DIGITE DE NOVO
  A e B? 1Ø, 2Ø  (O 1Ø e o 2Ø foram digitados pelo usuário).
   30
   OK
```

Respondendo à instrução **INPUT** com maior número de itens do que o solicitado, aparecerá a mensagem "?EXTRA ANULADO", e a próxima instrução será executada.

Exemplo:

```
LIST
  10 INPUT "A + B"; A, B
  20 PRINT A+B
  OK
RUN
  A+B? 1Ø, 2Ø, 3Ø  (Os 1Ø, 2Ø e 3Ø foram digitados pelo usuário).
  ? EXTRA ANULADO
  30
  OK
```

Respondendo à instrução **INPUT** com menor número de itens do que o solicitado, aparecerão 2 pontos de interrogação e ficará esperando a introdução de mais dados.

Exemplo:

(continua)

(continuação)

```
LIST
  10 INPUT "A + B"; A, B
  20 PRINT A + B
  OK
RUN
  A + B? 10      (O 10 foi digitado pelo usuário).
  ?? 20          (O 20 foi digitado pelo usuário).
  30
  OK
```

Para escapar da instrução **INPUT**, deve-se digitar CONTROL-C ou então as teclas "CTRL" e "STOP" juntas. O BASIC voltará ao nível de comando e aparecerá a mensagem "OK".

Digitando CONT o BASIC continua com a execução da instrução INPUT.

PROGRAMA EXEMPLO

```
10 CLS:PRINT"A instrução INPUT serve para";
20 PRINT" introduzir letras e algarismos";
30 PRINT" atraves do teclado"
40 INPUT"ENTENDEU(S/N)";Y$
50 Y$=LEFT$(Y$,1)
60 IFY$="S"ORY$="s"THEN110
70 PRINT"Então experimente introduzir coisas pelo teclado"
80 PRINT"Não esqueça de pressionar a teola RETURN apos cada
   entrada"
90 INPUT"A$,B=";A$,B
100 PRINT"A$=";A$,"B=";B:GOTO40
110 PRINT"Que bom!"
```

INPUT#

TIPO Instrução.
FORMATO **INPUT** # ⟨ número de arquivo ⟩, ⟨ lista de variáveis ⟩.
EXEMPLO **INPUT** # 1, A, B
USO Para ler itens de dados provenientes do canal especificado, e atribui-los a variáveis do programa.
Os tipos dos dados do arquivo devem concordar com os tipos especificados na ⟨ lista de variáveis ⟩. A instrução **INPUT**# não provoca a aparição do ponto de interrogação (como acontece com a instrução **INPUT**).
Os itens de dados do arquivo deverão aparecer na mesma forma que apareceriam se estivessem sendo introduzidos pelo teclado como resposta a uma instrução **INPUT**.
Quando se trata de valores numéricos, os espaços, retornos de carro e alimentação de linha (*line-feed*) iniciais são ignorados. O primeiro caractere encontrado, que não for um espaço, um retorno de carro ou uma alimentação de linha será considerado o começo de um número. O número termina com um espaço, com um retorno de carro, uma alimentação de linha ou uma vírgula.
Se o BASIC estiver procurando dados para um item "string", também os espaços, retornos de carro e alimentação de linha iniciais serão ignorados. O primeiro caractere que for encontrado e que não for um desses três mencionados, será considerado como o início do "string". Se esse primeiro caractere for aspas ("), o item "string" estará composto por todos os caracteres, compreendidos entre o primeiro e o segundo par de aspas. Isto é, um item entre aspas não poderá incluir aspas como um dos seus caracteres.
Quando o primeiro caractere do string não for aspas, o string será um string sem aspas, e terminará com uma vírgula, um retorno de carro, uma alimentação de linha ou após 255 caracteres terem sido lidos. Se o fim do arquivo for alcançado enquanto um item numérico ou "string" estiver sendo introduzido, o item será terminado.

INPUT$

TIPO Função.
FORMATO **INPUT$** (⟨ X ⟩ [, [#] ⟨ código do arquivo ⟩])
EXEMPLO **INPUT$** (6, # 2)
USO Fornece um string de ⟨ X ⟩ caracteres, lidos através de teclado ou provenientes de um arquivo. Nenhum caractere será repetido, e todos os caracteres do teclado serão aceitos com exceção de CONTROL-C, que parará a execução da função **INPUT$**.

PROGRAMA EXEMPLO

```
10 'A SENHA E "DIG-1"+ A TECLA ESC
20 P$="DIG-1"+CHR$(27)
30 PRINT"SENHA?"
40 A$=INPUT$(6):PRINT
50 IFA$<>P$THENPRINT"VOCE NÃO E O MEU USUARIO!!":GOTO30
60 BEEP:PRINT"SEJA BEMVINDO AO MUNDO DO DIG-1"
70 END
```

INSTR

TIPO Função.
FORMATO **INSTR** ([I,] X$, Y$)
EXEMPLO **INSTR** (A$, "J")
USO Procura a primeira aparição do string Y$ dentro do string X$ e fornece o valor da posição na qual essa coincidência de strings acontece.
[I] representa um deslocamento opcional que fixa a posição de início da busca. I deverá ser um número compreendido entre 0 e 255. No caso de ser I > LEN(X$), ou ser X$ nulo, ou se Y$ não for achado em X$, ou se X$ e Y$ forem nulos INSTR assumirá o valor 0. No caso de ser somente Y$ nulo, **INSTR**, assumirá o valor 1.
X$ e Y$ pode ser variáveis "string", expresssões "string" ou "string" de letras.

PROGRAMA EXEMPLO

```
10 '***PIANO***
20 PRINT"Pressione uma tecla maiusoula qualquer"
30 PRINT"C=Do,F=Do#,V=Re,G=Re#..."
40 TB$="AZSXCFVGBNJMK,L/:Q2WE4R5T6YU8I9OP-@^[\"
50 PLAY"L8"
60 IN$=INKEY$:IFIN$=""THENGOTO60
70 I=INSTR(TB$,IN$)
80 IFI>0THENI=I+32
90 PLAY"N=I;"
100 GOTO60
```

INT

TIPO Função.
FORMATO **INT** (⟨expressão⟩)
EXEMPLO PRINT **INT** (1.132)

USO Fornece o valor do maior inteiro menor ou igual que a ⟨expressão⟩.

PROGRAMA EXEMPLO

```
10 PRINT"   X","INT(X)"
20 PRINT
30 FORI=-2.4TO2.4
40 PRINTI,INT(I)
50 NEXT
60 END
```

INTERVAL ON/OFF/STOP

TIPO Instrução.

FORMATO
INTERVAL ON
INTERVAL OFF
INTERVAL STOP

USO Ativação, desativação ou suspensão do controle sobre um intervalo de tempo transcorrido. (Vide instrução ON INTERVAL GOSUB.)

A instrução **ON INTERVAL** (X) **GOSUB** (n) faz com que o BASIC se desvie para a subrotina definida pelo número de linha (n) cada vez que se completar o intervalo de tempo determinado por (x).

A instrução **INTERVAL ON** ativa a verificação desse intervalo pelo BASIC. Cada vez que o intervalo de tempo se completar, a execução do programa será interrompida, sendo executada a subrotina iniciada na linha (n).

A instrução **INTERVAL OFF** desativa a verificação do intervalo de tempo. Através dela, o BASIC ignorará a sucessão dos intervalos definidos por ON INTERVAL GOSUB, e não haverá interrupção na execução do programa.

A instrução **INTERVAL STOP** faz com que o BASIC continue com a verificação do transcurso do intervalo de tempo, mas libera o programa de seguir para a subrotina (n). No caso de ter passado o intervalo de tempo, e aparecer uma nova instrução **INTERVAL ON**, o programa seguirá imediatamente para a subrotina indicada por **ON INTERVAL GOSUB.**

PROGRAMA EXEMPLO

```
10 ONINTERVAL=3000GOSUB60
20 INTERVALON
30 FORI=0TO10000:NEXT
40 INTERVALOFF
50 END
60 K=K+5:PRINTK;"SEGUNDOS"
70 RETURN
```

KEY

TIPO Comando.

FORMATO **KEY** ⟨ número da tecla de função ⟩, ⟨ expressão "string" ⟩

EXEMPLO **KEY** 1, "BRASIL".

USO Definir um "string" de caracteres para uma determinada tecla de função. O ⟨ número de tecla ⟩ deverá ser um número inteiro compreendido entre 1 e 10.

A ⟨ expressão "string" ⟩ poderá ter um máximo de 15 caracteres.

EXEMPLO

```
KEY 1 , "CLS" + CHR$ (13)      Define a tecla 1 como CLS + RETURN

KEY 2 , "BRASIL"               Define a tecla 2 como BRASIL
```

KEY LIST

TIPO Comando.
FORMATO **KEY LIST**
EXEMPLO **KEY LIST**
USO Para obter uma listagem dos conteúdos da 10 teclas de função.
EXEMPLO (Condição inicial)

```
KEY LIST
color                F1
auto                 F2
goto                 F3
list                 F4
run                  F5
color 15, 4, 4       SHIFT + F1
cload"               SHIFT + F2
cont                 SHIFT + F3
list.                SHIFT + F4
run                  SHIFT + F5
```

O comando "color" se corresponde com a tecla "F1", "auto" com "F2", "goto" com "F3" e assim por diante.
Note-se que os caracteres de controle atribuídos a uma tecla de função são convertidos em espaços.

KEY
ON/OFF/STOP

TIPO Instrução.

FORMATO
KEY (⟨ número de tecla de função ⟩) ON
KEY (⟨ número de tecla de função ⟩) OFF
KEY (⟨ número de tecla de função ⟩) STOP

EXEMPLO
KEY (1) **ON**

USO
Ativa, desativa ou suspende a interrupção provocada por uma tecla de função. (Vide instrução **ON KEY GOSUB**.)
O ⟨ número de tecla de função ⟩ deverá ser um número inteiro compreendido entre 1 e 10.
Após a instrução **KEY** (n) **ON**, o BASIC verifica em cada instrução se tem sido pressionada a tecla de função indicada. Em caso afirmativo, o programa segue para a subrotina indicada pela instrução **ON KEY GOSUB**.
Após a instrução **KEY** (n) **OFF** o BASIC deixa de verificar se foi pressionada a tecla de função indicada.
Após a instrução **KEY** (n) **STOP** o BASIC continua verificando se dita tecla tem sido pressionada, mas não segue para a subrotina indicada na instrução **ON KEY GOSUB**. Ainda, ele lembra o evento, e em tal caso, seguirá para sub-rotina imediatamente após o aparecimento de um novo **KEY ON**.

PROGRAMA EXEMPLO

```
10 CLS:LOCATE5,5:PRINT"F1=SUB-ROTINA 1"
20 LOCATE5,7:PRINT"F3=SUB-ROTINA 2"
30 LOCATE5,9:PRINT"F5=END"
40 ONKEYGOSUB70,,90,,110
50 KEY(1)ON:KEY(3)ON:KEY(5)ON
60 GOTO60
70 LOCATE10,11:PRINT"SUB-ROTINA 1"
80 RETURN
90 LOCATE10,13:PRINT"SUB-ROTINA 2"
100 RETURN
110 LOCATE10,15:PRINT"END"
120 END
```

KEY ON/ OFF

TIPO — Instrução.
FORMATO — **KEY ON**
KEY OFF
EXEMPLO — **KEY ON**
KEY OFF
USO — Ativar ou desativar a visualização das teclas de função na 24ª linha da tela, nos modos texto 1 e texto 2.
Com a instrução **KEY OFF** os textos das teclas de função não aparecerão mais na tela.
A instrução **KEY ON** fará com que eles voltem a aparecer.

PROGRAMA EXEMPLO

```
10 FORI=1TO5
20 KEYOFF
30 FORJ=1TO200:NEXT
40 KEYON
50 FORJ=1TO300:NEXT
60 NEXTI
70 END
```

KILL

TIPO — Instrução.
FORMATO — **KILL** "⟨ dispositivo ⟩ : ⟨ nome arquivo ⟩"
EXEMPLO — **KILL** "A : DEMO"
USO — Apagar um arquivo de um disco.
O ⟨ dispositivo ⟩ poderá ser:
A = acionador de disco nº 1.
B = acionador de disco nº 2.
⟨ dispositivo ⟩ refere-se ao acionador de disco no qual tem sido inserido o disco que contém o arquivo a ser apagado.

NOTA: "KILL" só poderá ser utilizado quando o computador estiver carregado com o HB-DOS e os acionadores de disco 1 e 2.

LEFT$

TIPO	Função.
FORMATO	LEFT$ (⟨"string"⟩,⟨n⟩)
EXEMPLO	B$ = LEFT$ (A$, 4)
USO	Fornece uma expressão alfanumérica composta dos ⟨n⟩ caracteres situados no extremo esquerdo da ⟨string⟩. ⟨n⟩ é um número inteiro de 0 ~ 255. Quando ⟨n⟩ for maior que o número total de caracteres de ⟨string⟩, virá indicado o conteúdo completo da ⟨string⟩. Quando ⟨n⟩ for igual a zero, a função fornecerá uma expressão alfanumérica nula.

PROGRAMA EXEMPLO

```
10 A$="BASIC"
20 FORI=1TOLEN(A$)
30 PRINTLEFT$(A$,I)
40 NEXT
50 END
```

LEN

TIPO	Função.
FORMATO	LEN (⟨"string"⟩)
EXEMPLO	PRINT LEN (A$)
USO	Fornece o número de caracteres existentes em ⟨"string"⟩. São contados inclusive os espaços e os códigos dos caracteres do 1 ao 31.

PROGRAMA EXEMPLO

```
10 INPUT"Palavra";A$
20 PRINTA$," tem";LEN(A$);" caracteres"
30 L=LEN(A$)
40 PRINT:PRINTSTRING$(L+4,42)
50 PRINT"* ";A$;" *"
60 FORI=1TOL+4
70 PRINT"*";
80 NEXT
90 END
```

LET

TIPO	Instrução.
FORMATO	[**LET**] ⟨ variável ⟩ = ⟨ expressão ⟩
EXEMPLO	**LET** A = 10
	A$ = "20" + "Cr"
USO	Atribuir o valor de uma expressão a uma variável.
	NOTA: A palavra **LET** é opcional.

PROGRAMA EXEMPLO

```
10 'A PALAVRA LET E OPCIONAL
20 LETA=10
30 LETB=20
40 LETC$="Tudo bem?"
50 PRINTA;B;C$
60 A=10:B=20:C$="Tudo bem?"
70 PRINTA;B;C$
```

LINE

TIPO Instrução.

FORMATO LINE [|(x1, y1) / STEP (x1, y1)|] – [|(x2, y2) / STEP (x2, y2)|] [, ⟨ código de cor ⟩] [, | B / BF |]

EXEMPLO LINE (100, 100) – (135, 100), 8

USO Traçar uma linha nos modos gráficos (alta-resolução e multi-color) (SCREEN 2 ou SCREEN 3).

X_1 é a coordenada X do ponto de início da linha e deverá ser um número inteiro entre 0 e 255.

Y_1 é a coordenada Y do ponto de início da linha e deverá ser um número inteiro entre 0 e 191.

Quando se usa a palavra **STEP**, os valores X_1 e Y_1 são interpretados tomando como referência a posição do cursor. Nesse caso X_1 e Y_1 poderão inclusive ser números inteiros negativos.

Se X_1 ou Y_1 ou ambos forem omitidos, serão utilizadas as coordenadas correntes.

X_2 e Y_2 são as coordenadas X e Y do ponto final da linha. Também nesse caso as coordenadas serão relativas à última posição do cursor quando vierem precedidas da palavra "**STEP**".

⟨ código de cor ⟩ refere-se ao número da cor na qual a linha deverá ser traçada. Deverá ser um número inteiro entre 0 e 15, segundo a seguinte tabela.

0 – transparente
1 – preto
2 – verde
3 – verde claro
4 – azul escuro
5 – azul claro
6 – vermelho escuro
7 – ciano
8 – vermelho médio
9 – vermelho claro
10 – amarelo escuro
11 – amarelo claro
12 – verde escuro
13 – magenta
14 – cinza
15 – branco

Quando não vier especificada cor alguma, será utilizada a última cor usada anteriormente para desenho. O valor "default" para a cor é 15.

Quando vier escrita a letra B no fim da instrução, será desenhado um retângulo, sendo a linha descrita a sua diagonal.

Quando após a letra B houver uma letra F, o retângulo será pintado na cor especificada.

PROGRAMA EXEMPLO

```
10 SCREEN2
20 FORK=1TO15STEP7
30 FORI=0TO95STEP3
40 LINE(128-I,95-I)-(128+I,95+I),7,B
50 NEXT
60 FORJ=0TO400:NEXT
70 FORK=1TO15STEP7
80 FORX=KTO255STEP5
90 LINE(X,0)-(255-X,191),7
100 NEXTX
110 FORY=191-KTO0STEP-5
120 LINE(0,Y)-(255,191-Y),4
130 NEXTY
140 NEXTK
150 GOTO150
```

LINE INPUT

TIPO Instrução.
FORMATO **LINE INPUT** [⟨ "mensagem string" ⟩ ;] ⟨ variável string ⟩
EXEMPLO **LINE INPUT** "NOME" ; NA$
USO Para atribuir a uma variável "string" uma linha inteira de até 254 caracteres, introduzida através do teclado.
A "mensagem string" aparecerá impressa na tela antes da introdução ser aceita. O ponto de interrogação não aparecerá a menos que esteja incluído na "mensagem". Todas as entradas realizadas entre o fim da mensagem e o retorno de carro, serão atribuídas à ⟨ variável string ⟩. Para escapar do **LINE INPUT**, deve-se pressionar CONTROL + C ou "CTRL" e "STOP" simultaneamente. O BASIC retorna ao nível de comandos e aparece "OK". Digitando CONT continua com a execução na instrução **LINE INPUT.**

PROGRAMA EXEMPLO

```
10 CLS:PRINT"LINE INPUT permite introduzir aspas (";
20 PRINT CHR$(34);") e virgulas (";CHR$(44);")"
30 PRINT:PRINT"Introduza agora uma linha completa"
40 LINEINPUTA$
50 PRINT"E isso mesmo!"
60 PRINT"A$=";A$
```

LIST

TIPO Comando.
FORMATO **LIST** [⟨ número de linha ⟩ [— [⟨ número de linha ⟩]]]
EXEMPLO **LIST** 100 — 200
LIST 500 —
USO Para listar todo um programa ou parte dele.
Se ambos ⟨ números de linha ⟩ forem omitidos, o programa será listado completo, começando pelo número de linha mais baixo.
Se vier indicado somente o primeiro número de linha, só essa linha será listada.
Se vier indicado o primeiro número de linha seguido de "—", será listada essa linha e todas as que tiverem numeração superior.
Se vier indicado "—" seguido do segundo número de linha, serão listadas todas as linhas do programa desde o início até o número indicado.
Se ambos números forem indicados, será listada a parte do programa compreendida entre ambas linhas.
Para terminar a listagem deve-se pressionar "CRTL" e "STOP" simultaneamente. No caso de querer deter a listagem por um tempo, deve-se pressionar a tecla "STOP". Para continuar com a mesma, pressione "STOP" novamente.
Exemplos:
LIST 10
(Visualiza a linha nº 1/)
LIST 10–50
(Visualiza as linhas do programa compreendidas entre 10 e 50)
LIST—100
(Visualiza todas as linhas, entre a primeira e a número 100)
LIST
(Visualiza todas as linhas do programa).

TIPO Comando.
FORMATO **LLIST** [⟨ número de linha ⟩[— [⟨ número de linha ⟩]]]
EXEMPLO **LLIST**
USO Esse comando serve para listar um programa completo ou parte dele através de uma impressora.

(NOTA: Esse comando funciona do mesmo modo que o comando LIST).

LOAD

TIPO Comando.
FORMATO **LOAD** "⟨ dispositivo periférico ⟩ : ⟨ nome do arquivo ⟩" [,R]
EXEMPLO **LOAD** "CAS : DEMO".
USO Para carregar na memória do microcomputador um programa BASIC arquivado em fita cassette ou em disco flexível.
O comando **LOAD** fecha todos os arquivos abertos e apaga o programa que tiver na memória. Todavia, com a opção "R", os arquivos de dados permanecerão abertos e o programa carregado será imediatamente executado.
Se o nome do arquivo for omitido, o primeiro programa encontrado na fita (que deverá ser um arquivo gravado em formato ASCII através do comando SAVE), será carregado.
Control—Z é tratado como fim de arquivo.
REFERÊNCIAS CLOAD, SAVE, MERGE.

NOTA: "LOAD CAS" só poderá ser utilizado quando o computador estiver conectado a um gravador.

"LOAD A" ou "LOAD B"' só poderão ser utilizados quando o computador estiver carregado com o HB-DOS e os acionadores de disco 1 ou 2.

LOC

TIPO — Função.
FORMATO — **LOC** (< nº de arquivo >)
EXEMPLO — PRINT **LOC** (I)
USO — Fornece a localização atual no arquivo indicado.
Antes de usar a função "**LOC**", o arquivo deverá ter sido aberto.
< nº de arquivo > é o número sob a qual o arquivo foi aberto através da instrução "OPEN".
Esta função fornece o número do último dado lido ou escrito no arquivo aleatório. Nos arquivos seqüenciais, a função fornece o número de setores (blocos de 256 bytes) lidos ou escritos no arquivo a partir do momento em que foi aberto.

PROGRAMA EXEMPLO

```
10 MAXFILES=1
20 OPEN"A:TESTE"AS #1
30 FIELD#1,2ASN1$,4ASN2$,8ASN3$,20ASN4$
40 GET#1,1
50 PRINTLOC(1)
60 CLOSE#1
```

NOTA: "LOC" só poderá ser utilizado quando o computador estiver carregado com o HB-DOS e os acionadores de disco 1 ou 2.

LOCATE

TIPO	Instrução.
FORMATO	LOCATE [⟨X⟩] [,⟨Y⟩] [,⟨Z⟩]
EXEMPLO	LOCATE 10, 10
USO	Enviar o cursor a uma posição dada da tela, nos modos texto 1 e 2 a fim de posicionar os caracteres dados em uma instrução PRINT. ⟨X⟩ é a coordenada X e deverá ser um número inteiro entre 0 e 39. Quando o ⟨X⟩ não vier indicado, será utilizada a coordenada X da posição atual do cursor. ⟨Y⟩ é a coordenada Y e deverá ser um número inteiro entre o 0 e o 23. Quando ⟨Y⟩ não vier indicado, será utilizada a coordenada Y da posição atual do cursor. ⟨Z⟩ é o interruptor da visualização do cursor e deverá ser 0 ou 1. Zero significa que o cursor deverá ser visível na tela enquanto que 1 significa que deverá ser invisível. Quando ⟨Z⟩ não vier especificado, será assumindo o último valor utilizado por ⟨Z⟩. O valor standard de Z é zero.

PROGRAMA EXEMPLO

```
10 CLS
20 FORI=0TO30
30 X=INT(RND(1)*28)
40 Y=INT(RND(1)*23)
50 LOCATEX,Y:PRINT"AQUI";
60 LOCATE10,23:PRINTUSING"LOCATE## ###";X;Y;
70 FORJ=0TO700:NEXT
80 LOCATEX,Y:PRINTSPACE$(12);
90 NEXT
100 END
```

LOF

TIPO Função.
FORMATO **LOF** (⟨ X ⟩)
EXEMPLO PRINT **LOF** (I)
USO Fornece o comprimento de um determinado arquivo em função do número de bytes.
Antes de poder utilizar "**LOF**", o arquivo indicado deverá ter sido aberto ⟨ X ⟩ é o número sob o qual o arquivo tem sido aberto, através da instrução "OPEN".

PROGRAMA EXEMPLO

```
10 MAXFILES=1
20 OPEN"A:TESTE"FORINPUTAS#1
30 PRINTLOF(1)
40 CLOSE#1
50 END
```

NOTA: "**LOF**" só poderá ser utilizado quando o computador estiver carregado com o HB-DOS e os acionadores de disco 1 ou 2.

LOG

TIPO Função.
FORMATO **LOG** (⟨ expressão ⟩)
EXEMPLO PRINT **LOG** (35/9).
USO Fornece o logaritmo natural da ⟨ expressão ⟩. A ⟨ expressão ⟩ deverá ser maior que zero.

PROGRAMA EXEMPLO

```
10 SCREEN2
20 FORI=.1TO25STEP.1
30 Y=LOG(I)/3.2*100
40 LINE(0,115)-(255,115),7
50 PSET(20+X,115-Y),7
60 X=X+1
70 NEXT
80 GOTO80
```

LPOS

TIPO Variável do sistema.
FORMATO LPOS (⟨ X ⟩)
EXEMPLO LPOS (Ø)
USO Fornece a posição atual da cabeça da impressora dentro do buffer da impressora. Essa posição não necessariamente coincidirá com a posição física da cabeça da impressora. X é um pseudo-argumento que não tem valor real.

PROGRAMA EXEMPLO

```
10 FORI=1TO1000
20 LPRINT"Oi,tudo bem?";
30 IF LPOS(0)>38 THEN LPRINT CHR$(13)
40 NEXT
50 END
```

NOTA: Esse comando só poderá ser utilizado se o computador estiver ligado a uma impressora).

LPRINT

TIPO Instrução.
FORMATO **LPRINT** [⟨ lista de expressões ⟩]
LPRINT USING ⟨ expressão string ⟩; ⟨ lista de expressões ⟩
EXEMPLO **LPRINT** A$, B
LPRINT USING "& ###.##"; A$; B
USO Serve para imprimir dados com uma impressora. Funciona do mesmo modo que a instrução PRINT. (Para maiores detalhes vide PRINT.)

NOTA: **LPRINT** só poderá ser utilizada se o computador estiver ligado a uma impressora).

PROGRAMA EXEMPLO

```
10 PRINT"Introduza uma linha qualquer"
20 LINEINPUTA$
30 LPRINTA$
```

LSET

TIPO Instrução.
FORMATO **LSET** ⟨ X$ ⟩ = ⟨ Y$ ⟩
EXEMPLO **LSET** N4$ = D$
USO Preencher, começando da esquerda, a variável ⟨ X$ ⟩ com o conteúdo da variável ⟨ Y$ ⟩ ou com a expressão alfanumérica ⟨ Y$ ⟩.
Usam-se as instruções **"LSET"** ou "RSET" para inserir os dados no buffer para um arquivo aleatório em disco.
Os dados numéricos podem ser convertidos em alfanuméricos através das funções "MKI$", "MKS$", e "MKD$" quando são inseridos no buffer para um arquivo aleatório.
Os dados numéricos deverão ter sido convertidos de alfanuméricos em numéricos com as funções "CVI", "CVS" e "CVD" quando são tiradas do buffer para um arquivo aleatório.
REFERÊNCIAS "FIELD"

NOTA: "LSET" só poderá ser utilizado quando o computador estiver carregado com o HB-DOS e os acionadores de disco 1 ou 2.

MAXFILES

TIPO Instrução.
FORMATO **MAXFILES** = ⟨ expressão ⟩
EXEMPLO **MAXFILES** = 3
USO Especificar o máximo número de arquivos abertos ao mesmo tempo. ⟨ expressão ⟩ indica o número de arquivos e deverá estar entre 0 e 15. Quando for executado **'MAXFILES = 0"**, somente SAVE e LOAD poderão ser utilizados.
O valor "default" é 1.

PROGRAMA EXEMPLO.

```
10 MAXFILES=2
20 OPEN"CAS:DEMO"FORINPUT AS#1
30 OPEN"LPT:"FOROUTPUTAS#2
40 INPUT#1,A$
50 PRINT#2,A$
60 CLOSE
70 END
```

MERGE

TIPO Comando.
FORMATO MERGE "⟨ dispositivo periférico ⟩ : ⟨ nome do programa ⟩"
EXEMPLO MERGE "CAS : TESTE".
USO Anexar as linhas de um arquivo de programa ASCII ao programa que está atualmente na memória.
Os dispositivos ⟩ podem ser:
CAS = gravador
A = acionador de disco 1
B = acionador de disco 2
Se algumas linhas do arquivo que está sendo introduzido, tiverem os mesmos números que as linhas do programa na memória, as linhas do arquivo substituirão as linhas correspondentes da memória.
Após o comando MERGE, o programa aumentado permanece na memória e o BASIC volta ao nível de comando.
Se o ⟨ nome do programa ⟩ for omitido, o primeiro arquivo encontrado (deverá ser um arquivo ASCII gravado através do SAVE) será anexado.
Control–Z será tratado como fim de arquivo.

NOTA: "MERGE CAS" só poderá ser utilizado se o computador estiver conectado a um gravador.

"MERGE A" ou "MERGE B" só poderão ser utilizados se o computador estiver carregado com o HB-DOS e os acionadores de disco 1 ou 2.

MID$

TIPO Função.
FORMATO MID$ (X$, I [, ⟨ J ⟩])
EXEMPLO PRINT MID$ (A$, 2, 3)
USO Fornece uma expressão alfanumérica composta de J caracteres, começando pela posição I de X$.
I e J deverão estar compreendidos entre 1 e 255. Se J for omitido ou se houver menos do que J caracteres à direita do I-ésimo caractere de X$, todos os caracteres da direita começando pelo J-ésimo, serão incluídos.
Se I ⟩ LEN(X$), MID$ devolverá um string nulo.

PROGRAMA EXEMPLO

```
10 A$="Branca de neves e os sete anões"
20 L=LEN(A$):N=1
30 FORI=1TOL
40 S=INSTR(N+1,A$," ")
50 IFS=0THENS=L+1
60 PRINTMID$(A$,N,S-N)
70 I=S:N=S:NEXT
80 INPUT"Qual é a estória do seu agrado";A$
90 A$=" "+A$:GOTO20
```

MID$

TIPO Instrução.
FORMATO MID$ (⟨ expressão string 1 ⟩), n [, m]) = ⟨ expressão string 2 ⟩
EXEMPLO MID $ (A$, 3) = "ABC"
USO Para substituir uma parte de uma variável alfanumérica com uma outra variável alfanumérica ou com uma constante numérica.
Os caracteres da ⟨ expressão string 1 ⟩, começando a partir da posição n, são substituídos pelos caracteres da ⟨ expressão string 2 ⟩.
O [,m] opcional refere-se ao número de caracteres do ⟨ string 2 ⟩ que serão utilizados na substituição.
Se o m for omitido, serão utilizados todos os caracteres do ⟨ string 2 ⟩, sempre limitando-se ao comprimento do string original "1".

PROGRAMA EXEMPLO

```
10 A$="ABCDEFG"
20 B$="****"
30 PRINTA$,B$
40 PRINT
50 FORI=1TO7
60 C$=A$
70 MID$(C$,I,1)=B$
80 PRINTC$,I
90 NEXT
```

MKI$
MKS$
MKD$

TIPO Função.
FORMATO MKI$ (⟨ X ⟩)
MKS$ (⟨ Y ⟩)
MKD$ (⟨ Z ⟩)
EXEMPLO N1$ = MKI$ (A%)
USO Converter valores numéricos em alfanuméricos.
Os valores numéricos que são inseridos no buffer para um arquivo aleatório no disco, devem ser convertidos para a forma alfanumérica.
"MKI$" converte um inteiro em uma variável alfanumérica de 2 bytes.
"MKS$" converte uma variável de precisão simples em uma variável alfanumérica de 4 bytes.
"MKD$" converte uma variável de precisão dupla em uma variável alfanumérica de 8 bytes.
REFERÊNCIAS "FIELD"

> NOTA: "MKI$", "MKS$" e "MKD$" só poderão ser utilizados quando o computador estiver carregado com o HB-DOS e os acionadores de disco 1 ou 2.

MOTOR ON/OFF

TIPO — Instrução.
FORMATO — **MOTOR [ON]**
MOTOR [OFF]
EXEMPLO — **MOTOR**
USO — Modificar o estado da chave interruptora do motor do gravador. Quando o argumento não vier indicado, o estado da chave, mudará, qualquer que ele seja. Isto é, se estiver ligado, desligará e viceversa. Caso contrário, habilitará ou desabilitará o motor.

PROGRAMA EXEMPLO

```
10 'ISTO N&O E MUITO UTIL
20 FORI=1TO100
30 MOTOR
40 FORJ=0TO10
50 NEXTJ,I
60 END
```

NAME

TIPO — Instrução.
FORMATO — **NAME** "⟨ dispositivo ⟩ : ⟨ nome arquivo-1 ⟩" AS "⟨ nome-arquivo 2 ⟩".
EXEMPLO — **NAME** "CAS : TESTE 1" AS " TESTE 2".
USO — Mudar o nome de um arquivo no disco.
O ⟨ dispositivo ⟩ pode ser:
CAS = gravador
A = acionador de disco 1
B = acionador de disco 2
⟨ nome-arquivo 1 ⟩ é o nome original, enquanto.
⟨ nome-arquivo 2 ⟩ é o novo nome do arquivo.

NOTA: **"NAME"** só poderá ser utilizado quando o computador estiver carregado com o HB-DOS e os acionadores de disco 1 ou 2.

NEW

TIPO Comando.
FORMATO NEW
EXEMPLO NEW
USO Apagar da memória de trabalho um programa completo. Ele "reseta" também todas as variáveis.
Esse comando é utilizado normalmente antes de iniciar um novo programa a fim de cancelar o anterior.

PROGRAMA EXEMPLO

```
10 'ESTE PROGRAMA JA RODOU,VAMOS APAGA-LO
20 A$="T C H A U !"
30 FORI=1TO10
40 BEEP
50 PRINTMID$(A$,I,1)
60 FORJ=0TO100:NEXT
70 NEXT
80 NEW
```

OCT$

TIPO Função.
FORMATO OCT$ (⟨ expressão ⟩)
EXEMPLO LPRINT OCT$ (1Ø)
USO Fornece uma representação octal da ⟨expressão⟩ decimal dada com o argumento.
O resultado dessa função é um valor alfanumérico.
⟨ expressão ⟩ é um valor numérico compreendido entre −32768 e 65535.
Se n for negativo, será utilizada a forma do complemento a dois. Isto é, OCT$(−n) = OCT$ (65536−n).

PROGRAMA EXEMPLO

```
10 I=100
20 PRINT"   X    OCT$(X)    HEX$(X)"
30 PRINTI;" ";OCT$(I);"         ";HEX$(I)
40 END
```

TIPO	Instrução.
FORMATO	ON ERROR GO TO ⟨ número de linha ⟩
EXEMPLO	ON ERROR GO TO 1000
USO	Habilitar uma rotina de tratamento de erros e especificar o número da primeira linha dessa rotina. Uma vez que a rotina foi habilitada, todos os erros detectados, incluindo os do modo direto (por exemplo: erros de sintaxe), provocarão um desvio para a rotina de tratamento dos mesmos. Se o ⟨ número de linha ⟩ não existir, resultará num erro do tipo "Nº LINHA INEXISTENTE". Para desabilitar a rotina de tratamento de erros, deve-se executar: **ON ERROR GOTO Ø**. Os erros subseqüentes provocarão a visualização de uma mensagem de erro e uma detenção na execução. Se houver dentro da rotina de tratamento de erros, uma instrução ON ERROR GOTO Ø, isto fará com que o BASIC pare e imprima a mensagem de erro que motivou a entrada na rotina. Para retornar ao programa principal após a execução da sub-rotina de erros, deve-se colocar a instrução "RESUME". É conveniente que todas as sub-rotinas de tratamento de erro executem a instrução **ON ERROR GOTO Ø** no caso de encontrar um erro para o qual não existe procedimento de recuperação. Se um erro acontecer durante a execução de uma sub-rotina de tratamento de erros, aparecerá a mensagem de erro do BASIC e a execução terminará. A habilitação da sub-rotina não acontece quando o erro está dentro da mesma.
REFERÊNCIAS	"ERL", "ERR", "ERROR".

PROGRAMA EXEMPLO

```
10 'EXERCICIO DE DETECCAO DE ERRO
20 ONERRORGOTO80
30 INPUT"Introduzir um numero de 1 a 999";A
40 PRINT"Meu numero e:";A
50 IFA<1ORA>999THENERROR26
60 GOTO30
70 '***SUB-ROTINA DE ERROS***
80 IFERR<>26THEN120
90 PRINT"Favor introduzir numero correto":PRINT
100 PRINT"A linha na qual apareceu o erro foi ";ERL:PRINT
110 RESUME NEXT
120 ONERRORGOTO0
```

ON - GOTO/ ON - GOSUB

TIPO Instrução.

FORMATO **ON** ⟨ expressão ⟩ **GOTO** ⟨ lista de números de linhas ⟩
ON ⟨ expressão ⟩ **GOSUB** ⟨ lista de números de linhas ⟩

EXEMPLO **ON** A **GOSUB** 100, 200, 300, 400
ON A **GOTO** 100, 200, 300

USO Provocar um desvio para uma das diversas linhas especificadas em ⟨ lista de números de linhas ⟩, dependendo do valor fornecido pelo resultado da ⟨ expressão ⟩. O valor dessa ⟨ expressão ⟩ determina qual é o número de linha da ⟨ lista ⟩ que será utilizado no desvio. Por exemplo, se o valor for 3, o terceiro número da lista indicará a linha destinatária do desvio. Se o resultado não for um número inteiro, a porção fracionária será desconsiderada.

Na instrução **ON...GOSUB** cada número de linha dentro da ⟨ lista de números ⟩ deverá ser a primeira linha de uma subrotina.

Se o valor da ⟨ expressão ⟩ for zero ou maior que o número de itens da lista (embora menor ou igual a 255), o BASIC seguirá para a próxima instrução executável. Se o valor da ⟨ expressão ⟩ for negativo ou maior que 255, aparecerá a mensagem de erro "FUNÇÃO ILEGAL".

PROGRAMA EXEMPLO

```
10 PRINT"Pressione as teclas 1,2,ou 3"
20 A=VAL(INKEY$)
30 ONAGOTO50,60,70
40 GOTO20
50 PLAY"O4C","E","G":GOTO20
60 PLAY"O4C","F","A":GOTO20
70 PLAY"O3B","D","G":GOTO20
```

ON INTERVAL GOSUB

TIPO Instrução.

FORMATO **ON INTERVAL** = ⟨ intervalo de tempo ⟩ **GOSUB** ⟨ número de linha ⟩

EXEMPLO **ON INTERVAL = 60 GOSUB 100**

USO Indicar o número de linha de início da subrotina para a qual o programa se desvia ao acontecer uma interrupção provocada por um timer incorporado.

A interrupção do timer é gerada cada ⟨ intervalo de tempo⟩ / 60 segundos. Quando a interrupção acontece, um **INTERVAL STOP** automático é executado, de forma que enquanto estiver dentro da subrotina, o BASIC não aceitará novas interrupções. O RETURN da sub-rotina provocará a execução automática de um INTERVAL ON, a menos que um **INTERVAL OFF** tenha sido colocado em forma explícita dentro da subrotina.

A sucessão de interrupções não acontece enquanto o BASIC não está executando o programa.

No caso do BASIC entrar numa subrotina de tratamento de erros (através de uma instrução ON ERROR), automaticamente será executado um **INTERVAL OFF.**

PROGRAMA EXEMPLO

```
10 ONINTERVAL=60GOSUB100
20 INTERVALON
30 SCREEN2,1
34 A$=CHR$(&H18)+CHR$(&H3C)+CHR$(&H66)+CHR$(&HDB)
36 B$=CHR$(&HE7)+CHR$(&H7E)+CHR$(&H24)+CHR$(&H42)
40 SPRITE$(1)=A$+B$
50 GOTO50
100 X=INT(RND(1)*256):Y=INT(RND(1)*192)
110 C=INT(RND(1)*14)+2
120 PUTSPRITE1,(X,Y),C,1
130 RETURN50
```

ON KEY GOSUB

TIPO — Instrução.

FORMATO — **ON KEY GOSUB** ⟨ lista do número de linhas ⟩

EXEMPLO — **ON KEY GOSUB** 100, 200, 300

USO — Indicar qual é a subrotina que deve ser seguida em caso de ser pressionada uma das teclas de função.

Antes de poder seguir para a subrotina da tecla de função, deverá ser ativada a interrupção através da instrução "KEY (X) ON".

Quando a interrupção acontece, um KEY (X) STOP é executado automaticamente, de forma que, enquanto o programa estiver executando a subrotina indicada, não aceitará novas interrupções. O RETURN da subrotina provocará a execução automática de um KEY (X) ON, a menos que um KEY (X) OFF tenha sido colocado em forma explícita dentro da subrotina.

A sucessão de interrupções não acontece enquanto o BASIC não está executando o programa.

No caso do BASIC entrar numa subrotina de tratamento de erros (através de uma instrução ON ERROR) automaticamente será executado um KEY OFF.

PROGRAMA EXEMPLO

```
10 CLS:PRINT"Pressione uma das teclas de função:"
20 ONKEYGOSUB90,100,110,120,130,140,150,160,170,180
30 KEY(1)ON:KEY(2)ON
40 KEY(3)ON:KEY(4)ON
50 KEY(5)ON:KEY(6)ON
60 KEY(7)ON:KEY(8)ON
70 KEY(9)ON:KEY(10)ON
80 GOTO80
90 N=1:GOTO190
100 N=2:GOTO190
110 N=3:GOTO190
120 N=4:GOTO190
130 N=5:GOTO190
140 N=6:GOTO190
150 N=7:GOTO190
160 N=8:GOTO190
170 N=9:GOTO190
180 N=10:GOTO190
190 PRINT"Você pressionou a tecla F##";N
200 RETURN
```

ON SPRITE GOSUB

TIPO — Instrução.

FORMATO — **ON SPRITE GOSUB** ⟨ número de linha ⟩

EXEMPLO — **ON SPRITE GOSUB 100**

USO — Indica a sub-rotina que deverá ser seguida em caso de colisão de 2 sprites. Ao entrar na sub-rotina um SPRITE STOP é executado automaticamente, de forma que o BASIC não aceitará interrupções enquanto estiver dentro dela. O RETURN provocará a execução de um SPRITE ON automaticamente, a menos que um **SPRITE OFF** tenha sido colocado em forma explícita na sub-rotina.

No caso do BASIC entrar numa sub-rotina de tratamento de erros (através de uma instrução ON ERROR), automaticamente será executado um SPRITE OFF.

PROGRAMA EXEMPLO

```
10 DATA60,66,165,129,165,153,66,60
20 DATA60,120,219,255,255,219,102,60
30 A$=""
40 FORI=1TO8
50 READA:A$=A$+CHR$(A)
60 NEXT
70 B$=""
80 FORI=1TO8
90 READA:B$=B$+CHR$(A)
100 NEXT
110 SCREEN2,1:COLOR15,4,1
120 ONSPRITEGOSUB210
130 SPRITE$(0)=A$:SPRITE$(1)=B$
140 SPRITEON
150 A=INT(RND(1)*256)
160 B=INT(RND(1)*256)
170 FORI=0TO191
180 PUTSPRITE0,(A,I),1
190 PUTSPRITE1,(B,191-I),15
200 NEXT:GOTO140
210 SPRITEOFF
220 PLAY"L4CDEFEDCREFGAGFER"
230 IFPLAY(0)THEN230
240 PUTSPRITE0,(0,208)
250 I=191:RETURN
```

ON STOP GOSUB

TIPO Instrução.

FORMATO **ON STOP GOSUB** ⟨ número de linha ⟩

EXEMPLO **ON STOP GOSUB** 1ØØ

USO Indicar a sub-rotina que deverá ser seguida em caso de pressionar simultaneamente as teclas CTRL e STOP.

Ao entrar na sub-rotina, um STOP STOP é executado automaticamente, de forma que o BASIC não aceitará interrupções enquanto estiver dentro dela. O RETURN provocará a execução de um STOP ON automaticamente, a menos que um STOP OFF tenha sido colocado em forma explícita na sub-rotina.

No caso do BASIC entrar numa sub-rotina de tratamento de erros (através de uma instrução ON ERROR), automaticamente será executado um STOP OFF.

O usuário deverá tomar muito cuidado com o uso desta instrução. Por exemplo, o seguinte programa não poderá ser abortado. A única forma seria resetando o sistema.

EXEMPLO

```
10 ON STOP GOSUB 4Ø
20 STOP ON
30 GO TO 3Ø
40 RETURN
```

PROGRAMA EXEMPLO

```
10 ONSTOPGOSUB50
20 STOPON
30 INPUTA$
40 IFA$="END"THENSTOPOFF:ENDELSEGOTO30
50 PRINT"DIGITAR END E PRESSIONAR TECLA RETURN":RETURN
```

ON STRIG GOSUB

TIPO — Instrução.

FORMATO — **ON STRIG GOSUB** ⟨ lista de números de linhas ⟩

EXEMPLO — **ON STRIG GOSUB 100, 200**

USO — Indicar quais são as subrotinas que devem ser seguidas ao serem pressionadas os botões de acionamento de disparos (TRIGGER).

As diversas subrotinas correspondem aos diversos botões (ex.: a primeira subrotina será a correspondente à barra de espaços, a segunda corresponde ao disparador do comando manual Nº 1, etc. (Vide **STRIG ON/STOP/OFF.**)

Ao entrar numa subrotina um STRIG STOP é executado automaticamente de forma que o BASIC não aceitará interrupções enquanto estiver dentro de uma delas. O RETURN provocará a execução de um STRIG ON automaticamente a menos que um STRIG OFF tenha sido colocado em forma explícita na subrotina.

No caso do BASIC entrar numa subrotina de tratamento de erros (através de uma instrução ON ERROR), automaticamente será executado um STRIG OFF.

PROGRAMA EXEMPLO

```
10 CLS:ONSTRIGGOSUB80
20 STRIG(0)ON
30 PRINT"Pressione a tecla de espaço"
40 LOCATE0,20:PRINT"(Para parar este programa"
50 PRINT"aperte as teclas CTRL+STOP)"
60 GOTO60
70 COLOR15,1,1
80 PLAY"T255CEG"
90 LOCATE8,12:PRINT"Acertou!!"
100 FORI=0TO300:NEXTI
110 LOCATE8,12:PRINTSPC(10)
120 COLOR15,4,7
130 RETURN
```

OPEN

TIPO Instrução.

FORMATO **OPEN** " ⟨ descritor do periférico ⟩ : [⟨ nome do arquivo ⟩] "
[FOR ⟨ modo ⟩] AS [#] ⟨ número do arquivo ⟩

EXEMPLO **OPEN** "CAS : TESTE" FOR INPUT AS # 1

USO Colocar um buffer para entrada e saída (I/O), e fixar o modo que será usado com esse buffer.

Esta instrução habilita um buffer para posteriores processamentos.

Os periféricos podem ser:

CAS : gravador (cassete)	LPT : impressora
CRT : tela de texto	A : acionador de disco 1
GRP : tela de gráficos	B : acionador de disco 2

⟨ modo ⟩ refere-se a um dos seguintes:

OUTPUT : especifica o modo de saída seqüencial.
INPUT : especifica o modo de entrada seqüencial.
Se for omitida a cláusula FOR, o BASIC assumirá que o arquivo é aleatório.

Não é possível utilizar os dois modos em todos os periféricos.

As possibilidades são as seguintes:

Periférico	INPUT		OUTPUT
CAS	*		*
CRT			*
GRP			*
LPT			*
A ou B	*	ou	*

O ⟨ número de arquivo ⟩ é uma expressão inteira cujo valor está entre 1 e o número máximo de arquivos, especificado na instrução MAXFILES =

O ⟨ número de arquivo ⟩ é o número que estará associado a esse arquivo enquanto o mesmo estiver aberto (OPEN), e será usado por outras instruções I/O sempre que se referir a esse arquivo.

O OPEN deverá ser executado sempre antes que qualquer instrução de entrada ou saída referente a um arquivo, por exemplo:

PRINT #, PRINT # USING
INPUT #, LINE INPUT #
INPUT $, GET, PUT

PROGRAMA EXEMPLO

```
10 CLS:I=10000
20 PRINT"Vamos escrever um texto em modo gráfico"
30 PRINT:PRINT"Pressione uma teola qualquer"
40 IFINKEY$=""THEN40
50 SCREEN2
60 OPEN"GRP:"FOROUTPUTAS#1
70 LINE(30,10)-(220,170),7,B
80 PRESET(70,50)
90 PRINT#1,"***teste***"
100 PRESET(100,90)
110 PRINT#1,USING"I=#####";I
120 CLOSE#1
130 GOTO130
```

NOTAS: "OPEN LPT" só poderá ser utilizado quando o computador estiver conectado a uma impressora.

"OPEN CAS" só poderá ser utilizado quando o computador estiver conectado a um gravador.

"OPEN A" E "OPEN B" só poderão ser utilizados quando o computador estiver carregado com o HOT-DOS e os acionadores de disco A e B.

OUT

TIPO Instrução.
FORMATO OUT ⟨ número de "port" ⟩ , ⟨ expressão inteira ⟩
EXEMPLO OUT &H22, &HØØ
USO Transmitir um byte para um "port" de saída da máquina.
⟨ número de port ⟩ e ⟨ expressão inteira ⟩ deverão estar entre 0 e 255.
⟨ expressão inteira ⟩ representa o dado (byte) a ser transmitido.
REFERÊNCIAS INP.
PROGRAMA EXEMPLO

```
10 OUT&HA8,INP(&HA8)
20 END
```

PAD

TIPO Função.
FORMATO PAD (⟨ n ⟩)
EXEMPLO X = PAD (1)
USO Fornece o estado de um dos "touch pad" (acessório).
É possível ligar um ou dois "touch pad" ao computador, através dos terminais de conexão dos joysticks.
⟨ n ⟩ é um número inteiro que vai de Ø a 7.
Quando ⟨ n ⟩ tem um valor compreendido entre Ø e 3 presume-se que o "touch pad" está ligado no terminal de conexão para joystick 1.
Quando ⟨ n ⟩ tem um valor compreendido entre 4 e 7, presume-se que o "touch pad" está ligado no terminal de conexão para joystick 2.
Quando ⟨ n ⟩ é Ø ou 4, fornece o estado do "touch pad".
Nesses casos o valor da função é −1 quando o pad foi pressionado e Ø quando não foi ainda pressionado.
Quando ⟨ n ⟩ é 1 ou 5, fornece o valor da coordenada X.
Quando ⟨ n ⟩ é 2 ou 6, fornece o valor da coordenada Y.
Quando ⟨ n ⟩ é 3 ou 7, fornece o estado do interruptor sob o "touch pad".
O resultado dessa função será −1 no caso do interruptor ter sido pressionado e 0 em caso contrário.
PROGRAMA EXEMPLO

```
10 SCREEN2
20 AA=Ø
30 IFPAD(Ø)=ØTHEN2Ø
40 X=PAD(1):Y=PAD(2)
50 IFAA=ØTHENPSET(X,Y)ELSELINE-(X,Y)
60 AA=1
70 GOTO3Ø
```

NOTA: Essa função (PAD) só poderá ser utilzada se tiver ligado um "touch pad" (acessório) ao microcomputador.

PAINT

TIPO Instrução.

FORMATO **PAINT** | (x, y) / STEP (x, y) | [, ⟨ Z ⟩] [, ⟨ xx ⟩]

EXEMPLO **PAINT** (5Ø, 12Ø), 3, 4.

USO Preencher uma figura gráfica com uma cor determinada, nos modos gráficos (SCREEN 2, SCREEN 3).

(X, Y) são as coordenadas de início da pintura. Para maiores detalhes sobre essas coordenadas, vide PUT SPRITE. X deverá estar entre (Ø e 255) e Y entre (Ø e 191). PAINT não aceitará coordenadas que estiverem fora da tela.

Ao ser usada a palavra "STEP", os valores X e Y serão interpretados relativamente à posição do cursor. Nesse caso, X e Y poderão assumir valores negativos.

⟨ Z ⟩ é o número correspondente à cor utilizada para preencher a figura.

⟨ xx ⟩ é o número da cor da linha de contorno da figura.

No modo gráfico de alta resolução (SCREEN 2), a cor utilizada para preencher a figura deverá ser a mesma que a do contorno. Nesse caso, não deverá ser colocado o dado ⟨ xx ⟩.

No modo gráfico multicolor (SCREEN 3), a cor do preenchimento deverá ser diferente da cor da linha de contorno.

⟨ Z ⟩ e ⟨ xx ⟩ deverão ser números inteiros de Ø a 15.

Ø : transparente
1 : preto
2 : verde
3 : verde claro
4 : azul escuro
5 : azul claro
6 : vermelho escuro
7 : ciano
8 : vermelho médio
9 : vermelho claro
10 : amarelo escuro
11 : amarelo claro
12 : verde escuro
13 : magenta
14 : cinza
15 : branco

PROGRAMA EXEMPLO

```
10 SCREEN2:COLOR15,4,7
20 CIRCLE(80,80),20,8
30 PAINT(80,80),8
40 FORI=1TO2000:NEXT
50 SCREEN3:COLOR15,4,7
60 LINE(10,10)-(100,100),8,B
70 PAINT(45,45),2,8
80 FORI=1TO2000:NEXT
90 END
```

PDL

TIPO Função.
FORMATO PDL (⟨ n ⟩)
EXEMPLO PDL (2)
USO Fornece o estado de um dos "paddles" (acessório).
Existe a possibilidade de ligar um ou dois "paddles" para joystick no microcomputador através dos terminais de conexão correspondentes.
A função PDL fornece um valor que vai de Ø a 255.
⟨ n ⟩ é um número inteiro de 1 a 12.
Quando ⟨ n ⟩ vale 1, 3, 5, 7, 9 ou 11, presume-se que o "paddle" está ligado no terminal de conexão para joystick 1 (port 1).
Quando ⟨ n ⟩ vale 2, 4, 6, 8, 1Ø ou 12, presume-se que o "paddle" está ligado no terminal de conexão para joystick 2 (port 2).

PROGRAMA EXEMPLO

```
10 PRINTPDL(1),PDL(2)
20 GOTO10
```

NOTA: A função "PDL" só poderá ser utilizada se tiver ligado um "paddle" (acessório) ao microcomputador.

PEEK

TIPO Função.
FORMATO PEEK (I)
EXEMPLO A = PEEK (&HAØØØ)
USO Fornece o conteúdo de um byte da memória.
Essa função proporciona o valor decimal correspondente ao conteúdo de um endereço de memória, e por isso terá um valor entre Ø e 255.
⟨ I ⟩ é o endereço de memória e deverá ter um valor entre −32768 e 65535.
Se ⟨ I ⟩ for negativo, será utilizada a forma de complemento a dois. Isto é: PEEK (−1) = PEEK (65536−1).
Para colocar um valor específico em um endereço de memória, usa-se a instrução "POKE".

PROGRAMA EXEMPLO

```
10 '***MEMORY DUMP***
20 INPUT"ENDEREÇO=";A
30 IFA<0THENA=A+65536!
40 FORI=ATOA+64STEP8
50 PRINT"    ";RIGHT$("000"+HEX$(I),4);
60 PRINT"(";RIGHT$("000"+STR$(I),5);")";
70 FORJ=ITOI+7
80 PRINTRIGHT$("0"+HEX$(PEEK(J)),2);
90 NEXTJ:PRINT
100 NEXTI:END
```

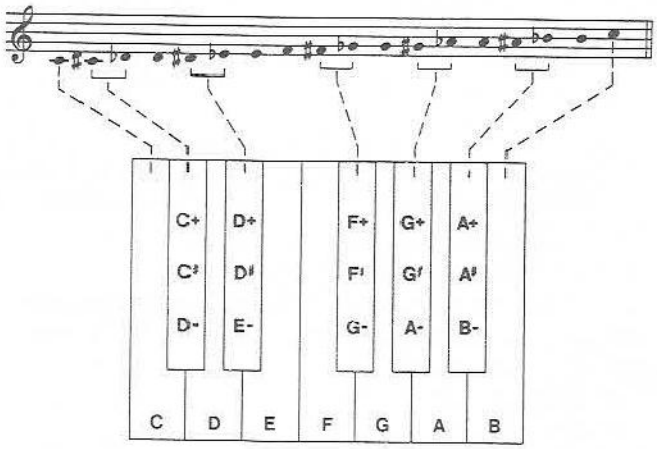
TIPO Instrução.

FORMATO **PLAY** ⟨ expressão "string" nº 1 ⟩
[, ⟨ expressão "string" nº 2 ⟩ [, ⟨ expressão "string" nº 3 ⟩]]

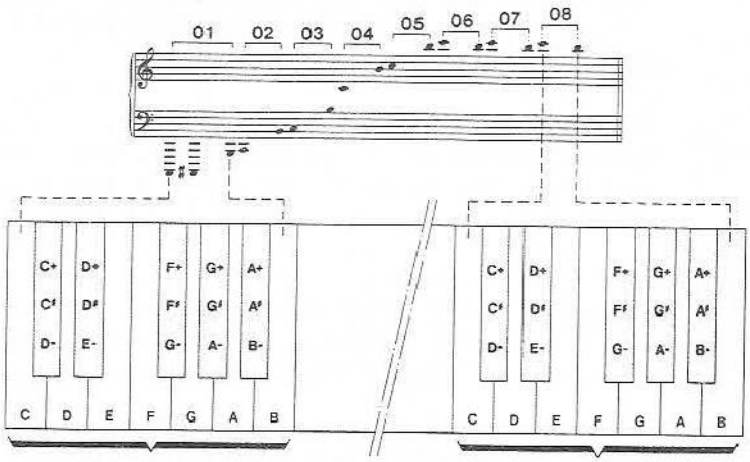
EXEMPLO **PLAY** "CDEFGABO5C"
PLAY A$ + B$, C$

USO Produzir trechos musicais de acordo com a linguagem macro de música. A instrução **PLAY** implementa um conceito similar ao da instrução DRAW, representando uma "linguagem macro de música" através de uma expressão "string".
As ⟨ expressões "string nº n ⟩ são constantes alfanuméricas compostas de sub-comandos, sendo a nº 1 correspondente à primeira voz, a nº 2 correspondente à segunda voz e a nº 3 à terceira.
São possíveis os seguintes sub-comandos:

1. Para produzir as notas da escala musical
As notas LA, SI, DO, RE, MI, FA, SOL correspondem respectivamente as letras A, B, C, D, E, F e G da notação anglo-saxônica.
Os semitons sustenido e bemol são indicados usando-se "+" ou "#" para sustenidos e "–" para bemóis.

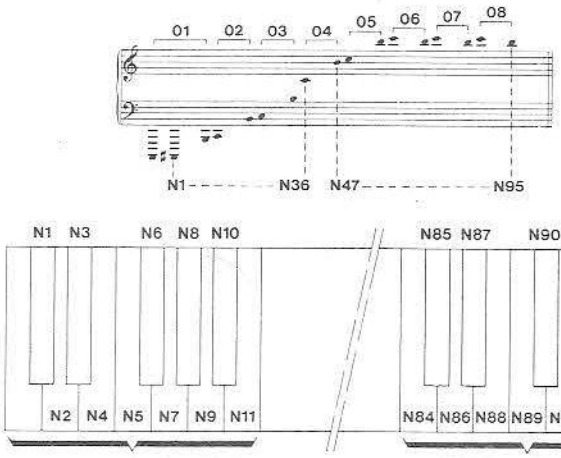


2. Para selecionar oitavas.
O sub-comando "O ⟨ x ⟩" fixa o número da oitava da nota que será tocada a seguir.
⟨ x ⟩ é um número inteiro que vai de 1 a 8.
A ilustração a seguir indica qual é o número usado por cada uma das oitavas.



No caso de não haver indicação explícita de oitava, será utilizada a última oitava definida.
O valor "default" é 4.

3. Para definir as notas através de números.
Em vez de utilizar um número de oitava e um número de nota, pode-se utilizar um número N ⟨ x ⟩.
O sub-comando N ⟨ x ⟩ é utilizado para atribuir um número a cada nota ⟨ x ⟩ é um inteiro que vai de Ø a 95.
Quando ⟨ x ⟩ vale zero, não será tocada nenhuma nota. Nesse caso, se produzirá uma pausa.



4. Para fixar a duração de uma nota.
A duração da nota vem fixada pelo sub-comando "L ⟨ x ⟩".
⟨ x ⟩ é um número inteiro de 1 a 64 e tem o seguinte significado:
1 = Nota inteira.
2 = Meia nota
3 = Um terço de nota
etc.
A duração é de 1/⟨ x ⟩.
A duração pode também seguir a nota no caso de desejar mudar a duração dessa única nota. Por exemplo, A16 é equivalente a LIGA.
O valor "default" é 4.

5. Para mudar a duração de uma nota.
Anexando um ponto imediatamente após à nota dada (.), sua duração será multiplicada por 3/2. Poderão aparecer vários pontos após a nota. Por exemplo "A..." significa que a nota A terá a sua duração multiplicada por 27/8.

6. Para fixar uma pausa.
O sub-comando R ⟨ x ⟩ é utilizado para determinar uma pausa. ⟨ x ⟩ é um inteiro que vai de 1 a 64 e tem o mesmo significado que no sub-comando L ⟨ x ⟩ — ⟨ x ⟩ fixa a duração da pausa.

7. Para fixar o tempo.
O tempo é fixado através do sub-comando "T ⟨ x ⟩".
⟨ x ⟩ é um inteiro que vai de 32 a 255 e indica o número de quartos de nota em um minuto. O "default" é de 12Ø.

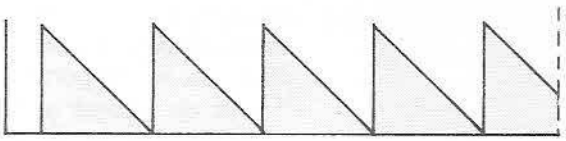
8. Para fixar o volume.
O volume é fixado com o sub-comando 'V ⟨ x ⟩".

⟨ x ⟩ é um número que vai de Ø a 15. Em caso de não ter volume especificado será utilizado o último fixado.
O valor "default" é 8.

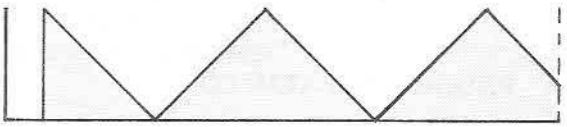
9. Para fixar o perfil da envoltória da onda de som.
O perfil da envoltória da onda de som, é fixado com o sub-comando "S ⟨ x ⟩".
⟨ x ⟩ é um número inteiro que vai de Ø a 15.
Os números representam os seguintes perfis.

4 – 7 e 15

8

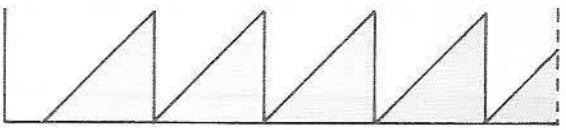


10

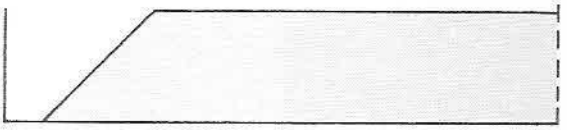


11

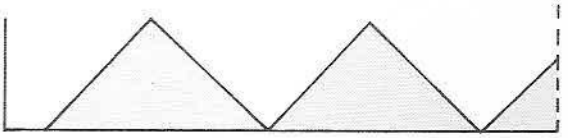
12



13

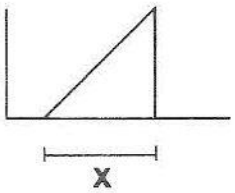


14



Quando não for indicado perfil algum, será utilizado o último definido.
O valor "default" é 1.

10. Modulação do perfil.
O período da envoltória do perfil é determinado pelo sub-comando "M ⟨ x ⟩".
⟨ x ⟩ é um inteiro que vai de 1 a 65535 e fixa o período da envoltória. Quando não vier indicada modulação alguma, será utilizada a última definida.
O valor "default" é 255.



11. Para fixar a execução de uma música.
Através do sub-comando "X ⟨ A$ ⟩;" pode-se tocar um trecho musical indicado pela variável alfanumérica A$.
Não esquecer o ponto e vírgula após o "string"!
Todos os valores numéricos nos sub-comandos podem ser substituídos por variáveis numéricas. Essas variáveis deverão estar precedidas por um sinal de igualdade (=) e terminadas com um ponto e vírgula (;) por exemplo: AB = 1Ø ; PLAY "N = AB ; "
Note-se que os valores especificados nos comandos anteriores, serão "resetados" ao tocar o "beep" do sistema.

PROGRAMA EXEMPLO

```
10 '***BACH***
20 PLAY"T240L6V12","T240L2V9"
30 PLAY"R8O6GABO7DCCED","O4GO5GE"
40 PLAY"DGF#GDO6BGAB","O4BO5EO4E"
50 PLAY"O7CDEDCO6BABG","O4ABO5C"
60 PLAY"F#GADF#AO7CO6BA","O5DF#D"
70 PLAY"BGABO7DCCED","GGC"
80 PLAY"DGF#GDO6BGAB","O4BO5ED"
90 PLAY"EO7DCO6BAGDGF#G2","CC#DG"
```

PLAY

TIPO Função.

FORMATO PLAY (⟨ canal de voz ⟩)

EXEMPLO PLAY (Ø)

USO Fornece informação sobre o estado de uma atividade musical. O ⟨ canal de voz⟩ será um número inteiro entre Ø e 3.

Ø = voz 1, voz 2, e voz 3.
1 = voz 1
2 = voz 2
3 = voz 3.

Essa função assume o valor −1 enquanto a atividade musical, na voz em questão, estiver sendo tocada.

Quando o trecho musical for completado, ela passa a assumir o valor Ø.

Ao ser utilizada essa função imediatamente após a instrução "PLAY", indicará sempre o valor −1.

PROGRAMA EXEMPLO

```
10 A$="":BEEP
20 FORI=1TO6
30 READAA$:A$=A$+AA$
40 NEXT
50 PLAY"XA$;"
60 SCREEN0:CLS
70 IFPLAY(0)=-1THEN PRINT"Estou tocando músioa":GOTO70
80 PRINT"A MUSICA TERMINOU"
90 END
100 DATACCGGAAGR
110 DATAFFEEDDCR
120 DATAGGFFEEDR
130 DATAGGFFEEDR
140 DATACCGGAAGR
150 DATAFFEEDDCR
```

POINT

TIPO Função.
FORMATO **POINT** (x, y)
EXEMPLO **POINT** (5Ø, 5Ø)
USO Fornece o número da cor correspondente a um determinado ponto de imagem (pixel) nos modos gráficos.
〈X〉 é a coordenada X do ponto e deve ser um inteiro entre Ø e 255.
〈Y〉 é a coordenada Y do ponto e deve ser um inteiro entre Ø e 191.

PROGRAMA EXEMPLO

```
10 SCREEN2
20 OPEN"GRP:"FOROUTPUTAS#1
30 FORI=0TO20
40 FORJ=0TO8
50 PRESET(J*24,I*8)
60 C=INT(RND(1)*13)+2
70 COLORC:PRINT#1,"♥"
80 NEXTJ,I
90 COLOR15
100 PRESET(100,170)
110 PRINT#1,"Números das cores"
120 FORI=0TO20
130 FORJ=0TO8
140 K=POINT(J*24+4,I*8+4)
150 PRESET(J*24+8,I*8)
160 PRINT#1,USING"##";K
170 NEXTJ,I
180 GOTO180
```

POKE

TIPO — Instrução.
FORMATO — **POKE** ⟨ endereço de memória ⟩, ⟨ dado ⟩
EXEMPLO — **POKE** &HFØØØ, &HFF
USO — Escrever um byte de dado em uma posição da memória.
⟨ endereço de memória ⟩ é o endereço da posição da memória onde se quer colocar o dado.
⟨ dado ⟩ representa o byte que será colocado na posição indicada. O ⟨ dado ⟩ deverá estar compreendido entre Ø e 255. O endereço de memória deverá estar compreendido entre −32768 e 65535. Se o valor for negativo, o endereço será calculado subtraindo de 65536. Por exemplo, −1 é igual a 65535 (65536−1). Caso contrário dará erro de "overflow".
Para recuperar o valor de um byte colocado em uma determinada posição de memória, usa-se a instrução inversa PEEK.

PROGRAMA EXEMPLO

```
10 CLEAR256,&HE000
20 FORI=&HE001TO&HE010
30 POKEI,256+IMOD256
40 NEXT
50 FORI=&HE000TO&HE020STEP8
60 FORJ=ITOI+7
70 PRINTUSING"&  &";HEX$(PEEK(J));
80 NEXTJ:PRINT
90 NEXTI
```

POS

TIPO — Variável do sistema
FORMATO — **POS** (I)
EXEMPLO — P = **POS** (Ø)
USO — Fornece a posição atual do cursor. A posição extrema esquerda é Ø. (I) é um pseudo-argumento.
O resultado dessa função vai de zero a 39.
A função **POS** só poderá ser utilizada nos modos texto 1 e 2.

REFERÊNCIAS: CSRLIN

PROGRAMA EXEMPLO

```
10 SCREEN0
20 LOCATE12,10
30 PRINTPOS(0)
40 END
```

PRESET

TIPO Instrução.

FORMATO **PRESET** [(x, y) | STEP (x, y)] [, ⟨ código de cor ⟩]

EXEMPLO **PRESET** (100, 100).

USO Definir uma cor específica para um ponto dado da tela, nos modos gráficos.

⟨ X ⟩ é a coordenada X do ponto de imagem, e deverá estar compreendido entre zero e 255.

⟨ Y ⟩ é a coordenada Y do ponto de imagem, e deverá estar compreendido entre zero e 191.

Sendo utilizada a palavra STEP, os valores de X e Y serão relativos à posição do cursor. Nesse caso, ⟨ X ⟩ e ⟨ Y ⟩ poderão inclusive assumir valores negativos.

⟨ código da cor ⟩ define a cor do ponto e deverá ter um valor compreendido entre Ø e 15.

Quando não estiver especificado o ⟨ código de cor ⟩, será utilizado a última cor de fundo anteriormente usada. O valor "default" é 4.

A única diferença entre as instruções PSET e **PRESET**, é que na instrução PRESET, no caso de não especificar a cor, utiliza-se a cor do fundo. Usando o argumento de ⟨ cor ⟩ as duas instruções são idênticas.

PROGRAMA EXEMPLO

```
10 SCREEN2:COLOR15,4,7
20 LINE(40,40)-(215,151),15,BF
30 FORI=0TO1000
40 A=INT(RND(1)*173)+41
50 B=INT(RND(1)*109)+41
60 PRESET(A,B)
70 NEXTI
80 END
```

PRINT

TIPO Instrução.

FORMATO PRINT [⟨ lista de expressões ⟩]

EXEMPLO
```
PRINT "ABC"
? "CDE"
```

USO Saída de dados através da tela.

Se a ⟨ lista de expressões ⟩ não foi inclusa será impressa uma linha em branco. Se a ⟨ lista de expressões ⟩ estiver incluída, os valores das expressões serão impressos na tela. As expressões da lista poderão ser numéricas ou alfanuméricas. As alfanuméricas deverão vir escritas entre aspas. A posição de cada item impresso está determinada pela pontuação usada na separação dos itens dentro da lista. O BASIC divide cada linha em zonas de impressão de 14 espaços cada. Dentro da ⟨ lista de expressões ⟩, uma vírgula fará com que o próximo valor seja escrito no início da próxima zona. Um ponto e vírgula fará com que o próximo valor seja escrito imediatamente após o último. A digitação de um ou mais espaços entre as expressões terá o mesmo efeito que o ponto e vírgula.

Caso a ⟨ lista de expressões ⟩ termine com uma vírgula ou um ponto e vírgula, a próxima instrução **PRINT** começará a imprimir na mesma linha, com o espaçamento correspondente. Se a ⟨ lista de expressões ⟩ não tiver uma vírgula ou um ponto e vírgula no fim da mesma, um retorno de carro será impresso no fim da linha. Se a linha impressa for maior do que a largura total da tela, o BASIC segue para a próxima linha física e continua com a impressão.

Os números impressos sempre vem seguidos de um espaço. Os números positivos são precedidos por um expaço. Os números negativos são precedidos por um sinal de menos.

Um ponto de interrogação poderá substituir à palavra **PRINT** dentro de uma instrução PRINT.

PRINT USING

TIPO Instrução.
FORMATO PRINT USING "⟨ expressão string ⟩" ; ⟨ lista de expressões ⟩
EXEMPLO PRINT USING "# # # . # #" ; A, B
USO Para imprimir "strings" ou números, com um formato determinado. A ⟨ lista de expressões ⟩ compreende as expressões numéricas e a alfa-numéricas que devem ser impressas, separadas entre si por vírgulas.
A ⟨ expressão string ⟩ é um literal (ou variável) "string" formado por caracteres especiais de formatação. Esses caracteres de formatação (vide a seguir), determinam o campo e a forma de impressão dos "strings" ou números.
Existem 3 caracteres de formatação que podem ser usados na instrução **PRINT USING**:
1) O caracter "!" indica que somente o primeiro caracter da expressão dado deverá ser impresso.
Exemplo:

```
10 A$="BRASIL"
20 PRINTUSING"!";A$
30 END
```

2) O caracter "\ n espaços \" indica que o número de caracteres impressos será igual a 2 + o número de espaços compreendidos entre os dois \.
Exemplo:

```
10 A$="BRASIL"
20 PRINTUSING"\   \";A$
30 END
```

Se o campo for maior que o "string", ele será escrito do lado esquerdo, e ficarão espaços vazios do lado direito.

3) O caracter "&" indica que deve-se substituir o símbolo "&" com o "string" dado completo.
Exemplo:

```
10 A$="EU":B$="BRASIL"
20 PRINTUSING"& AMO O &";A$,B$
30 END
```

As expressões numéricas podem ser formatadas da seguinte forma:

1) **USING** "#" Indica a quantidade de cifras da expressão que serão impressas. Se a expressão tiver menos dígitos que o número de posições indicadas, o BASIC fará com que a expressão venha precedida de espaços. Se a expressão tiver mais dígitos que o número de posições indicadas, o BASIC escreverá a expressão precedida pelo símbolo de percentual (%). Caso haja necessidade, o BASIC arredondará a cifra.
Exemplo:

```
10 A=109:B=7:C=1198
20 PRINTUSING"###";A,B,C
30 END
```

(continua)

(continuação)

2) **USING** " . " permite introduzir um ponto decimal em qualquer lugar do campo.
Exemplo:

```
10 A=10.21:B=5.5:C=.245:D=3
20 PRINTUSING"##.##";A,B,C,D
30 END
```

3) **USING** " , " A vírgula inserida no lado esquerdo do ponto decimal, fará com que seja impressa uma vírgula antes de cada grupo de três cifras, à esquerda do ponto decimal.
Se a vírgula vier indicada no fim do formato, a vírgula será impressa imediatamente após a expressão.
Exemplo:

```
10 B=1234.5
20 PRINTUSING"####,.##";B
30 PRINTUSING"####.##,";B
40 END
```

4) **USING** "+" indica que será impresso um sinal mais (+) nas expressões positivas e um sinal menos (—) nas negativas.
Se o sinal mais (+) vier indicado como primeiro elemento no formato de impressão, ele será impresso antes da expressão.
Se ele estiver indicado no fim do formato, será impresso imediatamente após a expressão.
Exemplo:

```
10 A=1.25:B=-1.25
20 PRINTUSING"+#.##";A,B
30 PRINTUSING"#.##+";A,B
40 END
```

5) **USING** "—" Esse símbolo só poderá ser utilizado no fim de um formato de impressão. O resultado será a impressão de um espaço após uma expressão positiva e um sinal menos (—) após uma expressão negativa.
Exemplo:

```
10 A=1.25:B=-1.25
20 PRINTUSING"#.##-";A,B
30 END
```

6) **USING** "**" Os dois primeiros asteriscos poderão ser colocados somente no início de um formato de impressão, fazendo com que os espaços da frente venham preenchidos com asteriscos significando zeros.
Exemplo:

```
10 A=10.25:B=1.25:C=-1.25
20 PRINTUSING"**#.##";A,B,C
30 END
```

(continua)

(continuação)

7) **USING** "$$" Os dois símbolos de dólar poderão ser utilizados somente no início de um formato de impressão. Isso se traduzirá numa impressão de um símbolo de dólar antes da expressão.
Exemplo:

```
10 A=12.35:B=-12.35
20 PRINTUSING"$$#####.##";A,B
30 PRINTUSING"$$####.##-";A,B
40 END
```

8) **USING** " **$ " Esses símbolos só poderão ser utilizados no início de um formato de impressão.
Se trata de uma combinação das duas formas anteriormente detalhadas.
Exemplo:

```
10 A=12.35
20 PRINTUSING"**$#.##";A
30 END
```

9) **USING** "^^^^" Esses símbolos devem ser colocados no fim do formato de impressão, e a sua função é a de fazer com que a expressão seja impressa em formato exponencial. O ponto decimal poderá ser colocado em qualquer lugar.
A primeira posição será um espaço, a menos que no formato de impressão seja colocado um sinal de + ou de — no início ou no fim.
Exemplo:

```
10 A=234.56:B=12.34:C=-12.34
20 PRINTUSING"##.##^^^^";A
30 PRINTUSING"#.##^^^^-";C
40 PRINTUSING"+#.##^^^^";B,C
50 END
```

PROGRAMA EXEMPLO

```
10 PRINTUSING"######";123456!
20 PRINTUSING"#.####";12.3456
30 PRINTUSING"####.#";1234.56
40 PRINTUSING"$$####";12345!
50 PRINTUSING"####,#";12345!
60 PRINTUSING"**$###";12!
70 PRINTUSING"#####+";12345
80 PRINTUSING"#####^^^^";1234567890#
```

(continua)

PRINT#

TIPO Instrução.

FORMATO **PRINT** # ⟨número de arquivo⟩, [USING ⟨formato de impressão⟩;] ⟨lista de expressões⟩

EXEMPLO **PRINT** # 2, A, B, C.

USO

Para escrever dados em um arquivo seqüencial.

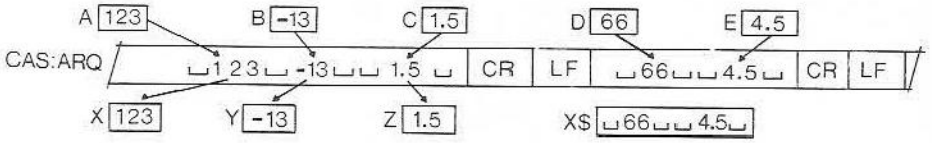
⟨número de arquivo⟩ é o número com o qual o arquivo tem sido aberto através da instrução "OPEN". O arquivo deverá ter sido aberto no modo "output".

Após cada instrução **PRINT**#, no arquivo vem escrito um código "CR" e "LF" (carriage return e linefeed).

Para ler dados de um arquivo seqüencial, usa-se a instrução INPUT#.

Exemplo:

```
OPEN "CAS : ARQ" FOR OUTPUT AS # 1
PRINT # 1, A ; B ; C
PRINT # 1, D ; E
CLOSE # 1
```




```
OPEN "CAS : ARQ" FOR INPUT AS # 1
INPUT # 1, X, Y, Z
LINE INPUT # 1, X$
CLOSE # 1
```

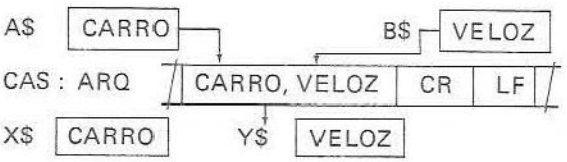
O BASIC coloca automaticamente uma vírgula entre as expressões numéricas indicadas por uma mesma instrução PRINT#, como sinal de separação quando são escritas no arquivo. No caso de expressões alfanuméricas, a vírgula de separação deverá ser colocada explicitamente.

Exemplos:

a)
```
OPEN "CAS : ARQ" FOR OUTPUT AS # 1
PRINT # 1, A$ ; B$
CLOSE # 1
```


```
OPEN "CAS : ARQ" FOR INPUT AS # 1
INPUT # 1, X$
CLOSE # 1
```

b)
```
OPEN "CAS : ARQ" FOR OUTPUT AS # 1
PRINT # 1, A$ ; "," ; B$
CLOSE # 1
```

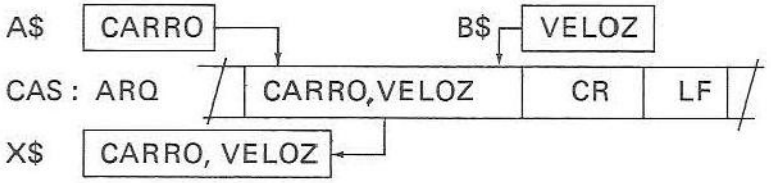



```
OPEN "CAS : ARQ" FOR INPUT AS # 1
INPUT # 1, X$, Y$
CLOSE # 1
```

A vírgula permitiu recuperar as duas expressões por separado

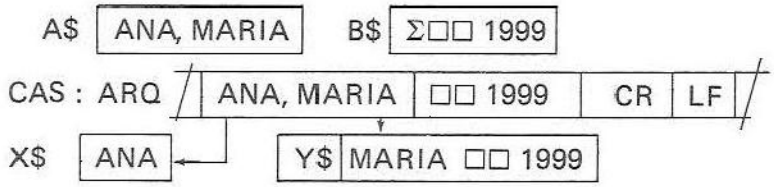
c) No caso de querer recuperar as duas expressões separadas por vírgula como sendo uma expressão só, deverá usar-se a instrução LINE INPUT #.

```
OPEN "CAS : ARQ" FOR OUTPUT AS # 1
PRINT # 1, A$ ; " , " : B$
CLOSE # 1
```



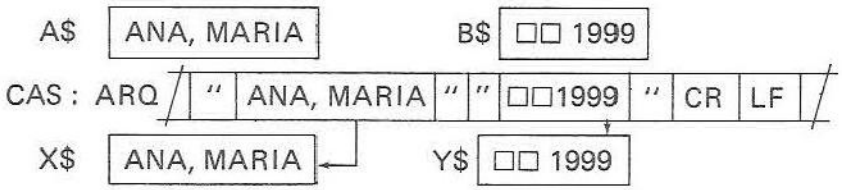

```
OPEN "CAS : ARQ" FOR INPUT AS # 1
LINE INPUT # 1, X$
CLOSE # 1
```

d)
```
OPEN "CAS : ARQ" FOR OUTPUT AS # 1
PRINT # 1, A$ ; B$
CLOSE # 1
```




```
OPEN "CAS : ARQ" FOR INPUT AS # 1
INPUT # 1, X$, Y$
CLOSE # 1
```

e)
```
OPEN "CAS : ARQ" FOR OUTPUT AS # 1
PRINT # 1, CHR$ (34); A$ ; CHR$ (34); CHR$ (34);B$ ; CHR$ (34)
CLOSE # 1
```




```
OPEN "CAS : ARQ" FOR INPUT AS # 1
INPUT # 1, X$, Y$
CLOSE # 1
```

PSET

TIPO Instrução.

FORMATO PSET [(X, Y) | STEP (X, Y)] [, código de cor]

EXEMPLO PSET (100, 100), 8

USO Atribuir a um ponto de imagem especificado, uma cor determinada.
⟨ X ⟩ é a coordenada X do ponto de imagem, e deverá ser um número inteiro entre 0 e 255.
⟨ Y ⟩ é a coordenada Y do ponto de imagem, e deverá ser um número inteiro entre 0 e 191.
Quando vier indicada a palavra "STEP", os valores X e Y serão interpretados relativamente à posição do cursor. Nesse caso, poderão até assumir valores negativos.
⟨ código de cor ⟩ é um número inteiro de 0 a 15 que indica o número da cor. Se não for indicado o número da cor, será utilizada a cor do primeiro plano.
O valor "default" é 15.
(Vide "PRESET").

PROGRAMA EXEMPLO

```
10 SCREEN3
20 FORI=1TO200
30 X1=INT(RND(1)*256)
40 Y1=INT(RND(1)*192)
50 CL=INT(RND(1)*14+2)
60 PSET(X1,Y1),CL
70 NEXTI
80 END
```

PUT

TIPO Instrução.

FORMATO PUT [#] ⟨ X ⟩ [, ⟨ Y ⟩]

EXEMPLO PUT # 1, 1

USO Escrever um dado do buffer em um arquivo aleatório em disco flexível.
⟨ X ⟩ é o número sob o qual o arquivo tem sido aberto através da *instrução "OPEN"*.
⟨ Y ⟩ é o número do dado que deverá ser escrito. Deverá ser um número inteiro compreendido entre 0 e 32767.
Quando não vier indicado ⟨ Y ⟩, será escrito o dado sucessivo, ou seja o dado imediato ao último dado lido ou escrito.
O buffer na memória deverá ter sido reservado com a instrução "FIELD".
Usa-se a instrução "GET" para ler no buffer um dado de um arquivo aleatório de um disco.

NOTA: "PUT" só poderá ser utilizado quando o computador estiver carregado com o HB-DOS e os acionadores de disco 1 ou 2.

PUT SPRITE

TIPO Instrução.

FORMATO **PUT SPRITE** ⟨Z⟩[,[STEP](⟨X⟩,⟨Y⟩)][,⟨xx⟩][,⟨yy⟩]

EXEMPLO **PUT SPRITE** 1,(50,50),8,2.

USO Inserir um determinado sprite na tela do TV, (SCREEN 1, 2 ou 3).

⟨Z⟩ indica a prioridade (ou número do plano) do sprite, e deverá ser um número inteiro entre Ø e 31.

⟨X⟩ é a coordenada X da posição do sprite na tela, e deverá estar compreendida entre −32 e 255.

⟨Y⟩ é a coordenada Y da posição do sprite na tela, e deverá estar compreendida entre −32 e 191, aceitando também os valores 2Ø8 e 2Ø9. Se Y assume o valor 2Ø8 todos os sprites com prioridade menor (planos de trás) desaparecem da tela.

Se Y assume o valor 2Ø9, então esse sprite desaparece da tela.

Ao ser utilizada a palavra "STEP", os valores ⟨X⟩ e ⟨Y⟩ são interpretados relativamente à posição do cursor. Nesse caso, X e Y poderão até assumir valores negativos.

Se ⟨X⟩ ou ⟨Y⟩, ou ambos forem omitidos, serão utilizadas as coordenadas ativas de X ou Y.

⟨XX⟩ é o número da cor do sprite, e deverá ser um inteiro compreendido entre Ø e 15 (vide tabela de cores).

Quando XX não vier indicado, será assumida a última cor utilizada no primeiro plano. O valor "default" é 15.

⟨YY⟩ é o número do sprite, tal como foi registrado na variável "SPRITE$(X)".

Se ⟨YY⟩ não for indicado, o número do sprite corresponderá ao número que indica a prioridade. O número do sprite deverá ser menor que 256 quando o tamanho do sprite é Ø ou 1, e menor que 64 quando o tamanho do sprite é 2 ou 3.

A dimensão do sprite é determinada pela instrução "SCREEN".

PROGRAMA EXEMPLO

```
10 '***BORBOLETAS***
20 DEFINTA-Z
30 DIMX(10),Y(10)
40 SCREEN2,3:COLOR,1,1:CLS:I=RND(-TIME)
50 FORI=1TO10:X(I)=96:Y(I)=I*15:NEXT
60 FORI=0TO31:READA$:B$=B$+CHR$(VAL("&H"+A$)):NEXT
70 SPRITE$(0)=B$
80 FORI=1TO10
90 PUTSPRITEI,(X(I),Y(I)),I+4,0
100 NEXT
110 FORI=1TO10
120 X(I)=(X(I)+(RND(1)*21-10))MOD256
130 Y(I)=(Y(I)+(RND(1)*21-10))MOD192
140 NEXT
150 GOTO80
160 DATA0C,0C,62,F2,FA,DD,CF,C7
170 DATAFF,7F,3F,1B,37,3E,1C,00
180 DATA30,30,46,4F,5F,BB,F3,E3
190 DATAFF,FE,FC,D8,EC,7C,38,00
```

READ

TIPO Instrução.

FORMATO **READ** ⟨ lista de variáveis ⟩

EXEMPLO **READ** K, M, J$

USO Ler uma constante de uma instrução "DATA" e atribui-la a uma variável. Uma instrução "READ" somente poderá ser usada em combinação com uma instrução "DATA".

A instrução **READ** atribui variáveis aos valores indicados na instrução DATA numa relação de um-a-um. As variáveis da instrução READ poderão ser numéricas ou alfanuméricas, e os valores lidos deverão concordar com o tipo de variável especificada. No caso de não concordar aparecerá a mensagem de erro "ERRO SINTAXE".

Uma única instrução "**READ**" poderá acessar a uma ou mais instruções DATA. Também é possível acessar a uma única instrução "DATA" através de diversas instruções "**READ**".

Se o número de variáveis da instrução "**READ**" for maior que o número de constantes na instrução "DATA", o BASIC emitirá uma mensagem de erro "SEM DATA".

Se o número de variáveis da instrução "**READ**" for menor que o número de constantes na instrução "DATA", a próxima instrução "**READ**" continuará com a leitura das constantes, na mesma instrução DATA. Se não houver uma outra instrução "**READ**", as constantes que não tem sido lidas serão ignoradas.

Para reler instruções DATA a partir do início, usa-se a instrução RESTORE.

PROGRAMA EXEMPLO

```
10 PRINT"***NOTAS***":PRINT
20 PRINT"nome      matem. geogr. total":PRINT
30 FORI=1TO4
40 READA$,B,C
50 PRINTA$;
60 D=B+C
70 PRINTTAB(6)USING"######";B;C;D
80 NEXTI
90 DATA JOÃO,90,73
100 DATA MARIA,85,80
110 DATA ANA,82,65
120 DATA PEDRO,75,92
```

REM

TIPO Instrução.
FORMATO **REM** [⟨ observação ⟩]
EXEMPLO **REM** PROGRAMA TESTE
'PROGRAMA TESTE
USO Permite inserir comentários dentro do programa.
A instrução **REM** não será executada durante o desenvolvimento do programa, mas será impressa quando o programa for listado.
O programa poderá ser mandado a uma instrução **REM** através de uma instrução "GOTO" ou uma "GOSUB". Nesse caso a execução do mesmo continuará a partir da instrução imediatamente após à **REM**.
No lugar da palavra **REM** poderá ser utilizada uma apóstrofo.
A instrução **REM** não deve ser utilizada em uma instrução DATA.

PROGRAMA EXEMPLO (1)

```
10 REM CALCULOS
20 FORI=1TO10
30 SOM=SOM+V(I)
40 NEXTI
50 SOM=SOM/10
```

PROGRAMA EXEMPLO (2)

```
10 FORI=1TO10'CALCULOS
20 SOM=SOM+V(I)
30 NEXTI
40 SOM=SOM/10
```

RENUM

TIPO Comando.
FORMATO **RENUM** [[⟨ novo número ⟩] [, [⟨ velho número ⟩] [, ⟨ incremento ⟩]]]
EXEMPLO **RENUM**
USO Permite a renumeração das linhas do programa.
⟨ novo número ⟩ é a primeira linha do programa que será utilizada na nova seqüência. Se esse número não vier indicado, o programa começará com a linha 10.
⟨ velho número ⟩ é a linha do programa existente a partir da qual deverá começar a renumeração. Se esse número não vier indicado a renumeração começará a partir da primeira linha do programa.
⟨ incremento ⟩ é o valor do incremento entre uma linha do programa e a seguinte, na nova seqüência. Se não vier indicado, o BASIC suporá automaticamente que o valor do incremento é 10.
O comando **RENUM** renumerará automaticamente todas as referências aos números de linha dentro do programa, nas instruções "GOTO", "GOSUB", "IF THEN ELSE", "ON GOTO" e "ON GOSUB".
Se o BASIC achar uma referência a um número de linha inexistente, aparecerá a mensagem de erro "NUM. LINHA INDEFINIDO". O número de linha inexistente não será modificado pelo comando RENUM.
Exemplo:
RENUM (Dessa forma será renumerado o programa inteiro, a partir da primeira linha. A primeira linha terá o número 10. As linhas seguintes serão numeradas com um incremento de 10).
RENUM 1000, 900, 50 (Nesse caso será renumerado o programa começando pela linha 900. A linha 900 será mudada para 1000 e as linhas seguintes serão numeradas com um incremento de 50).

RESTORE

TIPO Instrução.
FORMATO **RESTORE** [⟨ número de linha ⟩]
EXEMPLO **RESTORE 800**
USO Permite que uma instrução DATA seja relida a partir de uma linha específica, através de uma inscrição READ.
⟨ número de linha ⟩ é um número de linha de programa de uma instrução DATA.
Após a execução da instrução **RESTORE**, a primeira constante na instrução DATA da linha dada será lida com a próxima instrução READ. Se ⟨ número de linha ⟩ não vier indicado, a próxima instrução READ provocará a leitura da primeira constante da primeira instrução DATA do programa.

RIGHT$

TIPO Função.
FORMATO **RIGHT$** (⟨ X$ ⟩, ⟨ M ⟩)
EXEMPLO **PRINT RIGHT$**
USO Fornece uma expressão alfanumérica composta pelos ⟨ n ⟩ caracteres do extremo direito de X$.
⟨ n ⟩ é um número inteiro entre 0 e 255.
Quando ⟨ n ⟩ for maior que o comprimento total de ⟨ X$ ⟩ será indicado o conteúdo completo de ⟨ X$ ⟩.
Quando ⟨ n ⟩ = 0, obtém-se uma expressão alfanumérica vazia.

PROGRAMA COMPLETO

```
10 CLS
20 INPUT"Palavra";A$
30 L=LEN(A$)
40 FORI=1TOL
50 B$=RIGHT$(A$,I)
60 B$=RIGHT$(SPACE$(50)+B$,L)
70 PRINTB$
80 NEXT
90 END
```

RND

TIPO Função.
FORMATO RND (⟨ X ⟩)
EXEMPLO R = RND (1)
USO Fornece um número aleatório compreendido entre Ø e 1.
Cada vez que o programa é executado através do comando RUN, a mesma seqüência de cifras aleatórias é gerada.
Quando ⟨ X ⟩ é maior que zero, será gerado o próximo número aleatório sempre dentro da mesma seqüência independentemente do valor de X.
Quando ⟨ X ⟩ é igual a zero, repete o último número gerado.
Quando ⟨ X ⟩ é menor que zero, gera um número aleatório que corresponde ao valor de X após o qual segue uma seqüência de números aleatórios.
Para obter um número totalmente aleatório, o gerador deverá iniciar uma nova seqüência cada vez que o programa começa a ser executado. Isso pode ser conseguido usando a variável "TIME", com valor negativo.

EXEMPLO DE X ⟩ Ø

```
10 FORN=1TO10
20 PRINTRND(1)
30 NEXTN
```

```
.59521943994623
.10658628050158
.76597651772823
.57756392935958
.73474759503023
.18426812909758
.37075377905223
.94954151651558
.63799556899423
.47041117641358
```

EXEMPLO de X = Ø

```
10 PRINTRND(1)
20 PRINTRND(Ø)
30 PRINTRND(-1)
40 PRINTRND(Ø)
```

```
.59521943994623
.59521943994623
.04389820420821
.04389820420821
```

EXEMPLO DE X ⟨ Ø

```
10 PRINTRND(-1)
20 FORN=1TO10
30 PRINTRND(N):NEXTN
```

```
.04389820420821
.0962486816692
.21069655852301
.3265173630504
.47775124336581
.3409147084636
.12971184081661
.0977770174288
.35157860175541
.835389696666
.63902641386221
```

PROGRAMA EXEMPLO

```
10 'RABISCOS
20 SCREEN2
30 R=RND(-TIME)
40 X=RND(1)*256:Y=RND(1)*192
50 L=RND(1)*16+16
60 XS=X:YS=Y
70 X=X+RND(1)*L-L/2
80 Y=Y+RND(1)*L-L/2
90 C=INT(RND(1)*16)
100 LINE(XS,YS)-(X,Y),C
110 IFX<ØORX>255THEN40
120 IFY<ØORY>191THEN40
130 GOTO60
```

RSET

TIPO Instrução.
FORMATO **RSET** ⟨ X$ ⟩ = ⟨ Y$ ⟩
EXEMPLO **RSET** A$ = B$
USO Preencher, começando pela direita, a variável ⟨ X$ ⟩, com o conteúdo da variável ⟨ Y$ ⟩ ou da expressão alfanumérica ⟨ Y$ ⟩.
As instruções "LSET' e **"RSET"** devem ser usadas para inserir os dados no buffer para um arquivo aleatório em disco.
Os dados numéricos devem ser convertidos em dados alfanuméricos com as funções "MKI$", "MKS$" e "MKD$" quando são inseridos no buffer para um arquivo aleatório.
Os dados numéricos deverão ser convertidos de alfanuméricos em numéricos com as funções "CVI", "CVS" e "CVD" quando são tirados do buffer par um arquivo aleatório.

NOTA: "RSET" só poderá ser utilizado quando o computador estiver carregado com o HB-DOS e os acionadores de disco 1 ou 2.

RUN

TIPO Comando.
FORMATO **RUN** [⟨ dispositivo ⟩ : ⟨ nome do programa ⟩] [⟨ X ⟩]
EXEMPLO **RUN** 500
USO Fazer rodar um programa carregado na memória do computador ou em um ⟨ dispositivo ⟩.
O ⟨ dispositivo ⟩ pode ser:
CAS : gravador
A : acionador de disco 1
B : acionador de disco 2
O programa deve ter sido escrito na fita cassete ou no disco sob o formato ASCII, através do comando "SAVE".
O nome do programa é uma constante alfanumérica que foi especificada no momento de escrever o programa na fita ou no disco.
Se o ⟨ dispositivo ⟩ : ⟨ nome do programa ⟩ não vier indicado, será rodado o programa carregado na memória do computador.
⟨ X ⟩ é o número de linha do programa a partir da qual começará a execução do mesmo.
Se ⟨ X ⟩ não vier indicado, a execução do programa começará pela primeira linha.

NOTA: **"RUN CAS"** só poderá ser utilizado quando o computador estiver conectado a um gravador.

"RUN A" ou **"RUN B"** só poderão ser utilizados quando o computador estiver carregado com o HB-DOS e os acionadores de disco.

SAVE

TIPO Comando.

FORMATO SAVE "⟨ dispositivo ⟩ : [⟨ nome do arquivo ⟩]"

EXEMPLO SAVE "CAS : PROGR"

USO Transferir um programa BASIC carregado na memória do computador para um ⟨ dispositivo ⟩.

Os ⟨ dispositivo ⟩ podem ser:

CAS : gravador
A : acionador de disco 1
B : acionador de disco 2

CTRL–Z é tratado como fim de arquivo.

O programa será escrito no cassete ou no disco em formato ASCII e poderá ser lido pelo computador através dos comandos "LOAD" ou "MERGE".

O nome do programa é uma constante alfanumérica.

NOTA: "SAVE CAS" só poderá ser utilizado quando o computador estiver conectado a um gravador.
"SAVE A" e "SAVE B" só poderão ser utilizados quando o computador estiver carregado com o HB-DOS e os acionadores de disco 1 ou 2.

SCREEN

TIPO Instrução.

FORMATO **SCREEN** [⟨X⟩] [,⟨Y⟩] [,⟨Z⟩] [,⟨XX⟩] [,⟨YY⟩]

EXEMPLO **SCREEN** 1, 2.

USO Definir o modo de trabalho na tela, a dimensão do sprite, o estalido das teclas, a velocidade de transmissão do gravador e o tipo de impressora usada.

⟨X⟩ identifica o modo da tela. Deverá ser um inteiro entre Ø e 3.

- Ø : Modo texto 1 (40 caracteres x 24 linhas)
- 1 : Modo texto 2 (32 caracteres x 24 linhas)
- 2 : Modo gráfico 1 (Alta resolução)
- 3 : Modo gráfico 2 (Multi-color)

Quando X não vier indicado, será reutilizado o último modo usado. O valor "default" de X é zero.

As instruções gráficas "PUT SPRITE", "CIRCLE", "DRAW", "LINE", "PAINT", "PSET", "PRESET", "ON SPRITE GOSUB, "SPRITE ON/OFF/STOP" e "POINT" poderão ser utilizadas somente nos modos gráficos.

⟨Y⟩ determina a dimensão dos sprites. Deverá ser um inteiro entre Ø e 3.

- Ø : sprites pequenos (8 x 8 pixels)
- 1 : sprites pequenos aumentados (16 x 16 pixels)
- 2 : sprites grandes (16 x 16 pixels)
- 3 : sprites grandes aumentados (32 x 32 pixels)

Quando Y não vier indicado, será utilizada a dimensão do último sprite visualizado. O valor "default" de Y é zero.

⟨Z⟩ determina se deverá ser produzido o som de estalo quando for pressionada uma tecla. Poderá valer Ø ou 1.

- Ø : sem estalido
- 1 : com estalido

Quando ⟨Z⟩ não vier indicado, será adotado o último valor utilizado. O valor "default" de Z é 1.

⟨XX⟩ indica a velocidade de transmissão para o gravador. Deverá ser 1 ou 2.

- 1 : velocidade de transmissão de 12ØØ bauds.
- 2 : velocidade de transmissão de 24ØØ bauds.

Quando XX não vier indicado, será utilizada a última velocidade em uso. O valor "default" de XX é 1.

⟨YY⟩ determina o tipo de impressora utilizada. Poderá valer Ø ou 1.

- Ø : impressora padrão HB-8000
- 1 : impressora padrão ABICOMP
  (Os símbolos gráficos são convertidos em espaços.)

Quando YY não vier indicado, será reutilizada a indicação da última impressora usada. O valor "default" de YY é 1.

PROGRAMA EXEMPLO

```
10 FORI=2TO3
20 SCREENI:COLOR15,4,7
30 LINE(32,32)-(200,132),6
40 FORK=0TO1500:NEXT
50 NEXTI
60 END
```

SGN

TIPO Função.
FORMATO **SGN** (X)
EXEMPLO PRINT **SGN** (A)
USO Assume o valor 1 quando X ⟩ Ø, Ø quando X = 0 e –1 quando X ⟨ Ø.
PROGRAMA EXEMPLO

```
10 INPUT"Número";I
20 J=SGN(I)
30 J=J+2
40 ONJGOSUB80,90,100
50 PRINTA$
60 PRINT
70 GOTO10
80 A$="Seu número é negativo":RETURN
90 A$="Seu número é zero":RETURN
100 A$="Seu número é positivo":RETURN
```

SIN

TIPO Função.
FORMATO **SIN** (X)
EXEMPLO PRINT **SIN** (1)
USO Fornece o seno de X em radianos.
SIN (X) calcula o seno de X com dupla precisão.
PROGRAMA EXEMPLO

```
10 '***ESPIRAL***
20 K=75
30 PI=3.141592653#
40 SCREEN2
50 PSET(128,177)
60 FORI=0TOPI*2STEPPI/30
70 K=K-.25
80 IFK<0THEN140
90 X=128+K*SIN(I)
100 Y=88+K*COS(I)*1.2
110 LINE-(X,Y)
120 NEXT
130 GOTO60
140 GOTO140
```

SOUND

TIPO Instrução.
FORMATO **SOUND** ⟨ registro do GSP ⟩, ⟨ dado ⟩
EXEMPLO **SOUND** 7, 254.
USO Inserir um valor em um dos registros do gerador de sons programável (GSP).
O ⟨ registro do GSP ⟩ é o número do registro e deverá estar compreendido entre Ø e 15.
O ⟨ dado ⟩ é o valor a ser inserido no registro e deverá ser um número inteiro compreendido entre Ø e 255.

PROGRAMA EXEMPLO

```
10 PRINT"Isto é barulho aleatório"
20 FORI=ØTO13
30 SOUNDI,Ø:NEXT
40 SOUND7,254:SOUND8,15
50 FORI=1TO255STEP.1
60 GOSUB100
70 NEXT
80 GOSUB100
90 GOTO80
100 RØ=INT(RND(1)*I)
110 SOUNDØ,RØ
120 RETURN
```

SPACE$

TIPO Função.
FORMATO SPACE$ (⟨ X ⟩)
EXEMPLO PRINT SPACE$ (10); "ABC"
USO Fornece um "string" de ⟨ X ⟩ espaços.
⟨ X ⟩ deverá fornecer um número inteiro compreendido entre Ø e 255.

PROGRAMA EXEMPLO

```
10 OPEN"CRT:"FOROUTPUTAS#1
20 CLS:I=RND(-TIME):FORI=ØTO10
30 S=INT(RND(1)*16)
40 PRINT#1,"*";
50 PRINT#1,SPACE$(S);"*";
60 PRINT#1,SPACE$(16-S);S;"espaços"
70 NEXT
80 CLOSE:END
```

SPC

TIPO Função.
FORMATO SPC (⟨ I ⟩)
EXEMPLO PRINT SPC (10); A$
USO Imprimir I espaços na tela.
SPC somente pode ser utilizado em combinação com as instruções "PRINT" ou "LPRINT".
⟨ I ⟩ deve ser um número inteiro compreendido entre Ø e 255.

PROGRAMA EXEMPLO

```
10 CLS:I=RND(-TIME):FORI=ØTO10
20 S=INT(RND(1)*16)
30 PRINT"*";
40 PRINTSPC(S);"*";
50 PRINTSPC(16-S);S;"espaços"
60 NEXT:END
```

SPRITE ON/ OFF/STOP

TIPO	Instrução.
FORMATO	**SPRITE ON** **SPRITE OFF** **SPRITE STOP**
EXEMPLO	**SPRITE ON**
USO	Ativar o controle do BASIC sobre uma colisão entre sprites. Após a instrução **"SPRITE ON"**, o BASIC controlará em cada instrução, a ocorrência de uma colisão entre sprites. Caso a colisão aconteça, o BASIC se desviará para a subrotina indicada na instrução "ON SPRITE GOSUB". Após a instrução **"SPRITE OFF"**, o BASIC deixará de controlar a ocorrência de colisões entre sprites. Após a instrução **"SPRITE STOP"**, o BASIC continua controlando a ocorrência de colisões, mas não se desvia para a subrotina indicada em "ON SPRITE GOSUB". Caso a colisão aconteça, o evento fica na memória e a subrotina é seguida logo após um **"SPRITE ON".**

SPRITE$

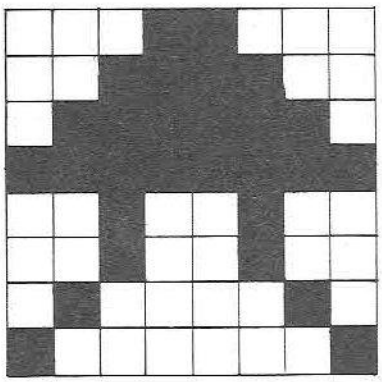
TIPO Variável do sistema
FORMATO **SPRITE$** (⟨ número do sprite ⟩)
EXEMPLO **SPRITE$** (0) = **STRING$** (32, CHR$ (&HFF))
USO Definir o desenho do sprite.

O ⟨ número do sprite ⟩ deve ser um inteiro compreendido entre Ø e 255 quando o tamanho do sprite é Ø ou 1.

Quando o tamanho é 3 ou 4 (definido através do "SCREEN"), o ⟨ número ⟩ deve estar compreendido entre Ø e 63.

O comprimento da variável está fixado em 32 bytes. Se o string for definido por menos do que 32 caracteres, será preenchido com CHR$ (Ø).

O conteúdo da variável **SPRITE$** pode ser um valor binário, hexadecimal ou decimal, como ilustra o exemplo a seguir.



Binário		Hexadecimal		Decimal
&B00011000	=	&H18	=	24
&B00111100	=	&H3C	=	60
&B01111110	=	&H7E	=	126
&B11111111	=	&HFF	=	255
&B00100100	=	&H24	=	36
&B00100100	=	&H24	=	36
&B01000010	=	&H42	=	66
&B10000001	=	&H81	=	129

DEFINIÇÃO BINÁRIA

```
10 SCREEN2,1
20 B$=""
30 FORI=1TO8
40 READA$:B$=B$+CHR$(VAL("&B"+A$))
50 NEXTI
60 SPRITE$(0)=B$
70 PUTSPRITE0,(100,100),10,0
80 GOTO70
90 DATA00011000,00111100
100 DATA01111110,11111111
110 DATA00100100,00100100
120 DATA01000010,10000001
```

(continua)

(continuação)

DEFINIÇÃO DECIMAL

```
10 SCREEN2,1
20 B$=""
30 FORI=1TO8
40 READA:B$=B$+CHR$(A)
50 NEXTI
60 SPRITE$(0)=B$
70 PUTSPRITE0,(100,100),10,0
80 GOTO70
90 DATA24,60,126,255,36,36,66,129
```

DEFINIÇÃO HEXADECIMAL

```
10 SCREEN2,1
20 B$=""
30 FORI=1TO8
40 READA$:B$=B$+CHR$(VAL("&H"+A$))
50 NEXTI
60 SPRITE$(0)=B$
70 PUTSPRITE0,(100,100),10,0
80 GOTO70
90 DATA18,3C,7E,FF,24,24,42,81
```

PROGRAMA EXEMPLO

```
10 SCREEN1,3
20 PRINT"***RATINHO***
30 FORI=1TO16
40 READD$
50 A$=A$+CHR$(VAL("&B"+LEFT$(D$,8)))
60 B$=B$+CHR$(VAL("&B"+RIGHT$(D$,8)))
70 NEXTI
80 SPRITE$(0)=A$+B$
90 PUTSPRITE0,(50,70),15,0
100 PUTSPRITE1,(90,70),14,0
110 PUTSPRITE2,(130,70),1,0
120 PUTSPRITE3,(170,70),13,0
130 PRINT"Para apagar esses ratos"
140 PRINT"digitar PUT SPRITE0,(0,208)"
150 DATA0000000000011110
160 DATA0000100000101001
170 DATA0001011111101101
180 DATA0001111111111001
190 DATA0011111011111111
200 DATA0001111111111000
210 DATA0000001111111000
220 DATA0000011111110000
230 DATA0000001111100010
240 DATA0000000111100100
250 DATA1100001111100100
260 DATA0011111111110010
270 DATA0000001111110010
280 DATA0000001111110010
290 DATA0000000111111100
300 DATA0000000111111100
310 DATA0000011111110000
```

SQR

TIPO Função.
FORMATO **SQR** (⟨ X ⟩)
EXEMPLO A = **SQR** (2)
USO Fornece o valor da raiz quadrada de X.
X deverá ser ⟩ = Ø.

PROGRAMA EXEMPLO

```
10 INPUT"NUMERO";I
20 PRINT"RAIZ QUADRADA DE I="
30 PRINTSQR(I)
40 PRINT"I^.5="
50 PRINTI^.5
60 PRINT
70 GOTO10
```

STICK

TIPO Função.

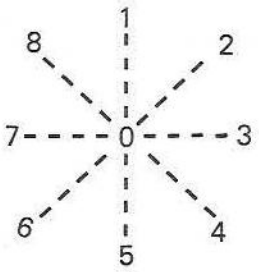
FORMATO **STICK (⟨X⟩)**

USO Fornece a posição de um joystick ou estado das teclas de controle do cursor.

⟨X⟩ poderá representar uma das seguintes indicações:

Ø : as teclas de controle do cursor servem como joystick.
1 : responde ao joystick ligado ao conector nº1.
2 : responde ao joystick ligado ao conector nº 2.

A função fornece os seguintes valores.



0 = neutro
1 = norte
2 = nordeste
3 = leste
4 = sudeste
5 = sul
6 = sudoeste
7 = oeste
8 = noroeste

PROGRAMA EXEMPLO

```
10 SCREEN1,1
20 SPRITE$(0)=STRING$(8,CHR$(255))
30 X=120:Y=90
40 PUTSPRITE0,(X,Y),8,0
50 ONKEYGOSUB240
60 KEY(1)ON
70 PRINT"Movimente as teclas do cursor"
80 PRINT
90 PRINT"Para parar o programa,pressione";
100 PRINT" a tecla F1"
110 A=STICK(0)
120 ONAGOTO140,150,160,170,180,190,200,210
130 GOTO110
140 Y=Y-1:GOTO220
150 X=X+1:Y=Y-1:GOTO220
160 X=X+1:GOTO220
170 X=X+1:Y=Y+1:GOTO220
180 Y=Y+1:GOTO220
190 X=X-1:Y=Y+1:GOTO220
200 X=X-1:GOTO220
210 X=X-1:Y=Y-1:GOTO220
220 PUTSPRITE0,(X,Y),8,0
230 GOTO110
240 PUTSPRITE0,(0,209)
250 END
```

STOP

TIPO	Instrução.
FORMATO	**STOP**
EXEMPLO	**STOP**
USO	Para interromper a execução de um programa e voltar ao nível de comando. A instrução **STOP** poderá ser utilizada em qualquer lugar de um programa. Ao ser achada a instrução **STOP**, será impressa a mensagem "Parei em nnn" (nnn é o número de linha do programa). A diferença com a instrução END é que **STOP** não fecha os arquivos. A execução do programa poderá ser retomada usando o comando CONT.

STOP ON/ OFF/STOP

TIPO	Instrução.
FORMATO	**STOP ON** **STOP OFF** **STOP STOP**
EXEMPLO	STOP ON
USO	Controlar a situação em que as teclas CTRL e STOP são pressionadas simultaneamente. Após a instrução **"STOP ON"** o BASIC verificará antes de cada instrução se as teclas CTRL e STOP tem sido pressionadas simultaneamente. Em caso afirmativo, o BASIC seguirá para a subrotina indicada pela instrução **"ON STOP GOSUB".** Após a instrução **"STOP OFF"**, o BASIC deixará de verificar se as teclas CTRL e STOP foram pressionadas simultaneamente. Após a instrução **"STOP STOP"**, o BASIC continua controlando a situação de serem pressionadas ambas teclas simultaneamente, mas não segue para a subrotina indicada por "ON STOP GOSUB". Caso aconteça, o evento fica na memória e o BASIC, seguirá para a subrotina assim que aparecer uma instrução **"STOP ON".**

STRIG

TIPO Função.
FORMATO **STRIG** (⟨n⟩)
EXEMPLO **STRIG** (2)
USO Fornece o estado do botão disparador do joystick ou da barra de espaçamento.
⟨n⟩ é um inteiro que vai de Ø a 4, e tem os seguintes significados:

Ø : barra de espaçamento
1 ou 3 : botão disparador do joystick 1
2 ou 4 : botão disparador do joystick 2.

Ao ser pressionada a barra de espaçamento ou um dos disparadores, a função assume o valor −1. Caso contrário, vale zero.

PROGRAMA EXEMPLO

```
10 PRINT"Pressione a barra de espaços"
20 I=STRIG(Ø)
30 IFITHENBEEP
40 GOTO20
```

STRIG ON/ OFF/STOP

TIPO Instrução.
FORMATO **STRIG** (⟨n⟩) **ON**
STRIG (⟨n⟩) **OFF**
STRIG (⟨n⟩) **STOP**
EXEMPLO **STRIG** (Ø) **ON**
USO Ativar o controle do BASIC sobre o estado dos botões disparadores do joystick ou da barra de espaçamento.
⟨n⟩ é um número inteiro que vai de zero a 4 e tem o seguinte significado:

0 : barra de espaçamento
1 ou 3 : botão disparador do joystick nº 1
2 ou 4 : botão disparador do joystick nº 2.

Após a instrução **"STRIG (n) ON"**, o BASIC verificará em cada instrução, se tem sido pressionado o botão disparador do joystick indicado.
Em caso afirmativo, o BASIC seguirá para a sub-rotina indicada em "ON STRIG GOSUB".
Após a instrução **"STRIG (n) OFF"**, o BASIC deixará de verificar se os botões disparadores tem sido pressionados.
Após a instrução **"STRIG (n) STOP"**, o BASIC continua controlando o estado dos disparadores do joystick, mas não segue para a sub-rotina indicada em "ON STRIG GOSUB". Caso aconteça, o evento fica na memória e o BASIC segue para a sub-rotina assim que aparecer uma instrução **"STRIG ON"**.

STR$

TIPO Função.
FORMATO STR$ (⟨ X ⟩)
EXEMPLO A$ = STR$ (123)
USO Fornece uma representação alfanumérica da expressão numérica ⟨ X ⟩. Para achar a representação numérica de uma expressão alfanumérica usa-se a função "VAL".

PROGRAMA EXEMPLO

```
10 INPUT"Número decimal";N:PRINT
20 A$=STR$(N)
30 H%=INSTR(A$,".")
40 IFH%<1GOTO80
50 MID$(A$,H%,1)=","
60 H$=LEFT$(A$,H%+2)
70 PRINTH$
80 END
```

STRING$

TIPO Função.
FORMATO STRING$ (⟨ I ⟩, |⟨ J ⟩| |)
|⟨ X$ ⟩|
EXEMPLO STRING$ (10, "+")
USO Fornece um valor alfanumérico de comprimento I, contendo somente caracteres cujo código ASCII é J ou o primeiro caracter do string X$. ⟨ I ⟩ é o comprimento do "string" e deverá estar compreendido entre zero e 255.
⟨ J ⟩ é um código de caracter, e deverá estar compreendido entre 32 e 255.

PROGRAMA EXEMPLO

```
10 SCREEN1:COLOR15,4,7
20 A$=CHR$(42)
30 FORK=1TO20
40 R=RND(-TIME):C=INT(RND(1)*20)
50 LOCATE3,K
60 PRINTUSING"##";C
70 LOCATE6,K
80 PRINTSTRING$(C,A$)
90 NEXTK
100 END
```

SWAP

TIPO Instrução.
FORMATO **SWAP** ⟨ variável ⟩, ⟨ variável ⟩
EXEMPLO **SWAP** A$, B$
USO Intercambiar os conteúdos de duas variáveis.
Pode ser utilizado qualquer tipo de variável (inteira, de simples precisão, de precisão dupla, alfanumérica).
A única condição é que ambas as variáveis sejam do mesmo tipo.

PROGRAMA EXEMPLO

```
10 FORI=0TO5
20 X(I)=INT(RND(1)*99)
30 Y(I)=INT(RND(1)*99)
40 NEXT:GOSUB110
50 PRINT
60 FORI=0TO5
70 SWAPX(I),Y(I)
80 NEXT
90 GOSUB110
100 END
110 PRINT"I  X(I)  Y(I)"
120 FORI=0TO5
130 PRINTUSING"#   ##   ##";I;X(I);Y(I)
140 NEXT:RETURN
```

TAB

TIPO Função.
FORMATO **TAB** (⟨ I ⟩)
EXEMPLO PRINT **TAB** (Ø) ; "BASIC"
USO Deslocar o cursor sobre a mesma linha até a posição I.
Se a posição atual do cursor já estiver à direita da posição I, TAB não terá efeito.
A função **TAB** só pode ser usado em combinação com "PRINT" ou "LPRINT".
⟨ I ⟩ deverá ser um inteiro compreendido entre Ø e 255.
A posição extrema esquerda é zero, e a extrema direita é igual à largura menos 1.

PROGRAMA EXEMPLO

```
10 PRINT
20 PRINT"*",TAB(9)"*"
30 PRINT"*","*"
40 PRINT"*"TAB(9)"*"
50 PRINT"*";TAB(9);"*"
60 END
```

TAN

TIPO Função.
FORMATO **TAN** (⟨ X ⟩)
EXEMPLO A = **TAN** (123)
USO Fornece o valor da tangente de X em radianos.
TAN (X) é calculada com dupla precisão.

PROGRAMA EXEMPLO

```
10 SCREEN2
20 PI=3.141592653#
30 X=5:LINE(5,100)-(240,100),7
40 FORI=-PITOPISTEPPI/59
50 Y=100-TAN(I)*10
60 PRINTX
70 PSET(X,Y)
80 X=X+2
90 NEXT
100 GOTO100
```

TIME

TIPO Variável do sistema.
FORMATO **TIME**
TIME = 0
EXEMPLO PRINT **TIME**
USO Inteiro gerado pelo timer interno do sistema.
Cada vez que o processador de vídeo (VDP) gera uma interrupção (60 vezes/segundo), o valor da variável é incrementado em 1. Assim sendo, quando uma interrupção é desabilitada (como por exemplo quando um programa é escrito em cassete ou é lido do cassete), TIME permanece no mesmo valor.

PROGRAMA EXEMPLO

```
10 '***TIME***
20 CLS
30 LOCATE10,8
40 PRINT"COMEÇOU!"
50 TIME=0
60 LOCATE10,10
70 T=TIME
80 H=INT(T/3600)
90 M=INT(T/60)
100 S=TMOD60
110 PRINTUSING"##:## '##";H;M;S
120 GOTO60
```

TRON
TROFF

TIPO	Comando.
FORMATO	**TRON** **TROFF**
EXEMPLO	**TRON** **TROFF**
USO	Ativar ou desativar a função de análise passo a passo da execução do programa. **TRON** ativa e **TROFF** desativa a função de execução passo a passo. Esse comando pode ser executado tanto no modo direto como no indireto, e ele habilita um "flag" que imprime cada número de linha do programa conforme ele vai sendo executado. Os números aparecem entre parênteses quadrados. O "flag" é desabilitado por meio do comando **TROFF** (ou através do comando NEW).

PROGRAMA EXEMPLO

```
10 FORI=1TO3
20 PRINTI
30 NEXT
40 END
```

USR

TIPO Função.
FORMATO **USR** [⟨ dígito ⟩] (X)
EXEMPLO T = **USR** 3 (J)
USO Chamar uma sub-rotina do usuário, em linguagem assembly.
⟨ dígito ⟩ é um número inteiro compreendido entre zero e 9, que indica qual é a sub-rotina em linguagem de máquina, que deve ser seguida. Se o ⟨ dígito ⟩ não vier indicado, assume-se o valor zero.
O primeiro endereço de cada sub-rotina em linguagem de máquina deverá ser indicado pela instrução "DEFUSR".
(X) é o valor fornecido pela subrotina em linguagem de máquina. Esse valor deverá vir indicado em uma das seguintes formas:

1. Para os valores alfanuméricos
O endereço de memória &HF663 contém o valor 3.
Os endereços de memória &HF7F8 e &HF7F9* contém o endereço do descritor do valor alfanumérico. O descritor compõe-se de 3 bytes. O primeiro byte indica o comprimento do valor alfanumérico, o segundo e o terceiro byte indicam o endereço de memória desse valor.

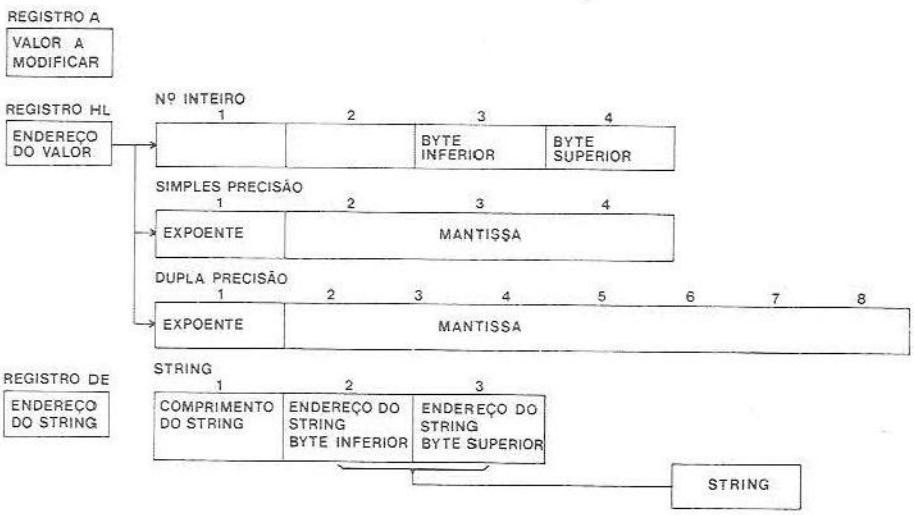
2. Para os valores numéricos inteiros.
O endereço de memória &HF663 contém o valor 2.
Os endereços de memória &HF7F8 e &HF7F9* contém o valor do número inteiro.

3. Para os valores de precisão simples
O endereço de memória &HF663 contém o valor 4.
Os endereços de memória de &HF7F6 até &HF7F9* contém o valor de precisão simples.

4. Para os valores de precisão dupla
O endereço de memória &HF663 contém o valor 8.
Os endereços de memória de &HF7F6 até &F7FD* contém o valor de dupla precisão.

* (Área FAC, vide Manual do Usuário – Apêndice Linguagem de Máquina).

Os valores que retornam da sub-rotina em linguagem de máquina ao BASIC deverão ser inseridos na memória da mesma forma descrita acima, nos 4 casos diferentes.
Utiliza-se a instrução "CLEAR" para reservar o espaço de memória necessário para a subrotina em linguagem de máquina.



PROGRAMA EXEMPLO

```
10 CLEAR200,&HEFFF
20 AB=&HF000
30 FORI=ABTOAB+9
40 READA$:A=VAL("&H"+A$)
50 POKEI,A
60 NEXTI
70 DEFUSR=&HF000
80 PRINT"Introduzir um número inteiro";
90 PRINT" entre -32768 e 32766":INPUTA%
100 PRINT"Número inteiro=";A%
110 R=USR(A%)
120 PRINT"O resultado é igual ao número";
130 PRINT" inteiro mais 1:";R
140 END
150 DATA23,23,4e,23,46,03,70,2b,71,c9
```

VAL

TIPO Função.
FORMATO VAL ((X$))
EXEMPLO VAL ("2")
USO Fornece o valor numérico do "string" X$.
A função VAL ignora os espaços e os caracteres de controle do argumento. Exemplo:

```
PRINT VAL (" -7")
-7
OK
```

Para obter a inversa, ou seja, a representação alfanumérica de uma expressão numérica, usa-se STR$.

PROGRAMA EXEMPLO

```
 10 NUMERO=12.34216#:PRINTNUMERO
20 A$=STR$(NUMERO)
30 H%=INSTR(A$,".")
40 IFH%<1GOTO80
50 H%=H%+2
60 B$=LEFT$(A$,H%)
70 NUMERO=VAL(B$):PRINTNUMERO
80 END
10 A=10:B=20
20 C=VARPTR(A):D=VARPTR(B)
30 IFC<0THENC=C+65536!
35 IFD<0THEND=D+65536!
40 C$="0000"+HEX$(C)
45 D$="0000"+HEX$(D)
50 PRINTRIGHT$(C$,4)
55 PRINTRIGHT$(D$,4)
60 END
```

VARPTR

TIPO Função.

FORMATO **VARPTR** (⟨nome da variável⟩)
(⟨número do arquivo⟩)

EXEMPLO
1) PRINT HEX$ (**VARPTR** (A))
2) BVAD = **VARPTR** (# 1)

USO

Fornece o endereço de memória do primeiro byte de uma variável ou do primeiro byte do bloco de controle de um arquivo.

No primeiro caso (endereço de uma variável), deve-se em primeiro lugar, designar um valor à variável. Caso contrário aparecerá a mensagem de erro "FUNÇÃO ILEGAL".

Na função **VARPTR** poderá ser introduzido qualquer tipo de variável (inteira, simples precisão, dupla precisão ou string), e até variáveis dimensionadas (definidas através da instrução DIM).

A todas as variáveis não dimensionadas deverá ser designado um valor antes de solicitar o endereço de uma variável dimensionada, através da função **VATPTR**. Isto é necessário porque os endereços das variáveis dimensionadas variam cada vez que é utilizada uma nova variável não dimensionada.

A função **"VARPTR"** é normalmente utilizada para obter o endereço de memória de uma variável que deve ser passada para uma subrotina em linguagem de máquina.

Para passar um "arranjo" normalmente se utiliza a função na forma **VARPTR** (A(Ø)), de forma que o valor fornecido será o do elemento de menor endereço.

No segundo caso, a função fornece o primeiro endereço do bloco de controle de um arquivo. ⟨número de arquivo⟩ deverá ser o número que foi aberto na instrução "OPEN".

O resultado da função **"VARPTR"** é um número compreendido entre −32768 e 32767.

Se vier indicado um número negativo, bastará somá-lo a 65536 para se obter o endereço real.

PROGRAMA EXEMPLO

```
10 A=10:B=20
20 C=VARPTR(A):D=VARPTR(B)
30 IFC<0THENC=C+65536!
35 IFD<0THEND=D+65536!
40 C$="0000"+HEX$(C)
45 D$="0000"+HEX$(D)
50 PRINTRIGHT$(C$,4)
55 PRINTRIGHT$(D$,4)
60 END
```

VDP

TIPO Variável do sistema.
FORMATO VDP (⟨ n ⟩)
EXEMPLO A = VDP (8)
USO Contém o conteúdo dos registros do processador de vídeo (VDP). ⟨ n ⟩ é o número do registro e deverá ser um inteiro compreendido entre zero e 8. Para n = 0 até 7 VDP especifica o valor atual dos registros "WRITE-ONLY" do processador de vídeo. Para n = 8 especifica o registro de STATUS do processador de vídeo.
O conteúdo do registro 8 não pode ser mudado. Ele poderá somente ser chamado.

NOTA: Essa variável só deverá ser utilizada se o usuário estiver amplamente familiarizado com o funcionamento do VDP (processador de vídeo).

PROGRAMA EXEMPLO

```
10 FOR I=0 TO 8
20 A=VDP(I)
30 B$="00000000"+BIN$(A)
40 PRINT RIGHT$(B$,8)
50 NEXT
60 END
```

VPEEK

TIPO Função.
FORMATO **VPEEK** (⟨ X ⟩)
EXEMPLO A% = **VPEEK** (1)
USO Fornece o conteúdo de um byte da memória de vídeo.
A função VPEEK fornece o valor decimal do conteúdo de um endereço de memória de vídeo e terá por tanto um valor compreendido entre Ø e 255.
⟨ X ⟩ é o endereço da memória de vídeo, e deverá ter um valor compreendido entre Ø e 16383.
Para inserir um valor dado em um endereço determinado, usa-se a instrução "VPOKE".

NOTA: Essa função deverá ser utilizada só se o usuário estiver amplamente familiarizado com o funcionamento do VDP (processador de vídeo).

PROGRAMA EXEMPLO

```
10 A=VPEEK(0)
20 A$="00"+HEX$(A)
30 PRINT RIGHT$(A$,2)
40 END
```

VPOKE

TIPO Instrução.
FORMATO **VPOKE** ⟨X⟩,⟨Y⟩
EXEMPLO **VPOKE** I%, &HFF
USO Inserir um valor determinado em um byte da memória de vídeo. ⟨X⟩ é o endereço da memória de vídeo, e deverá ter um valor compreendido entre zero e 16383.
⟨Y⟩ é o valor que será escrito no byte indicado, e seu valor deverá estar compreendido entre Ø e 255.
Para se obter um valor proveniente de um endereço da memória de vídeo, usa-se a função "VPEEK".

NOTA: Essa instrução deverá ser utilizada só se o usuário estiver amplamente familiarizado com o funcionamento do VDP (processador de vídeo).

PROGRAMA EXEMPLO

```
10 VPOKE0,(VPEEK(0))
20 END
```

WAIT

TIPO Instrução.
FORMATO **WAIT** ⟨PORT X⟩,⟨EXPRESSÃO Y⟩[,⟨EXPRESSÃO V⟩]
EXEMPLO **WAIT** 1, &H22, &H22.
USO Controlar o estado de um "port" de entrada de máquina.
A instrução WAIT provoca a suspensão da execução do programa enquanto um dos "port" de entrada da máquina desenvolve um esquema de bits específico.
O dado lido na ⟨PORT X⟩ é confrontado primeiro com a expressão inteira ⟨Y⟩ (através da operação XOR), e depois com a expressão inteira ⟨V⟩ (através de AND).
Se o resultado dessas operações for zero, o BASIC volta atrás e lê o dado do port novamente. Em qualquer outra circunstância, continuará com o programa.
Se ⟨V⟩ não vier indicado, assumirá-se automaticamente que seu valor é Ø.
REFERÊNCIAS "INP" e "OUT".

WIDTH

TIPO Instrução.
FORMATO **WIDTH** ⟨X⟩
EXEMPLO **WIDTH 30**
USO Indica o número de posições disponíveis em uma linha da tela, nos modos texto 1 e 2.
⟨X⟩ indica o número de posições por linha, e deverá ser um número inteiro compreendido entre 1 e 40 no modo texto 1 ou entre 1 e 32 no modo texto 2.

PROGRAMA EXEMPLO

```
10 FORI=1TO30
20 WIDTHI
30 PRINTSTRING$(I,"*");"WIDTH";I
40 FORJ=0TO500
50 NEXTJ,I
```

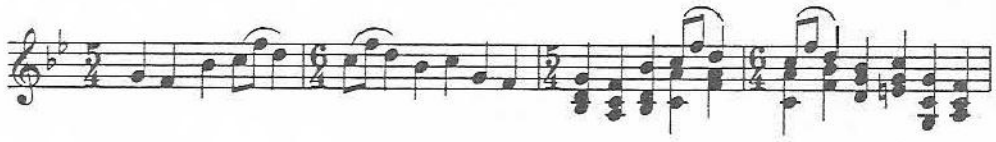
HOTBIT PROGRAMAS EXEMPLO

Neste capítulo apresentaremos alguns programas montados com instruções e funções explicadas no Capítulo 2, e uma breve explicação dos mesmos.
Cada um desses programas poderá ser utilizado como uma unidade independente, porém será interessante para o usuário tentar introduzir modificações ao mesmo ou tentar usá-los como parte de programas maiores.

INSTRUÇÃO "PLAY"

(... execução de um trecho musical de "QUADROS DE UMA EXPOSIÇÃO").

EXPLICAÇÃO Vamos executar um trecho musical da obra de Mussorgsky "Quadros de uma Exposição" utilizando a instrução "PLAY".



Os dados musicais são montados por trechos dentro dos diversos "DATA" e lidos através da instrução "READ".

PROGRAMA EXEMPLO

```
10 '*****************************
20 '*                           *
30 '*Quadros de uma exposicao***
40 '*                           *
50 '*****************************
60  CLS:PRINT"Trecho musical"
70 READA$,B$,C$
80 IFA$=""THENEND
90 PLAYA$,B$,C$
100 GOTO70
110 '
120 ' DADOS
130 '
140 DATA V13,V10,V10
150 DATA O4L4GFA+O5C8F8D
160 DATA RRRRR
170 DATA RRRRR
180 DATA C8F8DO4B-O5CO4GF
190 DATA RRRRR
200 DATA RRRRR
210 DATA O4L4GFA+O5C8F8D
220 DATA O4L4DCDAA
230 DATA O3L4B-AB-O4CF
240 DATA C8F8DO4B-O5CO4GF
250 DATA O4L4AB-GGCC
260 DATA O4L4CFDEO3GA
270 DATA FGDF8G8C
280 DATA RRRRR
290 DATA RRRRR
300 DATA G8A8FO5FDC8O4B-8F
310 DATA RRFFFF
320 DATA RRO3L4FB-GF
330 DATA O4L4FGDF8G8E-
340 DATA RRRRR
350 DATA RRRRR
```

```
360 DATA 04L4B-805C804A-05A-FE-8D-804A-
370 DATARRA-A-A-A-
380 DATARR03L4A-04D-03B-A-
390 DATA04L4A-B-A-B-805C8E-804B-8A-
400 DATA03L4A-B-A-B-804C8E-803B-8A-
410 DATA02L4G-2FG-G-03A-
420 DATA05L8D-E-FA-G-FE-G-FD-E-4
430 DATA04L8A-05CD-04A-05E-D-C04G-05D-04D-05C4
440 DATA03L8FE-D-4E-FA-4B-4A-4
450 DATA04L4A-B-A-L8B-05CE-04B-
460 DATA03L4A-B-A-L8B-04CE-03B-
470 DATA02G-2L4FG-G-8R8
480 DATA05L4CDCL8DFGDL4C
490 DATA02B-2L4AB-B-03B-
500 DATA02B-2L4AB-B-03B-
510 DATA05L8FGA06C05B-AGB-AFG4
520 DATA05L8CEFR8GFER8FR8E4
530 DATA03L8AGF4GA04C4D4C4
540 DATA05L4A8E8FADAD
550 DATA05L4CDC04B-05C04B-
560 DATA03L4AB-FGFG
570 DATA05L4F8C8DFDF8C8D
580 DATA04AB-AB-AB-
590 DATA03L4DGDGDG
600 DATA05L4C04AB-05C04AB-805D8
610 DATA04G8E8FFG8E8FF
620 DATA03L4CFGCFG
630 DATA05L4C04A05CFL8E-DC04B-
640 DATA04GFGRL8B-R8AF
650 DATA03L4C02F03C02FGA8B-8
660 DATA05L4CDFG8B-8FG
670 DATA04L4FF05CE-8R804FG
680 DATA02L4AB-AGFG
690 DATA05L4FL8E-DC04B-L405CDF
700 DATA04L4FL8B-R8AFL4FF05C
710 DATA02L4FGA8B-8AB-A
720 DATA05L4G8B-8FGF04GF
730 DATA05E-8R804FGF03GF
740 DATA02L4GFGF03GF
750 DATA05L4G8B-8FGF04GF
760 DATA05E8R804FGFE-C
770 DATA03L4C02FGF03E-F
780 DATA04L4B-05C8F8DC8F8D04B-
790 DATA04FA8R8B-A8R8B-F
800 DATA03DC02B-AG03G
810 DATA05L4C04GFGFB-
820 DATA04GECDCD
830 DATA03L4CEF02B-AG
840 DATA05L4C8F8D04B-05E-C04B-
850 DATA04A8R8B-G05C04AF
860 DATA02L4FB-03GCFB-
870 DATA""
880 DATA""
890 DATA""
```

INSTRUÇÃO "SOUND"

(... montagem de efeitos sonoros).

EXPLICAÇÃO Como a instrução **SOUND** permite a geração de sons através de uma determinação precisa da freqüência, forma da envoltória e freqüência de ruído, ela resulta muito útil na montagem de efeitos sonoros.
Nesse programa exemplo veremos somente algumas possibilidades. O usuário poderá montar no seu próprio programa só a parte do som necessário ou mudá-lo tendo este como referência.

PROGRAMA EXEMPLO

```
10 '******************
20 '*                *
30 '*EFEITOS SONOROS*
40 '*                *
50 '******************
60 '
70 'DEFINIR OS REGISTROS DE SOM
80 '
90 '
100 FORI=1TO9
110 PRINT"Pressione uma teola qualquer"
120 A$=INPUT$(1)
130 READA$:PRINTA$
140 FORJ=0TO13
150 READ DT
160 SOUNDJ,DT
170 NEXTJ
180 NEXTI
190 END
200 '
210 'PARTIDA!!
220 '
230 DATA PARTIDA!!
240 DATA0,0,0,0,0,0,1,7
250 DATA16,16,16,100,100,0
260 '
270 'APITO DO TREM
280 '
290 DATA APITO DO TREM
300 DATA84,0,52,0,151,0,0,56
310 DATA14,14,14,200,10,11
320 '
330 'PRIMERA VELOCIDADE
340 '
350 DATA PRIMERA VELOCIDADE
360 DATA0,0,0,0,255,15,16,3
370 DATA16,7,16,90,20,8
380 '
390 'SEGUNDA VELOCIDADE
400 '
```

```
410 DATA SEGUNDA VELOCIDADE
420 DATA0,0,0,0,255,15,12,3
430 DATA16,7,16,90,8,8
440 '
450 'TERCEIRA VELOCIDADE
460 '
470 DATA TERCEIRA VELOCIDADE
480 DATA 0,0,0,0,255,15,8,3
490 DATA16,7,16,90,4,8
500 '
510 'TRILHA
520 '
530 DATA TRILHA
540 DATA100,0,110,0,180,0,0,56
550 DATA16,16,16,90,8,8
560 '
570 'HELICOPTERO 1
580 '
590 DATA DENTRO DO HELICOPTERO 1
600 DATA0,0,0,0,24,0,22,3
610 DATA2,2,16,90,2,12
620 '
630 'HELICOPTERO 2
640 '
650 DATA DENTRO DO HELICOPTERO 2
660 DATA0,0,100,3,24,0,16,1
670 DATA0,16,16,90,2,12
680 '
690 'EXPLOSAO
700 '
710 DATA BOMB!!
720 DATA0,0,0,0,0,0,21,247
730 DATA16,0,0,100,60,0
```

INTERRUPÇÃO POR TIMER/IMPRESSÃO DE LETRAS NA TELA NO MODO GRÁFICO

(... Rotina para treino de datilografia).

EXPLICAÇÃO O programa exemplo a seguir serve como treino de datilografia, a fim do usuário poder verificar qual é o número de letras que pode digitar por minuto.
Uma vez começado o treino, passado um minuto, habilita-se a interrupção por timer, e o programa pula para a rotina de processamento de interrupção, na qual avalia o desempenho da datilografia.
Para gerar a impressão das letras na tela, no modo gráfico, abre-se o arquivo "GRP:". Usando esse método, o usuário poderá mudar as cores das letras conforme desejar.

USO (1) Começando a execução, será visualizada a mensagem inicial. Após pressionar uma tecla qualquer começará a aparecer na tela palavras que você deverá repetir. Cada erro acionará um sinal sonoro (Pi-pi), até você pressionar a tecla correta.
(2) Passado um minuto (1 min.) será visualizada a mensagem correspondente à avaliação do seu desempenho como datilógrafo.

PROGRAMA EXEMPLO

```
10 '*************************
20 '*                       *
30 '*TREINO DE DATILOGRAFIA*
40 '*                       *
50 '*************************
60 SCREEN2:CLEAR1000:DIMQ$(100)
70 OPEN"GRP:"FOROUTPUTAS#1
80 DEFINTA-Z:A=RND(-TIME)
90 '
100 '***LEITURA DE PALAVRAS***
110 '
120 Q=0
130 READQ$(Q):IFQ$(Q)="="THENGOTO150
140 Q=Q+1:GOTO130
150 '
160 '***MENSAGEM INICIAL***
170 '
180 COLOR1,14,14:CLS
190 DRAW"BM50,24":PRINT#1,"EXERCICIO DE DATILOGRAFIA"
200 DRAW"BM30,40":PRINT#1,"Pressione uma tecla qualquer"
210 IFINKEY$=""THEN210
220 IFINKEY$<>""THEN220
230 ONINTERVAL=3600GOSUB510
240 INTERVALON
250 '
260 '***IMPRESSAO***
270 '
280 SCREEN3:COLOR1,14,14
290 QS$=Q$(RND(1)*Q):LG=LEN(QS$)
300 DRAW"BM56,56":PRINT#1,QS$
310 '
320 '***DIGITACAO***
```

```
330
340 DRAW"BM56,104"
350 FORI=1TOLG
360 A$=INKEY$:IFA$=""THEN360
370 IFA$=MID$(QS$,I,1)THENGOTO410
380 COLOR8:BEEP:GOSUB450
390 DRAW"BM-32,0"
400 MS=MS+1:GOTO360
410 COLOR4:GOSUB450
420 NEXTI
430 WD=WD+1:PLAY"L3204CG":GOTO280
440 '**VISUALIZACAO DE 1 LETRA**
450 LINE-STEP(32,32),14,BF
460 DRAW"BM-32,-32":PRINT#1,A$;
470 RETURN
480 '
490 '***Passado 1 minuto***
500 '
510 SCREEN1:PLAY"L32C16DEFGAB05C"
520 LOCATE6,8:PRINT"Numero de palavras introduzidas=";WD
530 LOCATE6,10:PRINT"Numero de erros=";MS
540 CLOSE#1:COLOR15,4,7
550 IFPLAY(0)THEN560
560 IFINKEY$<>""THEN550
570 END
580 '***Palavras dados***
590 '
600 DATA DIG-1,BASIC,FRUTA,RATO,BOLSA
610 DATA RESET,IF,SPRITE,TIME,LOAD,ERASE
620 DATA GOSUB,BRANCO,SOUND,RETURN,ALO!,CANIL
630 DATA BARCO,TOKIO,PRINT,MANUAL,GATO,AMOR
640 DATA FOME,ERRO,DATA,DRAW,CALL,LOBO,=
```

SPRITE

... Jogo de Tênis (Paredão).

EXPLICAÇÃO Utilizando a variável SPRITE$, vamos fazer um jogo de tênis contra um paredão. Você deverá rebater com a raquete amarela a bola branca que está pulando na tela.
No programa, a bola e a raquete tem as suas formas desenhadas através da variável SPRITE$, dando-se a sua visualização e movimentação através da instrução PUT SPRITE. E ainda foi feito de forma que o sentido direcional da bola muda toda vez que a raquete bate nela, usando-se para isso a interrupção por time provocada pela colisão de sprites (ON SPRITE GOSUB).
Você poderá ainda melhorar este programa mudando a velocidade da bola, colocar um obstáculo e fazer com que cada vez que a bola bata nele apareçam na tela os pontos obtidos até esse momento.

USO Na parte inferior da tela existe uma raquete amarela. Pressionando a tecla ← ela se move para esquerda e pressionando → se move para direita. Com essa raquete amarela você deverá rebater a bola que vem caindo, movendo a raquete com velocidade.

PROGRAMA EXEMPLO

```
10 '**************
20 '*              *
30 '*  TENIS       *
40 '*              *
50 '**************
60 SCREEN2,2:COLOR15,4,7:CLS
70 DEFINTA-Z:A=RND(-TIME)
80 '
90 '***DESENHO DOS SPRITES****
100 '
110 FORI=1TO2
120 A$=""
130 FORJ=1TO32
140 READA:A$=A$+CHR$(A)
150 NEXTJ
160 SPRITE$(I)=A$
170 NEXTI
180 '**************RAQUETE***
190 DATA0,0,0,0,0,0,0,0
200 DATA0,0,0,0,255,255,255,255
210 DATA0,0,0,0,0,0,0,0
220 DATA0,0,0,0,255,255,255,255
230 '**************BOLA******
240 DATA1,7,31,63,63,127,127,255
250 DATA255,127,127,63,63,31,7,1
260 DATA128,224,248,252,252,254,254,255
270 DATA255,254,254,252,252,248,224,128
280 '
290 '***POSIÇÃO INICIAL RAQUETE E BOLA**
300 '
310 ONSPRITE GOSUB610
```

```
320 SPRITEON
330 X1=120:S1=2
340 PUTSPRITE1,(X1,175),10,1
350 MX=INT(RND(1)*2)*2*S1-S1:MY=S1
360 X2=INT(RND(1)*239):Y2=0
370 PUTSPRITE2,(X2,Y2),14,2
380 '
390 '***MOVIMENTO DA RAQUETE****
400 '
410 CR=STICK(0)
420 IFCR=3ANDX1<239THENX1=X1+S1:GOTO440
430 IFCR=7ANDX1>0THENX1=X1-S1
440 PUTSPRITE1,(X1,175)
450 '
460 '***MOVIMENTO DA BOLA***
470 '
480 X2=X2+MX
490 IFX2<10RX2>238THENMX=MX*-1:PLAY"L64C"
500 Y2=Y2+MY
510 IFY2<176THENGOTO550
520 PLAY"L64GFEDC"
530 PUTSPRITE2,(X2,209)
540 GOTO350
550 IFY2<1THENMY=MY*-1:PLAY"L64G"
560 PUTSPRITE2,(X2,Y2)
570 GOTO410
580 '
590 '***REBATER A BOLA*****
600 '
610 MY=S1*-1:X2=X2+MX
620 X2=X2+MX
630 IFX2<10RX2>238THENMX=MX*-1
640 Y2=Y2+MY:PLAY"L64CG"
650 PUTSPRITE2,(X2,Y2)
660 RETURN
```

INSTRUÇÃO DEF FN

(... Montar a função de definição do usuário).

EXPLICAÇÃO No caso de ter uma fórmula muito utilizada dentro do programa, o usuário poderá deixá-la definida através da instrução DEF FN, assim não será necessário repeti-la cada vez que for utilizada. Bastará indicar o nome da função e o argumento para poder calcular.
Aqui, como exemplo, vamos apresentar uma função de definição do usuário que executa o cálculo da seguinte forma. Durante a execução do programa você introduz o argumento (informação de entrada) pelo teclado, e o resultado sai na tela.

Função definida pelo usuário	Argumento	Resultado	Fórmula
Distância entre 2 pontos	b		
	c	a	$a = \sqrt{b^2 + c^2 - 2bc\cos A}$
		∠A	
seno (°)	d	sin (d)	sin(d) = sin(π/180*d)
tangente (°)	d	tan (d)	tan(d) = tan (π/180*d)
secante (°)	d	sec (d)	sec(d) = 1/cos(π/180*d)
cossecante (°)	d	cosec (d)	cosec(d) = 1/sin(π/180*d)

USO

(1) Executando o programa será visualizado na tela o Menu para escolher qual é o cálculo a efetuar. O usuário deverá introduzir o código da função que deseja.
(2) Respondendo à mensagem visualizada na tela, o usuário deverá introduzir o argumento. Em seguida aparecerá o resultado na tela. Assim, se o usuário pressionar a barra espaçadora, será visualizado o menu novamente.

PROGRAMA EXEMPLO

```
10 '***VALOR DEFINIDO PELO USUARIO***
20 DEFFNA(A,B,C)=SQR(B^2+C^2-2*B*C*COS(A*3.14159265#/180))
30 DEFFNSN(D)=SIN(3.14159265#/180*D)
40 DEFFNTN(D)=TAN(3.14159265#/180*D)
50 DEFFNSC(D)=1/COS(3.14159265#/180*D)
60 DEFFNCO(D)=1/SIN(3.14159265#/180*D)
70 '***MENU****
80 CLS:LOCATE6,5:PRINT"1:DISTANCIA ENTRE 2 PONTOS"
90 LOCATE6,7:PRINT"2:SIN(Gr.)"
100 LOCATE6,9:PRINT"3:TAN(Gr.)"
110 LOCATE6,11:PRINT"4:SEC(Gr.)"
120 LOCATE6,13:PRINT"5:COSEC(Gr.)"
130 LOCATE9,15:INPUT"Que cálculo você quer fazer";X
140 ONXGOSUB190,260,300,340,380:GOTO160
150 GOTO80
160 PRINT"PARA CONTINUAR PRESSIONE BARRA ESPAÇADORA"
165 IFINKEY$<>" "THEN165
```

```
170 GOTO80
180 '**ROTINA DE DISTANCIA ENTRE 2 PONTOS**
190 CLS:LOCATE1,5:INPUT"COMPRIMENTO DO SEGMENTO B";B
200 LOCATE1,7:INPUT"COMPRIMENTO DO SEGMENTO C";C
210 LOCATE1,9:INPUT"ANGULO B/C";A
220 Q=FNA(A,B,C)
230 LOCATE1,11:PRINT"DISTANCIA DO SEGMENTO A=";Q
240 RETURN
250 '***ROTINA DO SIN(Gr.)***
260 CLS:LOCATE3,5:INPUT"ANGULO(Gr.)";D
270 LOCATE3,7:PRINT"SIN(";D;")=";FNSN(D)
280 RETURN
290 '***ROTINA DA TAN(Gr.)***
300 CLS:LOCATE3,5:INPUT"ANGULO(Gr.)";D
310 LOCATE3,7:PRINT"TAN(";"Gr.";")=";FNTN(D)
320 RETURN '
330 '***ROTINA DA SEC(Gr.)***
340 CLS:LOCATE3,5:INPUT"ANGULO(Gr.)";D
350 LOCATE3,7:PRINT"SEC(";D;")=";FNSC(D)
360 RETURN
370 '***ROTINA DA COSEC(Gr.)***
380 CLS:LOCATE3,5:INPUT"ANGULO(Gr.)";D
390 LOCATE3,7:PRINT"COSEC(";D;")=";FNCO(D)
400 RETURN
```

FUNÇÃO USR

(Ordenação de strings em ordem alfabética).

EXPLICAÇÃO Usando uma rotina em linguagem de máquina, vamos modificar a ordenação dos strings armazenados em A$(1) ~ A$ (N), recolocando-os em ordem alfabética (segundo a ordem dos códigos ASCII de caracteres), a alta velocidade.
Neste programa o conteúdo de A$(1) até A$(N), é introduzido através do teclado em qualquer ordem. O usuário poderá pensar em outros aplicativos usando a mesma rotina em linguagem de máquina.

USO (1) Executando o programa será visualizada uma mensagem.
"O Nº de palavras é = ". O usuário poderá introduzir 2 e 1000 dados. Em seguida aparecerá "comece a introduzir palavras", após o qual o usuário deverá digitar os dados sem se preocupar com a ordem.
(2) Uma vez introduzidos todos os dados aparecerá na tela a mensagem "Começou", dando início à ordenação.
Terminada a ordenação serão impressas na tela as palavras introduzidas já em ordem alfabética. Por último será visualizado o tempo gasto na ordenação em unidades de 1/60 seg. Se o número de dados for de ~ 100 levará poucos segundos. Se for ~ 1000 já levará alguns segundos.

PROGRAMA EXEMPLO

```
10 '*************************
20 '*                       *
30 '* ORDENAÇAO DE STRINGS *
40 '*                       *
50 '*************************
60 '
70 CLEAR6000,&HF000:DEFINTA-Z
80 DEFUSR=&HF000:WIDTH30
90 INPUT"O Nº de palavras é ";N:DIMA$(N)
100 '
110 '*** Leitura da rotina em ling./máquina ***
120 '
130 FORI=0TO&H73
140 READA$:POKE&HF000+I,VAL("&h"+A$)
150 NEXT
160 U=0:V=0
170 '
180 '*** Introduçãо de palavras ***
190 '
200 PRINT"comece a introduzir palavras"
210 FORI=1TON
220 PRINT"A$(";I;")=":INPUTA$(I)
250 NEXT
260 '
270 '*** ordenaçãо ***
280 '
290 A$(0)=" "+"":V=VARPTR(A$(0))
300 PRINT"Começou!":TIME=0
310 V=USR(V)
320 U=TIME
330 FORI=1TON
```

```
340 PRINTUSING"&     &";A$(I);
350 NEXT:PRINT:PRINT"A ordena¢ão demorou "
355 PRINTU;"/60 SEGUNDOS"
360 END
370 '
380 '***Rotina em lingagem de máquina***
390 '
400 DATA23,23,5E,23,56,D5,DD,E1
410 DATADD,5E,FE,DD,56,FF,1B,DD
420 DATAE5,DD,46,00,DD,4E,03,DD
430 DATA6E,04,DD,66,05,E5,FD,E1
440 DATA7E,DD,6E,01,DD,66,02,96
450 DATA38,10,20,3A,05,28,37,0D
460 DATA28,08,23,FD,23,FD,7E,00
470 DATA18,ED,DD,2B,DD,2B,DD,2B
480 DATADD,7E,06,DD,46,03,DD,70
490 DATA06,DD,77,03,DD,7E,07,DD
500 DATA46,04,DD,70,07,DD,77,04
510 DATADD,7E,08,DD,46,05,DD,70
520 DATA08,DD,77,05,18,AB,DD,E1
530 DATADD,23,DD,23,DD,23,1B,7A
540 DATAB3,20,9C,C9
```

INSTRUÇÃO LINE

(... Desenho de uma árvore)

EXPLICAÇÃO Vamos desenhar na tela uma árvore com seus galhos nascendo da terra e crescendo em direção ao céu. Variando o comprimento e o ângulo do galho o usuário poderá modificar a árvore.

PROGRAMA EXEMPLO

```
10 '***********
20 '*         *
30 '* ARVORE  *
40 '*         *
50 '***********
60 SCREEN2:CLS
70 LL=45  '***comprimento inicial do galho
80 KK=4/5 '***taxa de crescimento do galho
90 RR=20 '***ângulo do galho da esquerda
100 RL=20 '***ângulo do galho da direita
110 N=0:AN=90:SIZE=LL*KK:PP=ATN(1)*8/360
120 DIMI(20)
130 LINE(128,190)-STEP(0,-LL)
140 I(N)=0:AN=AN-2*RR-RL
150 AN=AN+RR+RL
160 GOSUB250:LINE-STEP(-X,-Y)
170 N=N+1:SIZE=SIZE*KK
180 IFSIZE>10THEN140
190 N=N-1:SIZE=SIZE/KK
200 GOSUB250:LINE-STEP(X,Y)
210 I(N)=I(N)+1:IFI(N)<2THEN150
220 AN=AN-RL
230 IFN<>0THEN190
240 GOTO240
250 RA=AN*PP
260 X=COS(RA)*SIZE:Y=SIN(RA)*SIZE
270 RETURN
```

ENTRADAS E SAIDAS DE ARQUIVO

(Desenhar na tela/Guardar o desenho da tela em fita)

EXPLICAÇÃO Operando a tecla cursor, desenha-se na tela no modo gráfico de Alta Resolução, comprimem-se os dados e guardam-se em fita. A tela gravada em fita poderá posteriormente ser recarregada e corrigida. A cor utilizada no desenho deste programa é: fundo azul (4), frente branca (15) e o contorno azul claro (7).

USO (1) Executando o programa será visualizada a mensagem "carrega (s/n)" que pergunta ao usuário se ele vai carregar os dados da tela que estão na fita ou se vai criar um desenho novo.
(2) No caso de efetuar o carregamento pressiona-se "S", caso contrário "N".
No caso de introduzir "S", os dados do desenho deverão ter sido previamente gravados na fita. Introduzindo "S" os dados da tela que estão na fita aparecerão na tela novamente.
(3) Esperando alguns segundos aparecerá um cursor vermelho em forma de cruz no meio da tela. Experimente ficar com a tecla ↑ pressionada por algum instante. O ponto vermelho que estava embaixo do cursor é LP (ponto de referência). Você poderá agora desenhar e apagar utilizando as teclas abaixo:

[↑] Move o cursor para cima. Se estiver no modo de desenho, será traçada uma linha conforme o seu movimento.

[↓] Move o cursor para baixo. Se estiver no modo de desenho, será traçada uma linha conforme o seu movimento.

[→] Move o cursor para direita. Se estiver no modo de desenho, será traçada uma linha conforme o seu movimento.

[←] Move o cursor para esquerda. Se estiver no modo de desenho traçará uma linha conforme o seu movimento.

Pressionando [↑] + [→] , [↑] + [←] , [↓] + [←] , [↓] + [→] o cursor mover-se-á na diagonal.

[INS] Posiciona o modo de desenho. Depois disto, ao mover o cursor, será traçada uma linha conforme o seu movimento.

[DEL] Posiciona o modo apagamento. Depois disto, ao mover o cursor, será apagada a linha conforme o seu movimento.

[BARRA ESPAÇADORA] Posiciona o modo de movimento do cursor. Dentro desse modo, pode traçar linhas com [F4] e [F5] e pintar o desenho com [TAB] .

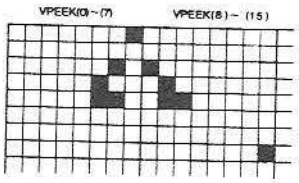
[ESC] Muda o LP para a posição do cursor.

[F4] Sem mudar o LP, traça a linha do LP até a posição do cursor. Nesse momento, se estiver no modo apagamento, apagará a linha ou os pontos traçados entre LP e a posição do cursor.
Se mover o cursor pressionando a tecla, o usuário poderá pintar ou apagar uma área grande.

F5	Traça a linha do LP até a posição do cursor, e se estiver no modo apagamento, apaga do LP até o cursor. Move o LP para a posição do cursor.
TAB	Pinta (chapado) o interior da linha que cerca a posição do cursor.
RETURN	Guarda o Arquivo de Dados na tela, sob o nome "dados" em fita cassete.

NOTA: Os dados da tela ficam guardados da seguinte forma:

CANTO → SUPERIOR ESQUERDO DA TELA



VPEEK valor de (0) ~(15)

1, 0, 2, 4, 6, 0, 0, 0, 0, 0, 128, 64, 96, 0, 0, 1

Forma em que os dados são guardados

1, 0, 0, 2, 4, 6, 0, 4, 128, 64, 96, 0, 1, 1

(1 x 0) (5 x 0)

(Como é contado a partir de 0, se houver 1 zero, virá indicado como 0,0)

Como normalmente o desenho da tela tem bastante área vazia, guardando na fita segundo esse método diminue-se bastante a quantidade de dados.

PROGRAMA EXEMPLO

```
10 '************************
20 '*                      *
30 '* editor de desenhos   *
40 '*                      *
50 '************************
60 '
70 DEFINTA-Z:DIMA$(2)
80 A$(0)="":A$(1)="@XPT"+CHR$(10)
90 A$(1)=A$(1)+CHR$(1)+CHR$(2)+CHR$(12)
100 A$(2)="@x"+CHR$(160)+"'A"+CHR$(2)
110 A$(2)=A$(2)+CHR$(5)+CHR$(9)
120 INPUT"carrega(s/n)";Y$:SCREEN2
130 IFY$<>"S"ANDY$<>"s"THEN260
140 '
150 '*** carrega dados ***
160 '
```

```
170 OPEN"dados"FORINPUTAS#1:AD=0
180 INPUT#1,A
190 IFA=0THENINPUT#1,Z:AD=AD+Z
200 IFA<>0THENVPOKEAD,A
210 AD=AD+1
220 IFAD<6144THEN180ELSECLOSE
230 '
240 '*** montar desenho ***
250 '
260 FORI=8192TO14335
270 VPOKEI,244:NEXT
280 A$=STRING$(3,16)
290 SPRITE$(0)=A$+CHR$(238)+A$+CHR$(0)
300 A$=STRING$(3,0)
310 SPRITE$(2)=A$+CHR$(24)+CHR$(24)+A$
320 X=128:Y=96:XS=X:YS=Y:F=0
330 '
340 '*** sensor de teclas ***
350 '
360 OUT170,(INP(170)AND&HF0)OR8
370 A=INP(169)XOR255
380 OUT170,(INP(170)AND&HF0)OR7
390 B=INP(169)XOR255
400 X=X-((AANDl6)=16)*(X>0)
410 X=X+((AAND128)=128)*(X<255)
420 Y=Y-((AAND32)=32)*(Y>0)
430 Y=Y+((AAND64)=64)*(Y<191)
440 PUTSPRITE0,(X-3,Y-3),15
450 PUTSPRITE1,(248,0),11
460 PUTSPRITE2,(XS-3,YS-3),6
470 N!=(AAND15)/4
480 IFN!<=0THEN510
490 F=N!:SPRITE$(1)=A$(F)
500 IFN!=.5THEN320
510 IFB=0THEN580
520 IFB=4THENXS=X:YS=Y:GOTO360
530 IFB>2THEN550
540 LINE(XS,YS)-(X,Y),-15*(F<2)-4*(F=2)
550 IFB=2THENXS=X:YS=Y:GOTO360
560 IFB=8THENPAINT(X,Y),15:GOTO360
570 IFB=128THENGOSUB640
580 IFF=0THEN360
590 PSET(X,Y+1),-15*(F=1)-4*(F=2)
600 GOTO360
610 '
620 '***salvar dados***
630 '
640 AD=0:OPEN"dados"FOROUTPUTAS#1
650 B=VPEEK(AD):IFB=0THEN690
660 PRINT#1,B;
670 AD=AD+1:IFAD<6144THEN650
680 CLOSE:RETURN
690 Z=0:IFVPEEK(AD+8192)=244THEN720
700 VPOKEAD,255:VPOKEAD+8192,244
710 GOTO650
```

```
720 AD=AD+1:IFAD>6143THEN780
730 B=VPEEK(AD):C=VPEEK(AD+8192)
740 IFB<>0THEN770
750 IFC=244THENZ=Z+1:GOTO720
760 VPOKEAD,255:VPOKEAD+8192,244
770 PRINT#1,0;Z;:GOTO650
780 PRINT#1,0;Z;:GOTO680
```

HOTBIT

TABELA DE MENSAGEM DE ERRO

Código	Mensagem	Descrição
1	"NEXT" SEM "FOR"	Em uma instrução "NEXT" uma variável não se corresponde com nenhuma outra previamente executada, ou então uma instrução "NEXT" que não é precedida pela FOR correspondente.
2	ERRO SINTAXE	Foi encontrada uma linha contendo uma seqüência incorreta de caracteres (por exemplo: falta fechar parênteses, comando com erro de sintaxe, sinais de pontuação erradas etc.).
3	"RETURN" SEM "GOSUB"	Foi encontrada uma instrução RETURN sem a sua correspondente GOSUB prévia.
4	SEM "DATA"	Foi executada uma instrução READ quando já não existe instrução DATA com dados ainda não lidos durante o programa.
5	FUNÇÃO ILEGAL	Um parâmetro que está fora da faixa foi passado para uma função matemática ou para uma função string. Esse erro também pode acontecer como resultado de: 1. Um índice negativo ou extremamente grande. 2. Um argumento negativo ou zero usado para LOG. 3. Um argumento negativo usado para SQR. 4. Um argumento impróprio usado em MID$, LEFT$, RIGHT$, INP, OUT, PEEK, POKE, TAB, SPC, STRING$; SPACE$, INSTR$ ou ON...GOTO.
6	OVERFLOW	O resultado de um cálculo é grande demais como para ser representado no formato numérico do BASIC.
7	FALTA MEMÓRIA	O programa é muito longo, tem excesso de arquivos, tem excesso de laços FOR ou GOSUB, tem excesso de variáveis ou expressões muito complexas.
8	Nº LINHA INEXISTENTE	Faz-se referência a uma linha inexistente em uma instrução GOTO, GOSUB, IF...THEN... ELSE.
9	ÍNDICE FORA DO LIMITE	Um elemento de arranjo foi definido com um índice que está fora das dimensões pré-fixadas para esse arranjo ou então com um número errado de índices.
10	"DIM" REDEFINIDO	Duas instruções "DIM" foram dadas para o mesmo arranjo, ou uma instrução "DIM" foi dada para um arranjo depois que a dimensão "default" de 1Ø foi fixada para esse mesmo arranjo.
11	DIVISÃO POR ZERO	Foi encontrada uma divisão por zero em uma expressão ou então uma situação na qual Ø é elevado a uma potência negativa.

Código	Mensagem	Descrição
12	DIRETO ILEGAL	Uma instrução que é ilegal no modo direto foi introduzida como comando no modo direto.
13	TIPO DESIGUAL	Foi atribuído um valor numérico a uma variável string ou vice-versa, ou então, uma função que espera um argumento numérico recebe um argumento string ou vice-versa.
14	FALTA ÁREA 'STRING'	As variáveis string fizeram com que o BASIC supere o espaço de memória disponível. O BASIC posiciona os strings dinamicamente conforme as necessidades, até ficar sem memória disponível. Pode-se usar a instrução CLEAR para criar maior espaço livre.
15	"STRING" LONGA	Tentou-se criar um "string" com mais de 255 caracteres.
16	"STRING" COMPLEXA	Foi criada uma expressão "string" excessivamente longa ou complexa. A expressão deverá ser partida em expressões menores.
17	NÃO CONTÍNUO!	Tentou-se continuar um programa que: 1. Tinha sido detido devido a um erro. 2. Foi modificado durante um "BREAK" na execução, ou 3. Não existe.
18	FUNÇÃO NÃO DEFINIDA	Foi chamada uma função definida pelo usuário (FN) sem antes tê-la definido através de uma instrução DEF FN.
19	ERRO/PERIFÉRICO	Aconteceu um erro I/O em uma operação de disco, cassete, impressora ou TRC. Este é um erro fatal, isto é, o BASIC não consegue se recuperar deste tipo de erro.
20	ERRO / VERIF.	O programa na memória é diferente do programa salvo na fita cassete.
21	"RESUME"	O BASIC entrou numa rotina de tratamento de erros que não contém a instrução RESUME.
22	"RESUME" SEM "ERROR"	Foi encontrada uma instrução "RESUME" antes de ter entrado numa rotina de tratamento de erros.
23	ERRO INDEFINIDO	Não existe mensagem de erro disponível para a condição de erro existente. Isto é causado normalmente por um ERROR, com um código de erro indefinido.
24	FALTA OPERANDO	Foi encontrada uma expressão que contém um operador sem o operando correspondente a seguir.

25	LINHA MUITO LONGA	Foi introduzida uma linha com excessivo número de caracteres.
26 . . 49	ERROS INDEFINIDOS	Esses códigos não tem definições. Foram reservados para futuras expansões do BASIC.
	ERROS DO DISK-BASIC	
50	CAMPO MAIOR	Uma instrução FIELD tentou posicionar maior número de bytes dos que foram especificados para o comprimento "record" de um arquivo aleatório (255 bytes). Ou então, o fim de um buffer "FIELD" foi encontrado enquanto estava-se fazendo I/O seqüencial (PRINT#, INPUT#) a um arquivo aleatório.
51	ERRO INTERNO	Erro provocado por alguma situação não prevista pelo interpretador BASIC.
52	Nº DO ARQUIVO	Algum comando ou instrução faz referência a um arquivo para o qual não foi dado previamente o OPEN ou então o Nº está fora da faixa fixada para o Nº de arquivos, através do MAXFILES.
53	ARQUIVO NÃO EXISTE	Um comando LOAD, KILL ou OPEN faz referência a um arquivo não existente no disco.
54	ARQUIVO ABERTO	Tenta-se abrir, através de uma instrução "OPEN" um arquivo que já tinha sido aberto, ou então dá-se um KILL para um arquivo aberto.
55	FIM DO ARQUIVO	Foi executada uma instrução INPUT após todos os dados já terem sido introduzidos, ou para um arquivo nulo (vazio). Para evitar este erro, usa-se a função EOF a fim de detectar o fim do arquivo.
56	NOME ARQUIVO	Foi utilizada uma forma ilegal na definição do nome do arquivo usada em LOAD, SAVE, KILL, NAME etc.
57	COMANDO DIRETO/ARQUIVO	Foi encontrada uma instrução direta enquanto estava sendo carregado (LOAD) um arquivo em formato ASCII. O carregamento do arquivo é terminado.
58	ARQUIVO SEQUENCIAL	Uma instrução própria para o acesso aleatório é dada para um arquivo seqüencial.
59	FALTA "OPEN"	O arquivo especificado em um PRINT#, INPUT# etc., não foi aberto através do OPEN correspondente.

60 . . . 255	ERROS INDEFINIDOS: Esses erros não estão definidos. O usuário poderá fazer a sua própria definição de códigos nessa área. Essa área poderá ser utilizada por algum dos periféricos. Consulte Manual do Periférico.

HOTBIT GUIA DE REFERÊNCIA RÁPIDA

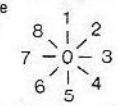
GUIA DE REFERÊNCIA RÁPIDA

NOTAÇÃO DE FORMATO

1. Todas as letras que dentro do "formato" vierem escritas em maiúsculas deverão ser digitadas através do teclado exatamente na mesma forma em que aparecem neste guia.
2. Os elementos entre parênteses angulares ⟨ ⟩ deverão ser definidos pelo usuário.
3. Os elementos entre colchetes [ ] são opcionais.
4. Todos os sinais de pontuação, à exceção dos colchetes e parênteses angulares, deverão ser digitados exatamente da mesma forma em que está indicado no formato. Isso se aplica aos pontos, vírgulas, parênteses, dois pontos, ponto e vírgula, etc.
5. "X", "Y" e "Z" ou "expressão" representam expressões numéricas, variáveis ou constantes.
6. "X$", "Y$" e "Z$" ou "string" representam expressões alfanuméricas.

Formato	Tipo	Descrição
ABS (⟨expressão⟩)	Função	Valor absoluto da expressão.
ASC (⟨string⟩)	Função	Valor ASCII do primeiro caractere da string.
ATN (⟨expressão⟩)	Função	Arco-tangente da expressão.
AUTO [⟨Nº de linha⟩] [,⟨incremento⟩]	Comando	Geração automática dos números de linha.
BASE (⟨n⟩)	Variável do sistema	Endereço atual do início de cada tabela.
BEEP	Comando	Geração do som de "beep".
BIN$ (⟨expressão⟩)	Função	String que representa o valor binário de n.
BLOAD "⟨dispositivo⟩: ⟨nome do arquivo⟩" [, R] [, ⟨offset⟩]	Comando	Carrega na memória um programa em linguagem de máquina a partir do dispositivo especificado.
BSAVE "⟨dispositivo⟩: ⟨nome do arquivo⟩", ⟨endereço inicial⟩, ⟨endereço final⟩ [, ⟨encereço de execução⟩]	Comando	Salvar um programa de linguagem de máquina localizado entre os endereços incicados através do dispositivo.
CALL ⟨nome da instrução expandida⟩ [⟨lista de argumentos⟩]	Instrução	Chamar uma instrução expandida de um cartucho ROM.
CDBL (⟨expressão⟩)	Função	Converte a expressão em um número de dupla precisão.
CHR$ (⟨expressão⟩)	Função	Fornece o caractere cujo código ASCII é = ⟨expressão⟩.
CINT (⟨expressão⟩)	Função	Converte a expressão em um número inteiro.
CIRCLE (X,Y),⟨raio⟩ [,⟨cor⟩] [,⟨ângulo inicial⟩] [,⟨ângulo final⟩] [,⟨relação de raios⟩]	Instrução	Desenha uma elipse com o centro, raio, cor, ângulos e relação entre raios especificados.
CLEAR [⟨espaço string⟩ [,⟨localização máxima⟩]]	Instrução	Zera todas as variáveis numéricas, anula todas as variáveis alfanuméricas, fecha todos os arquivos e, opcionalmente fixa o limite da memória.
CLOAD ["⟨nome do arquivo⟩"]	Comando	Carrega na memória um programa BASIC de uma fita cassete.
CLOAD? ["⟨nome do arquivo⟩"]	Comando	Compara e verifica se o programa da memória é igual ao da fita cassete.
CLOSE [[#]⟨Nº do arquivo⟩ [,[#]⟨Nº do arquivo⟩]...]	Comando	Fecha um ou mais arquivos e libera os buffers de memória associados.
CLS	Instrução	Limpa a tela gráfica.
COLOR [⟨cor do primeiro plano⟩] [,⟨cor do fundo⟩] [,⟨cor da moldura⟩]	Instrução	Define as cores da tela.
CONT	Comando	Continua com a execução de um programa após um BREAK ou um STOP.
COPY "⟨dispositivo⟩: ⟨nome do arquivo-1⟩" [TO "⟨dispositivo⟩: ⟨nome do arquivo-2⟩"]	Instrução	Copiar um arquivo em disco flexível.
COS (⟨expressão⟩)	Função	Cosseno da expressão em radianos.
CSAVE "⟨nome do arquivo⟩" [,⟨relação "baud"⟩]	Comando	Salvar em fita cassete um programa BASIC residente na memória.
CSNG (⟨expressão⟩)	Função	Converte a expressão em um número de simples precisão.
CSRLIN	Variável do sistema	Fornece a coordenada vertical do cursor.
CVI (⟨string⟩)	Função	Converte um string de 2 caracteres em um inteiro.
CVS (⟨string⟩)	Função	Converte um string de 4 caracteres, em um número de simples precisão.
CVD (⟨string⟩)	Função	Converte um string de 8 caracteres em um número de dupla precisão.
DATA ⟨lista de constantes⟩	Instrução	Armazena as constantes numéricas ou alfanuméricas que são acessadas através de uma instrução READ.
DEF FN ⟨nome⟩ [(⟨lista de parâmetros⟩)] = ⟨definição da função⟩	Instrução	Define e dá nome a uma função aritmética ou alfanumérica.
DEF INT ⟨lista de letras⟩	Instrução	Define o tipo das variáveis como inteiras, simples precisão, dupla precisão ou alfanuméricas.
DEF SNG ⟨lista de letras⟩		
DEF STR ⟨lista de letras⟩		
DEF DBL ⟨lista de letras⟩		
DEF USR [⟨dígito⟩] = ⟨expressão inteira⟩	Instrução	Define o endereço de entrada de uma sub-rotina em linguagem de máquina.
DELETE [⟨Nº de linha inicial⟩] [–⟨Nº de linha final⟩]	Comando	Apaga linhas de programa.
DIM ⟨variável⟩ (⟨índices⟩) [,variável⟩ (índices) ...]	Instrução	Especifica os valores máximos para os índices de variáveis "arranjo", e determina correspondentemente as suas posições de armazenamento.
DRAW ⟨string⟩	Instrução	Desenha uma figura na tela conforme a Linguagem Macro Gráfica.
DSKF (⟨Nº de acionador de disco⟩)	Função	Fornece o espaço livre em um disco flexível.
END	Instrução	Termina a execução de um programa, fecha todos os arquivos e volta ao nível de comandos.
EOF (⟨Nº do arquivo⟩)	Função	Assume o valor (–1) quando o fim do arquivo seqüencial é alcançado. Caso contrário vale (0).
ERASE ⟨lista de variáveis arranjo⟩	Instrução	Elimina "arranjos" de um programa.
ERL	Função	Fornece o Nº de linha do programa no qual o BASIC verificou um erro.
ERR	Função	Fornece o código do erro descoberto pelo BASIC.
ERROR ⟨código do erro⟩	Instrução	Simula a ocorrência de um erro e gera conseqüentemente a mensagem de erro correspondente.
EXP (⟨expressão⟩)	Função	Eleva a constante e à potência indicada por ⟨expressão⟩.
FIX (⟨expressão⟩)	Função	Fornece o inteiro truncado da expressão.
FIELD [#] ⟨X⟩, ⟨Y⟩ AS ⟨Y$⟩ [, ⟨Z⟩ AS ⟨Z$⟩ ...]	Instrução	Divide o buffer para um arquivo aleatório no disco.
FILES ["⟨dispositivo⟩: ⟨nome do arquivo⟩"]	Comando	Fornece o diretório de um disco.
FOR ⟨variável⟩ = ⟨X⟩ TO ⟨Y⟩ [STEP Z] NEXT [⟨variável⟩]	Instrução	Permite que uma série de instruções sejam executadas dentro de um laço um determinado número de vezes.
FRE (⟨argumento⟩)	Função	Fornece o Nº de bytes livres ainda não utilizados.
GET [#] ⟨Nº do arquivo⟩ [,⟨Nº do dado⟩]	Instrução	Ler um dado de um arquivo aleatório gravado em um disco.
GO SUB ⟨Nº de linha⟩ RETURN [⟨Nº de linha⟩]	Instrução	Provoca o desvio para uma sub-rotina e depois retorna. Se o ⟨Nº de linha⟩ após o RETURN não estiver especificado, o retorno se produz para a primeira instrução posterior ao último GOSUB executado.
GOTO ⟨Nº de linha⟩	Instrução	Desvio da seqüência normal do programa para uma linha determinada.
HEX$ (⟨expressão⟩)	Função	Converte um número em um string hexadecimal.
IF ⟨exp.⟩ THEN ⟨instrução/Nº de linha⟩ [ELSE ⟨instrução/Nº de linha⟩]	Instrução	Tomada de decisão conforme o fluxo do programa, baseado no resultado fornecido por uma expressão.
INKEY$	Função	Fornece um string de 1 caractere introduzido através do teclado ou um string nulo quando não houver introdução.
INP (⟨número de "port"⟩)	Função	Fornece o byte lido no "port" indicado.
INPUT ["⟨string mensagem⟩";] ⟨lista de variáveis⟩	Instrução	Permite entradas através do teclado durante a execução do programa.
INPUT # ⟨Nº do arquivo⟩, ⟨lista de variáveis⟩	Instrução	Lê dados através do canal especificado e os atribui a variáveis do programa.
INPUT$ (⟨n⟩ [, [#]⟨Nº do arquivo⟩]	Função	Fornece um string de n caracteres lido através do teclado ou de um arquivo.
INSTR ([I], ⟨X$⟩, ⟨Z$⟩)	Função	Fornece a posição na qual se encontra o caractere Z$ dentro do string X$.
INT (⟨expressão⟩)	Função	Fornece o valor do maior inteiro menor ou igual à expressão.
INTERVAL ON/OFF/STOP	Instrução	Ativa, desativa ou suspende o controle sobre um intervalo de tempo transcorrido.
KEY ⟨Nº da tecla de função⟩, "⟨string⟩"	Comando	Define um string para uma das teclas de função.
KEY LIST	Comando	Lista o conteúdo de todas as teclas de função programáveis.
KEY (⟨Nº da tecla de função⟩) ON/OFF/STOP	Instrução	Ativa, desativa ou suspende a interrupção provocado por uma tecla de função.
KEY ON/OFF	Instrução	Ativa ou desativa a visualização das teclas de função na 24ª linha da tela.
KILL "⟨dispositivo⟩: ⟨nome arquivo⟩"	Instrução	Apaga um arquivo de um disco flexível.
LEFT$ (⟨string⟩, ⟨n⟩)	Função	Fornece um string formado pelos n caracteres da esquerda do "string".
LEN (⟨string⟩)	Função	Fornece o número de caracteres existentes no string.
[LET] ⟨variável⟩ = ⟨expressão⟩	Instrução	Atribui o valor de uma expressão a uma variável. LET pode ser omitido.
LINE [STEP] [(x_1, y_1)] – [STEP] [(x_2, y_2)] [, ⟨código de cor⟩] [, B [F]]	Instrução	Traça uma linha reta ligando os dois pares de coordenadas (x, y). B provocará o traçado de um retângulo e BF pintará o retângulo na cor definida.
LINE INPUT [("mensagem string");] ⟨variável string⟩	Instrução	Atribui uma linha inteira (de até 254 caracteres) introduzida pelo teclado a uma variável string.
LIST [⟨Nº de linha⟩ [–[⟨Nº de linha⟩]]]	Comando	Lista na tela todo o programa BASIC ou parte dele.
LLIST [⟨Nº de linha⟩ [–[⟨Nº de linha⟩]]]	Comando	Lista na impressora todo o programa BASIC ou parte dele.
LOAD "⟨dispositivo⟩: ⟨nome do arquivo⟩" [, R]	Comando	Carrega na memória um programa BASIC (arquivo ASCII) proveniente do dispositivo periférico.

Formato	Tipo	Descrição
LOC ((Nº do arquivo))	Função	Fornece a localização atual dentro de um arquivo.
LOCATE [(X)] [, (Y)] [, (Z)]	Instrução	Envia o cursor para uma posição dada na tela, indicada por (x, y). Z indica se o cursor deve ser visível ou não.
LOF ((Nº do arquivo))	Função	Fornece o comprimento ce um arquivo, em Nº de bytes.
LOG ((expressão))	Função	Fornece o logaritmo natural da expressão.
LPOS ((X))	Variável do sistema	Fornece a posição atual da cabeça da impressora dentro do buffer da impressora.
LPRINT [(lista de expressões)	Instrução	Imprime dados em uma impressora.
LPRINT USING (expressão string) ; (lista de expressões)	Instrução	Imprime dados em uma impressora usando um formato determinado.
LSET (X$) = (Y$)	Instrução	Preenche começando pela esquerda a variável X$ com o conteúdo da variável Y$.
MAXFILES = (expressão)	Instrução	Especifica o número máximo de arquivos abertos ao mesmo tempo.
MERGE "(dispositivo) : (nome do progr.) "	Comando	Anexa as linhas de um programa ASCII ao programa atualmente na memória.
MID$ ((X$), (I) [, (J)])	Função	Fornece uma expressão alfanumérica composta de caracteres, começando na posição I de X$.
MID$ ((string 1)), n [, m] = (string 2)	Instrução	Substitui uma parte de uma variável alfanumérica com uma outra ou com uma constante.
MKI$ ((valor))	Função	Converte um inteiro em um string de 2 caracteres.
MKS$ ((valor))	Função	Converte um valor de simples precisão em um string de 4 caracteres.
MKD$ ((valor))	Função	Converte um valor de dupla precisão em um string de 8 caracteres.
MOTOR [ON]/[OFF]	Instrução	Liga/desliga o gravador.
NAME "(dispositivo) : (nome arquivo 1) " AS "(nome arquivo 2)"	Instrução	Mudar o nome de um arquivo no disco.
NEW	Comando	Apagar da memória um programa BASIC completo.
OCT$ ((expressão))	Função	Fornece uma representação octal do número decimal dado como argumento.
ON ERROR GOTO (Nº de linha) RESUME	Instrução	Habilita uma rotina de tratamento de erros.
ON (expressão) GOTO (lista de Nºs de linha)	Instrução	Provoca um desvio para uma das diversas linhas especificadas, dependendo do valor fornecida pela expressão.
ON (expressão) GOSUB (lista de Nºs de linha)		
ON INTERVAL = (intervalo de tempo) GOSUB (Nº de linha)	Instrução	Define o Nº de linha a ser executado em cada ciclo de interrupção da máquina especificado por (intervalo de tempo).
ON KEY GOSUB (lista co Nº de linha)	Instrução	Indica o Nº de linha de início das sub-rotinas que devem ser seguidas caso ser pressionada uma das teclas de função.
ON SPRITE GOSUB (Nº de linha)	Instrução	Indica o endereço do desvio caso aconteça colisão entre sprites.
ON STOP GOSUB (Nº de linha)	Instrução	Indica o endereço do desvio caso as teclas CTRL + STOP sejam pressionadas simultaneamente.
ON STRIG GOSUB (lista co Nº de linhas)	Instrução	Define o Nº de linha de início das sub rotinas que devem ser seguidas ao serem pressionados os botões disparadores dos joysticks.
OPEN "(dispositivo) : [(nome do arquivo)] " [FOR (modo)] AS [#] (número do arquivo)	Instrução	Fixa um buffer para I/O e define o modo que será utilizado com o buffer.
OUT (Nº de "port"), (byte)	Instrução	Manda um byte para um "port" de saída da máquina.
PAD ((n))	Função	Fornece o estado do "touch pad" (acessório).
PAINT [STEP] (x, y) [, (cor)] [, (cor da linha de contorno)]	Instrução	Preenche uma figura gráfica com uma cor determinada.
PDL ((n))	Função	Fornece o estado de um dos "paddles" (acessório).
PEEK (I)	Função	Lê o byte contido na posição de memória (I).
PLAY (expr. string 1) [, (expr. string 2)[, (expr. string 3)]]	Instrução	Produz música de acordo com a linguagem macro de música.
PLAY ((canal de voz))	Função	Fornece informação sobre o estado dos canais de voz.
POINT (x, y)	Função	Lê a cor de um pixel nos modos gráficos.
POKE (endereço de memória), (dado)	Instrução	Escreve um byte dado em uma determinada posição da memória.
POS (I)	Variável do sistema	Fornece a posição atual do cursor.
PRESET [STEP] (x, y) [, (código de cor)]	Instrução	Desenha um ponto em um lugar determinado da tela, na cor especificada.
PRINT [(lista de expressões)]	Instrução	Impressão de dados na tela.
PRINT USING "(string)"; (lista de expressões)	Instrução	Impressão de strings ou dados numéricos seguindo um formato determinado.
PRINT # (Nº do arquivo), [USING (formato de impressão)] ; (lista de expressões)	Instrução	Escrever dados através de um determinado canal em um circuito seqüencial.
PSET [STEP] (x, y) [, (código de cor)]	Instrução	Atribui a um ponto da tela uma cor determinada.

Formato	Tipo	Descrição
PUT [#] (Nº de arquivo) [, (Nº do dado)]	Instrução	Escreve um dado do buffer em um arquivo aleatório em disco flexível.
PUT SPRITE (plano do sprite) [, [STEP] (X,Y)] [, (cor)] [, (Nº do sprite)]	Instrução	Insere um determinado sprite na tela.
READ (lista de variáveis)	Instrução	Lê uma constante de uma instrução "DATA" e atribui a uma variável.
REM [(observação)]	Instrução	Permite inserir comentários dentro da listagem do programa.
RENUM [[(novo número)] [, [(velho número)] [, (incremento)]]]	Comando	Permite a renumeração de linhas do programa.
RESTORE [(número de linha)]	Instrução	Permite que a instrução DATA seja relida a partir de uma linha específica.
RIGHT$ ((X$), (n))	Função	Fornece um string composto pelos n caracteres da direita de X$.
RND ((X))	Função	Fornece um número aleatório compreendido entre Ø e 1.
RSET (X$) = (Y$)	Instrução	Preenche, começando pela direita, a variável X$, com o conteúdo da variável Y$.
RUN [(dispositivo) : (nome do programa)] [(X)]	Comando	Roda o programa existente na memória ou carregado de um dispositivo periférico.
SAVE "(dispositivo) : [(nome do arquivo)]"	Comando	Transfere um programa BASIC carregado na memória para um periférico.
SCREEN [(modo)] [, (dimensão dos sprites)] [, (som)] [, (velocidade do gravador)] [, (tipo de impressora)]	Instrução	Define o modo de trabalho na tela, tamanho dos sprites, estalidos das teclas, velocidade de transmissão ao gravador e tipo da impressora usada.
SGN (X)	Função	Assume o valor 1 quando X > Ø e –1 quando X < Ø.
SIN (X)	Função	Fornece o seno de X em radianos.
SOUND (registro do GSP), (dado)	Instrução	Insere um valor em um dos registros do gerador de som programável.
SPACE$ ((X))	Função	Fornece um string de X espaços.
SPC ((I))	Função	Imprime I espaços na tela.
SPRITE ON/OFF/STOP	Instrução	Ativa, desabilita ou suspende o controle do BASIC sobre a colisão entre sprites.
SPRITES ((Nº do sprite))	Variável do sistema	Define o desenho do sprite.
SQR ((X))	Função	Fornece o valor da raiz quadrada de X.
STICK ((X))	Função	Fornece a posição de um joystick ou o estado das teclas de controle do cursor. Valores da função 
STOP	Instrução	Interrompe a execução do programa e volta ao nível de comando.
STOP ON/OFF/STOP	Instrução	Ativa, desativa ou suspende o controle sobre a situação em que as teclas CTRL e STOP são pressionadas simultaneamente.
STRIG ((n))	Função	Fornece o estado do botão disparador do joystick ou da barra espaçadora.
STRIG ON/OFF/STOP	Instrução	Ativa, desativa ou suspende o controle do BASIC sobre o estado dos botões disparadores do joystick, ou da barra espaçadora.
STR$ ((X))	Função	Fornece uma representação alfanumérica da expressão numérica (X).
STRING$ ((I), [(J) / (X$)])	Função	Fornece um valor alfanumérico de comprimento I, contendo somente caracteres cujo código ASCII é J ou então o primeiro caractere do string X$.
SWAP (variável), (variável)	Instrução	Intercambiar os conteúdos de duas variáveis.
TAB ((I))	Função	Deslocar o cursor sobre a mesma linha até a posição I.
TAN ((X))	Função	Fornece o valor da tangente de X em radianos.
TIME/TIME = Ø	Variável do sistema	Inteiro gerado pelo timer interno do sistema.
TRON/TROFF	Comando	Ativa ou desativa a função de análise passo passo da execução do programa.
USR [(dígito)] (X)	Função	Chama uma sub-rotina do usuário em linguagem de máquina.
VAL ((string))	Função	Converte a representação string de um número em seu valor numérico.
VARPTR ((variável)) VARPTR (# (número do arquivo))	Função	Fornece o endereço do primeiro byte de dados identificado por (variável) ou o primeiro byte do bloco de controle de um arquivo.
VDP ((n))	Variável do sistema	Contém o conteúdo dos registros do VDP.
VPEEK ((endereco VRAM))	Função	Fornece o conteúdo de um byte da memória de vídeo.
VPOKE (endereço VRAM), (byte)	Instrução	Insere um valor determinado em um byte da memória de vídeo.
WAIT (port), (expressão Y) [, (expressão V)]	Instrução	Suspende a execução de um programa de leitura através de um port até que [(bit de entrada) XOR (Y) AND (V)] forneça um valor diferente de zero.
WIDTH (X)	Instrução	Indica o Nº de posições disponíveis em uma linha da tela.

PRODUZIDO PELA EPCOM
EQUIPAMENTOS ELETRÔNICOS DA AMAZÔNIA LTDA.
AV. BURITI, 3650 - BLOCO A
DISTRITO INDUSTRIAL - MANAUS/AM
C.G.C. 04.155.123/0001-52 - INDÚSTRIA BRASILEIRA