BOMANNS

GW BASIC

Guia de Referência Básica

Quick Reference

GW BASIC

Guia de Referência Básica

McGraw-Hill

H. F. BOMANNS

GW – BASIC

GW-BASIC

Guia de Referência Básica

H. F. Bomanns

Tradução
Felipe Armando Padolano

Revisão Técnica
Carlos Alberto Botelho
Professor e Consultor de Matemática

MAKRON *Books* do Brasil Editora Ltda.
Editora McGraw-Hill Ltda.
São Paulo
Rua Tabapuã, 1105, Itaim-Bibi
CEP 04533
(011) 829-8604 e (011) 820-8528

Rio de Janeiro • Lisboa • Porto • Bogotá • Buenos Aires • Guatemala • Madrid • México • New York • Panamá • San Juan • Santiago

Auckland • Hamburg • Kuala Lumpur • London • Milan • Montreal • New Delhi • Paris • Singapore • Sydney • Tokyo • Toronto

Do original
GW – Basic – A Quick Program Reference Guide
For IBM/PC/XT/AT and compatibles

EDITOR : MILTON MIRA DE ASSUMPÇÃO FILHO

Coordenadora de Revisão: Daisy Pereira Daniel
Supervisor de Produção: José Rodrigues
Capa:layout: Douglas Lucas

Editoração Eletrônica: JAG Composição Editorial e Artes Gráficas Ltda.

Dados de Catalogação na Publicação (CIP) Internacional (Câmara Brasileira de Livro, SP, Brasil)

Bomanns, H.F.
GW – BASIC : guia de referência básica / H.F. Bomanns ; tradução Felipe Armando Padolano ; revisão técnica Carlos Alberto Botelho. — São Paulo : McGraw-Hill, 1989.

1. GW – BASIC (Linguagem de programação para computadores) I. Título.

89-0843 CDD-001.6424

Índices para catálogo sistemático:

1.GW – BASIC : Linguagem de programação : Computadores : Processamento de dados 001.6424

SUMÁRIO

INTRODUÇÃO

GW – Basic – Guia de Referência Básica não é um texto introdutório para o GW – BASIC e sim um guia rápido de referência para o programador de nível intermediário até o avançado. Nós contamos, portanto, que você tenha um conhecimento geral e alguma experiência de trabalho no que se refere ao BASIC e ao MS-DOS.

SOBRE GW/PC-BASIC

PC BASIC é o nome da linguagem de programação BASIC para o IBM-PC, PC-XT e PC-AT. GW-BASIC é a versão de BASIC para PC-compatíveis. Ambos foram desenvolvidos pela Microsoft e, para todos os propósitos, os dois são idênticos. Tomaremos a liberdade de nos referirmos aos programas através do nome genérico de GW-BASIC.

GW-BASIC tem mais de 230 comandos, sentenças e funções. Isto o torna uma ferramenta de programação bastante completa. Mas o grande número de comandos dificulta o programador BASIC de lembrar sintaxe, parâmetros e uso de cada comando. Este guia fornece acesso conveniente a tais informações do GW-BASIC.

Apesar de o considerarmos uma versão genérica da linguagem de programação BASIC para o IBM-PC, PC-XT, PC-AT e compatíveis, GW-BASIC tem diferenças sutis dependendo de um

determinado computador para o qual ele estiver configurado. Descrevemos muitas dessas diferenças neste guia, mas refira-se ao seu manual original de BASIC caso você encontre alguma grande diferença.

COMO USAR ESTE GUIA

Este guia possui três seções principais: informações introdutórias e gerais, a seção dos Comandos, Funções e Sentenças, e os três índices.

A seção dos Comandos, Funções e Sentenças agrupa cada comando, função e sentença (conhecidas como palavras-chaves) de acordo com o seu uso, e então lista as palavras-chaves em ordem alfabética dentro desses grupos de uso.

Você encontrará *uma letra entre parênteses* ao final de cada título de palavra-chave. Essa letra lhe dirá se o item corrente é um comando (C), função (F), sentença (S) ou operador (O).

Certos caracteres e notações usados na sintaxe e explicações sobre as palavras-chaves têm o seguinte significado:

[ ] Parâmetros entre colchetes são opcionais e não são necessários à execução do comando. Eles geralmente controlam funções especiais dos comandos.

abc Parâmetros são indicados em letras minúsculas inclinadas (itálico). Isso permite que você distinga facilmente palavras-chaves e seus parâmetros.

| Parâmetros alternativos são separados por uma barra vertical. Somente *um* dos parâmetros de uma lista separados por esse caractere é que pode ser usado.

nome-de-arquivo

Nomes de arquivos podem ser especificados como uma *máscara*. Ela segue as convenções do MS-DOS para drive, caminho, nome e extensão. Por exemplo:

```
a:\dos\*.com
```

Valores numéricos são especificados como números decimais. Você também pode especificar um valor numérico como um número hexadecimal precedendo-o com &H, ou como um número octal precedendo-o com &O.

Se um parâmetro não for especificado, GW-BASIC considera que o valor usado mais recentemente é o valor padrão. Você deve inserir uma vírgula no lugar de qualquer parâmetro *requerido* para que esse valor padrão seja usado.

USANDO OS ÍNDICES

Se você souber o nome de uma palavra-chave, mas estiver inseguro quanto ao seu uso ou à sintaxe, consulte o "Índice de Comandos" no final do livro. Cada comando do GW-BASIC está listado em ordem alfabética, e lhe indica o(s) número(s) da(s) página(s) onde aquela palavra-chave pode ser encontrada no guia.

Quando estiver procurando por uma palavra-chave para executar uma tarefa específica, mas estiver incerto quanto ao seu nome ou sintaxe, consulte o "Índice Remissivo". Lá você encontrará uma lista de palavras-chaves agrupadas de acordo com o seu uso, juntamente com uma breve descrição dos seus efeitos, e referências a páginas relevantes.

Para informações e consultas sobre tópicos gerais do GW-BASIC e operações específicas, consulte o último Índice do guia.

EXEMPLO DE PÁGINA DE COMANDO DO GW-BASIC

Eis aqui uma rápida explicação do formato que este guia usa para descrever comandos, sentenças e funções do GW-BASIC, e seus parâmetros:

Palavra-Chave	**Descrição Resumida**	**Tipo**
CHAIN	**encadeia "overlays"(S)**	

Sintaxe com Parâmetros

=*CHAIN[MERGE] nome_de_arquivo*
[*primeira_linha*] [,ALL]
[,DELETE *de até*]

Descrição

Encadeia "overlays" ou partes adicionais de um programa

Descrição dos Parâmetros

MERGE	MERGE foi apresentado antes. Ele tem o mesmo efeito quando usado com CHAIN: o programa carregado com MERGE é combinado com o programa na memória de forma que as linhas existentes são substituídas ou, então, linhas são inseridas, dependendo do número de linha. A execução começa na *primeira_linha*.

nome_de_arquivo	Um nome de arquivo, de acordo com as convenções do MS-DOS, consistindo em drive, caminho e nome de arquivo.
primeira_linha.	A linha na qual a execução terá início após o CHAIN.
,ALL	Esse parâmetro diz ao GW-BASIC para preservar todos os valores de variáveis existentes. Sem ALL, CHAIN apaga todas as variáveis e libera a memória. Se apenas algumas variáveis forem passadas, use COMMON.
DELETE *de até*	DELETE somente pode ser usado junto com MERGE. DELETE apaga o conjunto de linhas especificadas pela faixa *de até* antes que o programa seja carregado com MERGE.

O programa a ser encadeado deve ser gravado no formato ASCII. Isto é feito usando-se o comando SAVE *nome_de_arquivo*,A.

Comentários Adicionais

INFORMAÇÕES GERAIS

TIPOS DE CONSTANTES E VARIÁVEIS

GW-BASIC distingue entre vários tipos de dados e de variáveis. Um nome de variável pode ter até 40 caracteres, consistindo em letras (A a Z), dígitos (0 a 9) e pontos (como separadores). O nome deve começar com uma letra e terminar com um caractere que indique o tipo de variável. Um nome de variável não pode ser uma palavra reservada do GW-BASIC. Da mesma forma, uma palavra reservada não deve estar contida dentro de um nome de variável, em nome da legibilidade.

TIPOS DE DADOS

Números são usados em funções matemáticas e cálculos. GW-BASIC reconhece três tipos de variáveis numéricas:

Variáveis Inteiras armazenam números inteiros de -32768 até +32767. Variáveis deste tipo são indicadas por um sinal de percentagem (%) logo após o nome da variável.

Variáveis Reais de Precisão Simples armazenam valores em ponto flutuante de -1.701415E-38 até +1.701415E+38. Variáveis

deste tipo são indicadas por um ponto de exclamação (!) logo após o nome da variável.

Variáveis Reais de Precisão Dupla armazenam valores em ponto flutuante de -1.701411834604692D-38 até +1.701411834 604692D+38. Variáveis deste tipo são indicadas por um sinal de número (#) logo após o nome da variável.

Se valores de tipos diferentes forem usados em um cálculo ou função, os resultados serão sempre do mesmo tipo que o valor mais simples. Se nenhum dos identificadores dados acima for especificado, a variável terá o tipo de dados do valor atribuído a ela.

Variáveis string (seqüência de caracteres) são os únicos tipos de dados usados pelo GW-BASIC para processar caracteres. Uma variável deste tipo pode armazenar qualquer seqüência de 1 a 255 caracteres. Essas variáveis são indicadas por um sinal de dólar ($) logo após o nome da variável.

TIPOS DE VARIÁVEIS

Variáveis que podem armazenar somente um valor por vez são denominadas *variáveis standard*. Isso é para diferenciá-las das *matrizes*, que podem armazenar mais de um valor de uma só vez.

As variáveis standard se parecem com esta forma:

```
10 ENDERECO%=12:SOBRE.NOME$="Pacheco"
```

As matrizes diferem nas suas dimensões. GW-BASIC permite matrizes simples e multidimensionais. Estas são matrizes simples:

```
10 MESES$(3)="Março":ARR!(15)=12.56
```

Você pode imaginar estas variáveis como gavetas de um armário. Cada gaveta pode ser identificada e "aberta" pelo seu número.

Estas são matrizes bidimensionais:

```
10 SOBRE.NOME$(3,1) = "Pacheco"

20 ARR!(10,2) = 78.45
```

As matrizes multidimensionais geralmente são as mais usadas. Você pode imaginar matrizes multidimensionais como gavetas em diversos armários colocados lado a lado, com as gavetas organizadas em linhas e colunas. Cada gaveta pode então ser identificada e localizada pelo seu número de linha e de coluna. Isso é conhecido como *indexação*.

GW-BASIC também permite matrizes com mais de duas dimensões, o que acrescenta profundidade (mais dimensões) aos exemplos anteriores. O número de dimensões permitidas para as matrizes depende da quantidade de memória disponível.

TECLAS DE EDIÇÃO

As seguintes teclas estão disponíveis para inserir e editar listagens de programas. Dependendo da sua versão do GW-BASIC, estas combinações de teclas podem produzir resultados diferentes dos listados abaixo, ou podem até não ter efeito nenhum.

Efeito	Tecla
Aceita a linha de programa entrada ou editada e coloca-a na memória.	Enter ou Ctrl+M
Toda a entrada e saída é direcionada para a impressora. Pressionando estas teclas novamente desliga este modo.	Ctrl+P ou Ctrl+PrtSc
Limpa a tela e coloca o cursor no canto superior esquerdo da tela.	Ctrl+Home ou Ctrl+L
Limpa o restante da linha de programa a partir da posição do cursor.	Ctrl+End ou Ctrl+E
Limpa a linha e coloca o cursor no início da linha.	Esc
Limpa o restante da tela abaixo da linha atual onde está o cursor.	Ctrl+Z ou Ctrl+PgDn
Apaga o caractere que está sob o cursor.	Del
Apaga o caractere à esquerda do cursor.	BackSpace ou Ctrl+H
Apaga o conteúdo da linha de programa a partir da posição do cursor até os próximos dois	PgDn

pontos (:) ou até o próximo espaço. Insere uma linha em branco dentro da linha de programa.	Ctrl+Enter ou Ctrl+J
Move o cursor até a próxima posição de tabulação para a direita.	Tab ou Ctrl+I
Coloca o cursor no final da linha de programa.	End ou Ctrl+N
Coloca o cursor no canto superior esquerdo da tela sem limpar a tela.	Home ou Ctrl+K
Coloca o cursor no início da palavra à sua direita.	Ctrl+→
Coloca o cursor no início da palavra à sua esquerda.	Ctrl+← ou Ctrl+B
Pára a saída de uma listagem na tela. A listagem é retomada quando outra tecla for pressionada.	Ctrl+ NumLock
Finaliza o modo de edição, fazendo quaisquer mudanças serem perdidas. No modo de programa, o programa é finalizado e o GW-BASIC retorna ao modo direto.	Ctrl+C ou Ctrl+Break
O conteúdo atual da tela é enviado à impressora. Normalmente, apenas cópia da tela é que pode ser impressa. O utili-	Shift+ PrtSc

tário GRAPHICS.COM do MS-DOS deve ser carregado para que se possam imprimir gráficos.	
A listagem é rolada para cima. O cursor deve estar em uma linha que possua um número de linha válido.	Ctrl+Y ou Ctrl+↓
A listagem é rolada para baixo na tela. O cursor deve estar em uma linha que possua um número de linha válido.	Ctrl+X ou Ctrl+↑
Alterna entre o modo de inserção e o de sobreposição.	Ins ou Ctrl+R
Desliga e liga a exibição das teclas de função na 25a. linha.	Ctrl+T

Além das teclas de edição, GW-BASIC possui um conjunto de comandos de edição para entrada de um programa e para edição de uma listagem (veja a seção "Edição", págs. 15-27).

Valores para parâmetros, variáveis e constantes podem ser digitados em notação decimal sem um prefixo; em notação hexadecimal com o prefixo &H; e em notação octal com o prefixo &O. Para alguns comandos, o ponto (.) é usado para representar a última linha processada. A última linha processada é a última linha que foi modificada ou a última linha na qual ocorreu um erro.

GW-BASIC normalmente indica erros de entrada ou erros durante a execução de um programa através de uma parada na execução do comando. Ele então mostra uma mensagem de erro que lista o conteúdo da linha na qual ocorreu o erro.

ABREVIAÇÕES COM A TECLA ALT

Os comandos mais usados podem ser digitados no GW-BASIC de um modo abreviado a partir da seguinte combinação de teclas:

Alt+ outra tecla

Mantenha apertada a tecla Alt e então pressione a tecla apropriada. Quando você soltar a tecla, a palavra-chave correspondente aparecerá na tela:

Tecla	Palavra-Chave	Tecla	Palavra-Chave
A	AUTO	N	NEXT
B	BSAVE	O	OPEN
C	COLOR	P	PRINT
D	DELETE	Q	não usada
E	ELSE	R	RUN
F	FOR	S	SCREEN
G	GOTO	T	THEN
H	HEX$	U	USING
I	INPUT	V	VAL
J	não usada	W	WINDOW
K	KEY	X	XOR
L	LOCATE	Y	não usada
M	MERGE	Z	não usada

Observação: Esta lista não é válida para todas as versões do GW-BASIC. A versão que você está usando pode ter algumas diferenças. Ao contrário das teclas de função, tais atribuições *não podem* ser modificadas.

INICIANDO O GW-BASIC

BASIC	
BASICA	executam o GW-BASIC (C)
GW-BASIC	

BASIC: BASICA: GWBASIC [*prog*] [<*entpad*] [>*saípad*] [/C:*buffer*] [/D] [/F:*arquivos*] [/M:[*endereço*] [,*blocos*]] [/S:*buffer*] [I]

Inicia a execução do interpretador BASIC.

prog — Especifica o nome de um programa BASIC que será executado depois que o interpretador BASIC for carregado. Se omitido, BASIC exibe a mensagem READY e vai para o modo de comando.

< *entpad* — Especifica o nome do arquivo ou dispositivo a partir do qual o GW-BASIC lê os dados de entrada.

> *saípad* — Especifica o nome do arquivo ou dispositivo para o qual o GW-BASIC envia a saída.

*/C:buffer** Especifica o tamanho do buffer de transferência de dados quando estiver sendo usada a porta de comunicação assíncrona. O tamanho padrão é de 256 bytes. O máximo é 32767.

/D Especifica que as funções matemáticas ATN, COS, EXP, LOG, SIN, SQR, TAN são executadas usando-se aritmética de precisão dupla.

/F:arquivos Especifica o número máximo de arquivos que podem ser acessados ao mesmo tempo. O valor padrão é 3. O máximo é 15.

/M:endereço,blocos

GW-BASIC normalmente possui 64K de memória disponível para os segmentos de pilha e de dados. Se a sua aplicação necessitar de uma parte dessa memória (por exemplo, rotinas em linguagem de máquina), você pode proteger uma parte desse espaço. Especifique *endereço* para indicar a posição de memória mais alta para o GW-BASIC.

Para reservar memória acima do GW-BASIC, os blocos devem ser múltiplos em bytes de 16 (segmentos).

Por exemplo, /M:&H1010 reserva 64K (&H1000 (=4096*16) para o GW-BASIC, mais 256 bytes (&H10(=16*16) para rotinas em linguagem de máquina.

/S:buffer Especifica o tamanho máximo de registro permitido para um arquivo de acesso randômico. O valor padrão é 128. O valor máximo é 32767.

/I Aloca dinamicamente o espaço de memória necessário para operações com arquivos.

* Nota do Tradutor: Buffer é uma memória temporária utilizada para receber ou enviar dados aos dispositivos.

COMANDOS, FUNÇÕES E SENTENÇAS

CONSTANTES E VARIÁVEIS

DEFXXX	definição de tipo global (S)
DEFINT	*de-para*
DEFSNG	*de-para*
DEFDBL	*de-para*
DEFSTR	*de-para*

Uma definição de tipo de dados globais permite que seja atribuído um determinado tipo de dados a diversas variáveis. Estas variáveis são agrupadas pela primeira letra dos seus nomes.

DEFINT	Define o tipo de dados como inteiro.
DEFSNG	Define o tipo de dados como real de precisão simples.
DEFDBL	Define o tipo de dados como real de precisão dupla.
DEFSTR	Define o tipo de dados como string.

de Especifica a primeira letra da faixa à qual se aplica a definição global.

para Especifica a última letra da faixa à qual se aplica a definição global.

DIM **dimensiona matrizes (S)**

DIM *variável* (índices) [,*variável (índices)*]

Reserva memória para matrizes de uma dimensão (unidimensionais) ou para matrizes multidimensionais com índices maiores que 10.

variável O nome da matriz a ser dimensionada. Para matrizes numéricas, um identificador de tipo (%, !, ou #) pode ser acrescentado ao nome da variável. Para matrizes strings, o identificador $ deve vir logo após o nome da variável (assumindo que o nome da variável não seja definido globalmente como uma string por DEFSTR).

índices Uma ou mais expressões numéricas, separadas por vírgulas, que especificam o número de elementos por dimensão da matriz. O número máximo de elementos por dimensão é 255. O número de elementos totais permitido é limitado pela memória disponível.

ERASE **apaga o dimensionamento (S)**

ERASE*variável [,variável...]*

Apaga uma ou mais matrizes previamente dimensionadas.

variável Especifica o(s) nome(s) da(s) matriz(es) da(s) qual(is) será eliminado o dimensionamento.

LET **inicializa variáveis (S)**

LET *variável = valor*

Atribui um valor a uma variável.

variável

Qualquer tipo de variável, incluindo um elemento de uma matriz.

valor

O valor atribuído à variável. Este deve ser do mesmo tipo da variável.

LET pode ser omitido de atribuições de valores no GW-BASIC.

OPTION BASE — inicializa matrizes (S)

OPTION BASE 0 | 1

Faz o limite mais baixo de todos os índices de matrizes começar em 0 ou 1.

0 Todas as matrizes do programa têm índices iniciais de 0.

1 Todas as matrizes do programa têm índices iniciais de 1.

READ..DATA — tabelas internas de dados (S)

READ *variável [,variável...]*

DATA *texto* | *valor*

Define e lê valores fixos de dados dentro de um programa.

variável

Especifica uma variável de qualquer tipo. Os dados devem corresponder ao tipo de dados da variável. Diversas variáveis podem vir logo após uma sentença READ.

texto | valor

Especifica os dados atribuídos às variáveis. Os dados que incluem vírgulas devem corresponder aos tipos de dados das variáveis. Dados individuais são separados por vírgulas. Strings podem ser colocadas entre aspas (" "). Valores numéricos são separados somente por vírgulas. Os tipos de dados podem ser misturados dentro de uma linha DATA.

RESTORE — posicionar o ponteiro DATA (S)

RESTORE *número_de_linha*

Posiciona o ponteiro DATA numa determinada linha de dados.

número_de_linha

Especifica o número de linha que contém o primeiro item de DATA a ser lido. Se não for especificado nenhum número de linha, o ponteiro é posicionado para a primeira linha DATA no programa.

SWAP — troca conteúdos de variáveis (S)

SWAP *variável_1, variável_2*

Troca os conteúdos de duas variáveis do mesmo tipo de dados.

variável_1, variável_2

Ambas as variáveis devem ser do mesmo tipo de dados. Elementos de matrizes podem ser trocados.

COMPARAÇÃO DE DADOS

= **compara dados (O)**

IF *expressão_1* = *expressão_2* THEN *sentença*

Testa a igualdade entre duas expressões.

expressão_1, expressão_2

Expressões a serem comparadas. Estas podem ser constantes, variáveis, ou os resultados de cálculos ou funções. As expressões são do mesmo tipo de dados.

O operador relacional = pode ser combinado com os operadores relacionais <e > a fim de testar se *expressão_1* é menor que, maior que, ou igual à *expressão_2*.

<> **compara dados (O)**

IF *expressão_1* < > *expressão_2* THEN *sentença*

Testa a desigualdade entre duas expressões.

expressão_1, expressão_2

Expressões a serem comparadas. Estas podem ser constantes, variáveis, ou os resultados de cálculos ou funções. As expressões são do mesmo tipo de dados.

O operador relacional pode ser combinado com o operador relacional = para testar se *expressão_1* é menor que, igual a, ou maior que *expressão_2*.

< compara dados (O)

IF *expressão_1* < *expressão_2* THEN *sentença*

Testa se a primeira expressão possui um valor menor que a segunda.

expressão_1, expressão_2

Expressões a serem comparadas. Estas podem ser constantes, variáveis, ou os resultados de cálculos ou funções. As expressões devem ser do mesmo tipo de dados.

O operador relacional pode ser combinado com o operador relacional = para testar se *expressão_1* é menor que ou igual a *expressão_2*.

> compara dados (O)

IF *expressão_1* > *expressão_2* THEN *sentença*

Testa se a primeira expressão possui um valor maior que a segunda.

expressão_1, expressão_2

Expressões a serem comparadas. Estas podem ser constantes, variáveis, ou os resultados de cálculos ou funções. As expressões são do mesmo tipo de dados.

O operador relacional > pode ser combinado com o operador relacional = para testar se *expressão_1* é maior que ou igual a *expressão_2*.

AND — combina comparações (O)

IF *condição_1* AND *condição_2* THEN *sentença*

Combina comparações múltiplas. A sentença é executada se ambas, *condição_1* e *condição_2*, forem verdadeiras.

condição_1, condição_2

Comparações usando os operadores relacionais =, <>, <=, =>, <, ou >.

NOT — inverte o resultado de uma comparação (O)

IF NOT *condição* THEN *sentença*

Inverte o resultado de uma comparação.

condição

Comparação usando os operadores =, <>, <=, =>, <, ou >.

OR — combina comparações (O)

IF *condição_1* OR *condição_2* THEN *sentença*

Combina múltiplas comparações. A sentença é executada se ou *condição_1* ou *condição_2* for verdadeira.

condição_1, condição_2

Comparações usando os operadores relacionais =, <>, <=, =>, <, ou >.

CONVERSÃO DE DADOS

ASC — caractere ASCII decimal (F)

ASC (*expressão*)

Retorna o valor decimal de um caractere ASCII.

expressão

expressão pode ser uma constante string, variável, ou o resultado de uma função string. Somente o primeiro caractere da expressão é convertido, independente de quantos caracteres a seqüência possuir.

CDBL — →precisão dupla (F)

CDBL (*expressão*)

Converte uma expressão numérica para um valor de precisão dupla.

expressão

expressão pode ser uma constante, o resultado de um cálculo, ou o resultado de uma função numérica.

CHR$ — decimal → caractere ASCII (F)

CHR$ (*número*)

Converte um valor numérico para um caractere ASCII.

número

Este pode ser uma constante, o resultado de um cálculo, uma variável numérica, ou o resultado de uma função numérica. É um valor inteiro na faixa de 0 a 255.

CINT → **valor inteiro (F)**

CINT (*expressão*)

Converte uma expressão numérica para um valor inteiro.

expressão

Especifica uma constante, o resultado de um cálculo, uma variável numérica, ou o resultado de uma função numérica. O resultado está na faixa de -32768 a +32767.

CSNG → **precisão simples (F)**

CSNG (*expressão*)

Converte uma expressão numérica para um valor de precisão simples.

expressão

Esta pode ser uma constante, o resultado de um cálculo, uma variável numérica, ou o resultado de uma função numérica.

DEF FN **funções matemáticas definidas pelo usuário (S)**

DEF FN *nome* (*parâmetro,...*) = *função*

FN *nome* (*parâmetro,...*)

Define/chama uma função matemática definida pelo usuário.

nome

Especifica o nome da função que é usado quando chamada com FN. Este pode ser um nome válido para uma variável numérica.

parâmetro

Especifica uma ou mais variáveis numéricas, que são definidas localmente, significando que elas são válidas somente para esta função. O número de variáveis na definição deve ser igual ao número de parâmetros informados quando da chamada.

função

Determina como as variáveis são processadas. Todas as funções matemáticas aceitas pelo GW-BASIC podem ser usadas.

A função não pode chamar a si mesma. Ela deve ser definida com DEF FN antes de ser chamada. A definição da função não pode exceder uma linha de programa em comprimento. Não são permitidas definições no modo direto.

HEX$ — decimal → hexadecimal (F)

HEX$(*número*)

Converte um valor decimal para uma seqüência de caracteres hexadecimais.

número

Este pode ser uma constante, o resultado de um cálculo, uma variável numérica, ou o resultado de uma função numérica. *número* é representável por um valor de 16bits na faixa de -32768 a +65535.

OCT$	**decimal → octal (F)**

OCT$(*número*)

Converte um valor decimal para uma seqüência de dígitos octais.

número

Este pode ser uma constante, o resultado de um cálculo, uma variável numérica, ou o resultado de uma função numérica. *número* é representável por um valor de 16 bits.

RANDOMIZE	**realimenta o gerador de números randômicos (S)**
RND	**gera um número randômico (F)**

RANDOMIZE [*número*]

RND [*número*]

Indica a base numérica para números randômicos e gera um número randômico.

número

Especifica o limite do número randômico a ser gerado. Se RND for usado sem nenhum parâmetro, GW-BASIC usa a

mesma seqüência para gerar números randômicos. *número* pode ser um valor entre -32768 e +32767.

Se você chamar RANDOMIZE sem *número*, terá de digitar um número para o gerador de números randômicos após a pergunta:

RANDOM NUMBER SEED (-32768 TO 32767)?

O valor inicial e a seqüência serão determinados pela sua entrada de dados.

STR$ **número → strings (F)**

STR$ (*expressão*)

Converte um valor numérico para uma string.

expressão

Pode ser uma constante, o resultado de um cálculo, uma variável numérica, ou o resultado de uma função numérica.

VAL **string → número (F)**

VAL (*STRING*)

Converte uma string contendo apenas algarismos para um valor numérico.

STRINGS

Pode ser uma constante, variável, ou o resultado de uma operação com strings.

O conteúdo da string é representado como um valor numérico. Se VAL encontrar um caractere dentro da seqüência que não pode ser representado como um valor, a conversão será finalizada sem qualquer mensagem de erro, e serão retornados os números convertidos até aquele ponto.

EDIÇÃO

AUTO — numeração automática de linhas (C)

AUTO [*linha_inicial*[.] [,*incremento*]

Exibe o número de linha seguinte conforme novas linhas de programa forem digitadas. AUTO será desativado quando você pressionar Ctrl+Break.

linha_inicial

Especifica o primeiro número de linha mostrado. Para iniciar com número de linha 10, digite AUTO 10. O valor padrão é 0.

O número de linha atual pode ser abreviado como um ponto.

incremento

Especifica o intervalo entre números de linha sucessivos. O valor padrão é 10.

Um asterisco será mostrado próximo a um número de linha caso aquele número de linha já exista no programa. Pressione Enter para reter a antiga linha de programa. Caso contrário, digite a linha que substituirá a linha antiga.

CLEAR — inicializa o armazenamento de variáveis (C)

CLEAR [,*espaço_de_dados*] [,*espaço_da_pilha*]

Inicializa variáveis numéricas e strings e anula as dimensões de variáveis de matrizes. Variáveis numéricas são inicializadas com zero, variáveis de strings com seqüências nulas e variáveis de matrizes são inicializadas não-definidas.

espaço_de_dados

Especifica o número máximo de bytes reservados para as variáveis de dados. O valor padrão é 65535. Decrementando este valor, você pode reservar uma área de memória acima da área de variáveis para rotinas em linguagem de máquina.

espaço_da_pilha

Especifica o número de bytes reservados para a memória na pilha. O valor padrão é 512 bytes.

CONT — continua a execução do programa (C)

CONT

Recomeça a execução do programa que foi interrompido por uma sentença STOP ou END, ou quando você pressionou Ctrl+Break. O programa é recomeçado a partir do ponto de interrupção.

Um programa não pode ser recomeçado se for mudado logo após a interrupção.

DELETE — apaga linhas do programa (C)

DELETE [*da_linha* | .] [-[*até_linha* |]

Apaga uma única linha ou uma faixa de linhas dentro do programa.

da_linha

Especifica a primeira linha da faixa a ser apagada.

até_linha

Especifica a última linha da faixa a ser apagada.

Separa *da_linha* e *até_linha* quando ambos os parâmetros forem especificados. Se *até_linha* for omitido,todas as linhas desde *da_linha* até o fim do programa serão apagadas. Se *da_linha* for omitido, todas as linhas desde o início do programa até *até_linha* serão apagadas.

O número de linha atual pode ser abreviado com um ponto.

EDIT **edita linhas (C)**

EDIT [*número_de_linha*|.]

Exibe uma linha de programa para mudanças ou correções. Todos os comandos de edição de tela podem ser usados para editar a linha de programa.

número_de_linha

Especifica a linha de programa sendo mudada ou corrigida.

A linha de programa atual pode ser abreviada com um ponto.

FILES **exibe diretório (C)**

FILES [*nome_de_arquivo*]

Exibe os nomes de arquivos de diretórios no formato dir/w do MS-DOS.

nome_de_arquivo

Especifica os nomes de arquivos a serem exibidos. *nome de arquivo* deve estar entre aspas (" "). Se omitido, todos os nomes de arquivos do diretório atual serão exibidos.

FRE (0)	
FRE ("")	**indica a memória livre (F)**

FRE (0) ou FRE ("")

Retorna a quantidade de memória livre. FRE ("") também remove as variáveis string não usadas da memória de variáveis. O comando *CLEAR* muda a memória de variáveis, portanto afetando a quantidade de memória disponível.

GOTO	**executa o número de linha de programa (S)**

GOTO *número_de_linha*

Executa o programa na memória começando em *número_de_linha*.

número_de_linha

Especifica o número de linha no qual terá início a execução do programa.

KEY — exibe definições de teclas de função (S)

KEY ON | OFF| LIST

Liga e desliga a exibição das teclas de função, ou lista as definições das teclas de função.

ON

Exibe as definições das teclas de função na 25a. linha da tela.

OFF

Apaga as definições das teclas de função na 25a. linha da tela.

LIST

Lista as definições das teclas de função na tela.

KEY — define as teclas de função (S)

KEY *número_da_tecla*, string

Muda as atribuições das teclas de função.

número_da_tecla

Especifica o número da tecla de função à qual será atribuída a string. O *número da tecla* pode ser de 1 até 10.

strings

Caracteres mostrados quando *a tecla_de_função* for pressionada. Se a string for " ", então a *tecla_de_função* será desativada. A string pode ter um tamanho máximo de 15 caracteres.

A *string* pode conter um caractere de controle, por exemplo, CHR$(13), o qual substitui a tecla Enter:

```
KEY 1,"FILES A:*.BAS"+CHR$(13)
```

LIST — lista um programa (C)

LIST [*da_linha* | .][-[*até_linha*|.]] [,*disp*|*nome_de_arquivo*]

Exibe linhas de programa na tela ou envia linhas de programa para um arquivo ou dispositivo.

da_linha

Especifica a primeira linha do programa a ser listado.

até_linha

Especifica a última linha do programa a ser listado.

Um traço separa *da_linha* e *até_linha* caso ambos os parâmetros sejam especificados. Se *até_linha* for omitido, o programa será listado começando desde *da_linha* até o fim do programa. Se *da_linha* for omitido, o programa será listado desde o início do programa até *até_linha*.

disp

Especifica o dispositivo no qual o programa será listado.

nome_de_arquivo

Especifica o nome de arquivo do programa listado.

A linha atual pode ser abreviada com um ponto.

Se ambos os parâmetros de linha forem omitidos, todo o programa será listado.

LLIST — lista um programa (C)

LLIST [*da_linha*|.][-.*até_linha*]]

Lista o programa atual na impressora de linha.

da_linha

Especifica a primeira linha do programa a ser listado.

até_linha

Especifica a última linha do programa a ser listado.

– Um traço separa *da_linha* e *até_linha* caso ambos os parâmetros sejam especificados. Se *até_linha* for omitido, o programa será listado começando desde *da_linha* até o fim do programa. Se *da_linha* for omitido, o programa será listado desde o início do programa até *até_linha*.

A linha atual pode ser abreviada com um ponto.

Se ambos os parâmetros de linha forem omitidos, todo o programa será listado.

LOAD — carrega um programa (C)

LOAD *nome_de_arquivo* [,R]

Executa um comando NEW e carrega um programa do disco.

nome_de_arquivo

Especifica o nome do programa BASIC a ser carregado do disco. *nome_de_arquivo* pode especificar um drive, caminho e uma extensão, e deve ser digitado entre aspas (" ").

Se o drive for omitido,será usado o drive atual. Se o caminho for omitido, será usado o diretório atual. Se a extensão for omitida, a extensão .BAS será assumida.

,R Executa imediatamente o programa após a carga.

MERGE **combina módulos de programa (C)**

MERGE *nome_de_arquivo*

Combina o programa na memória com um arquivo ASCII do disco.

nome_de_arquivo

Especifica o nome do programa BASIC a ser carregado do disco para ser combinado com o programa que está na memória.

nome_de_arquivo

Pode especificar um drive, caminho e uma extensão e deve ser digitado entre aspas (" ").

Se o drive for omitido, será usado o drive atual. Se o caminho for omitido, será usado o diretório atual. Se a extensão for omitida, a extensão .BAS será assumida.

nome_de_arquivo

Deve ser um programa BASIC no formato ASCII. Para criar tal arquivo, use o comando SAVE com ,A (por exemplo, SAVE *nome_de_arquivo,A).*

Se *nome_de_arquivo* contiver números de linha idênticos àqueles do programa na memória, a linha de programa de *nome_de_arquivo* substituirá a linha de programa na memória.

NEW — apaga a memória BASIC (C)

NEW

Apaga o programa na memória e inicializa todas as variáveis.

REM — insere comentários (S)

REM *texto* | 'texto

Insere comentários no programa.

texto Qualquer seqüência de caracteres. BASIC ignora todo o texto a partir do REM até a próxima linha de comando.

REM permite que você coloque comentários que descrevam o propósito de diferentes partes de um programa.

RENUM — renumera linhas (C)

RENUM [*nova_linha_de_início* | :.][,*da_linha*: | .]],*incremento*]

Renumera as linhas do programa atual.

nova_linha_de_início

Especifica o novo número inicial de linha da faixa.

da_linha

Especifica o número da linha na qual terá início a renumeração. Se omitido, BASIC começará renumerando a partir da primeira linha do programa.

incremento

Especifica o intervalo entre números de linhas sucessivos. O valor padrão é 10.

O ponto pode ser usado no lugar da linha atual.

RENUM substitui os números de linhas do programa atual. O primeiro número de linha a ser substituído será *da_linha*. Ele será substituído por *nova_linha_de_início*. O segundo número de linha será substituído por *nova_linha_de_início + incremento* etc.

RENUM também renumera as indicações de desvio dos comandos ON ERROR GOTO, GOTO, GOSUB etc., mas não atua sobre execuções condicionais com IF...THEN envolvendo a variável de sistema ERL. ERL armazena o número da linha onde ocorreu o último erro. A linha abaixo não é alterada por RENUM:

```
20 IF 1230=ERL THEN Sentença
```

A linha abaixo é tratada corretamente por RENUM:

```
20 IF ERL=1230 THEN Sentença
```

RUN — executa o programa atual (C)

RUN [*linha_de_início*|.]

Executa o programa atual.

linha_de_início

Especifica o número de linha na qual terá início a execução.

Se omitido, a primeira linha do programa será assumida.

O número de linha atual pode ser abreviado com um ponto.

RUN inicializa todas as variáveis numéricas com zero, variáveis string com uma seqüência nula e variáveis de matrizes como não-definidas. RUN então executa o programa na memória na *linha_de_início*. Se *linha_de_início* não for especificado, a execução começará na primeira linha do programa

RUN — carrega e executa um programa (C)

RUN *nome_de_arquivo* [,R]

Carrega e executa um programa do disco.

nome_de_arquivo

Especifica o nome de um programa BASIC a ser carregado do disco.O *nome_de_arquivo* pode especificar um drive, um caminho e uma extensão e deve ser digitado entre aspas (" ").

Se o drive for omitido, será usado o drive atual. Se o caminho for omitido, será usado o diretório atual. Se a extensão for omitida, a extensão .BAS será assumida.

,R Indica que quaisquer arquivos abertos devem permanecer abertos. Se omitido, quaisquer arquivos abertos serão fechados antes que RUN execute o programa.

RUN inicializa todas as variáveis numéricas com zero, variáveis string com uma seqüência nula e variáveis de matrizes como não-definidas. RUN então carrega o programa na memória e o executa.

SAVE — grava um programa (C)

SAVE *nome_de_arquivo* [,A | .P]

Grava o programa atual da memória para o disco.

nome_de_arquivo

Especifica o nome de um programa BASIC a ser gravado. *nome_de_arquivo* pode especificar um drive, um caminho, e uma extensão, e deve ser digitado entre aspas (" ").

Se o drive for omitido, será usado o drive atual. Se o caminho for omitido, será usado o diretório atual. Se a extensão for omitida, a extensão .BAS será assumida.

A Grava o programa atual no disco no formato ASCII. Se omitido, o programa será gravado no formato comprimido.

P Grava o programa atual no disco no formato protegido. Depois que um programa tiver sido gravado no formato protegido, ele não poderá ser listado ou editado.

SAVE grava uma cópia do programa atual na memória para um arquivo em disco denominado *nome_de_arquivo*. Se *nome_de_arquivo* já existir, ele será substituído.

STOP — pára de executar um programa (S)

STOP

Interrompe execução do programa.

A seguinte mensagem é exibida na tela:

Break in *número_de_linha*

Número_de_linha é a linha de programa contendo o comando STOP. A execução do programa pode ser retomada usando-se o comando CONT.

SYSTEM	**sai do GW-BASIC (C)**

SYSTEM

Sai do interpretador BASIC e retorna ao nível de comandos do MS-DOS. O conteúdo do programa atual na memória não será gravado automaticamente.

TRON:TROFF	**ativa / desativa o trace (indicador de acompanhamento) (C)**

TRON | TROFF

Liga e desliga o TRACE. Se ele estiver ativo (TRON), o interpretador BASIC exibe o número da linha de programa conforme ele vai sendo executado. Isso permite que você acompanhe a execução do programa.

TROFF desativa o TRACE.

TRATAMENTO DE ERRO

ERDEV — determina número de erro (F)

X=ERDEV

Retorna o número de erro indicado pelo MS-DOS através da interrupção 24 nos 8 bits mais baixos. Os 8 bits mais altos contêm informações (sob forma de um padrão de bits) sobre o controlador de dispositivo que ocasionou o erro.

ERDEV$

determina o nome do dispositivo causador do erro (F)

X$=ERDEV$

Retorna o nome do dispositivo que causou o erro:

PRN	Erro de impressora
A:,B:,C:	Erro no disco
outro	Erro de controlador de dispositivo

ERL — determina a linha de erro (F)

X=ERL

Retorna o número da linha na qual ocorreu o erro.

Não pode ser atribuído um valor a ERL.

ERR — **determina o número do erro (F)**

X=ERR

Retorna o número de código de um erro.

Não pode ser atribuído um valor a ERR.

ERROR — **gera um erro (S)**

ERROR *número_de_erro*

Simula um erro. Ele pode ser usado para testar rotinas de interceptação de erro, ou fazer tais rotinas interceptarem erros do usuário.

número_de_erro

O número do erro na forma inteira. A faixa dos números de erro é de 1 a 155.

Você não pode gerar os seus próprios textos de mensagens de erro. Um erro com o número especificado será gerado e à ERL será atribuído o número da linha da instrução ERROR.

ON ERROR GOTO — **intercepta erros (S)**

ON ERROR GOTO *número_de_linha*

Suspende a mensagem usual quando ocorrer um erro, muda para *número_de_linha* e executa a rotina de tratamento de erro que você desenvolveu.

número_de_linha

A linha de programa na qual começa a rotina de tratamento de erro. Se *número_de_linha* for 0, a interceptação de erro será desligada e GW-BASIC mostrará a mensagem usual.

RESUME — continua o programa após o erro (S)

RESUME [*número_de_linha* | EXT | 0]

Continua a execução do programa após ter tratado com um erro usando a sentença ON ERROR GOTO.

número_de_linha

Especifica o número de linha na qual a execução do programa deverá continuar.

NEXT

Se você especificar NEXT, o programa continuará na sentença logo a seguir da sentença que causou o erro.

0

A sentença que causou o erro será executada novamente. Isto é o mesmo que especificar RESUME sem parâmetros.

GERENCIAMENTO DE ARQUIVOS

CHAIN — encadeia overlays (arquivos)(S)

CHAIN[MERGE] *nome_de_arquivo*[*linha_inicial*][,ALL],[DELETE *faixa*]

Encadeia overlays ou partes adicionais de um programa.

MERGE MERGE foi apresentado antes. Ele tem o mesmo efeito quando usado com CHAIN: o programa carregado com MERGE é combinado com o programa na memória, de forma que as linhas existentes sejam substituídas, inseridas ou acrescentadas, dependendo de seus números de linha. A execução começa na *linha_inicial*.

nome_de_arquivo

Um nome de arquivo correspondendo às convenções do MS-DOS, consistindo em drive, caminho e nome de arquivo.

linha_inicial

A linha na qual a execução terá início após o CHAIN.

ALL

Esse parâmetro diz ao GW-BASIC para preservar todos os valores de variáveis existentes. Sem ALL, CHAIN apaga todas as variáveis e libera a memória. Se apenas algumas variáveis forem passadas, use COMMON.

DELETE *faixa*

DELETE pode somente ser usado junto com MERGE. DELETE apaga a faixa de linhas especificadas pela faixa antes que o programa seja carregado com MERGE.

O programa a ser encadeado deve ser gravado em formato ASCII. Isto é feito usando-se o comando SAVE *nome_de_arquivo*,A.

Se você usar MERGE sem ALL, todos os arquivos abertos serão fechados, todas as variáveis serão apagadas, e todas as estruturas abertas GOSUB...RETURN, FOR...NEXT, e WHILE...WEND serão finalizadas. CHAIN sem MERGE deixa todos os arquivos abertos. Programas que forem usados com MERGE não poderão conter nenhuma função definida com DEF FN, porque as funções são definidas no programa principal. CHAIN executa um RESTORE antes de carregar. Isto significa que o ponteiro DATA no programa encadeado deve ser redirecionado para READ. O valor indicado para matrizes através de OPTION BASE não será modificado.

Observação: RENUM não muda uma linha de início especificada, portanto isto deve ser corrigido pelo usuário.

CLOSE — fecha um arquivo (S)

CLOSE *[# número_de_arquivo,...]*

Grava o buffer de dados no disco e fecha o arquivo.

número_de_arquivo

O número de referência usado quando o arquivo foi aberto.

Depois que um arquivo foi fechado apropriadamente, todas as tentativas para gravar o arquivo, lê-lo, ou aplicar uma função a ele resultarão na mensagem de erro *Bad file number* ou *Bad file mode*.

COMMON **sobrepõe variáveis (S)**

COMMON *variável,...*

Define as variáveis que são passadas a um programa carregado com CHAIN.

variável

Especifica uma variável. Todos os tipos disponíveis de variáveis, incluindo matrizes, podem ser usados.

Um programa encadeado necessita da sentença COMMON somente se ele estiver carregando outros overlays para os quais os dados serão passados. A uma variável definida como COMMON deve ser dado um valor no programa que a chama. Se necessário, atribua 0 à seqüência de caracteres nula. Quando matrizes forem passadas, o comando DIM deve preceder COMMON.

CVD
CVI **descompacta campos (F)**
CVS

X=CVD(*var_CAMPO*)

X=CVI(*var_CAMPO*)

X=CVS(*var_CAMPO*)

Converte dados que foram compactados com MKD$, MKI$ ou MKS$ para o formato numérico normal.

var_CAMPO

O nome de um campo de arquivo ao qual foi atribuída uma string criada com MKD$, MKI$ ou MKS$.

CVD Retorna um resultado de precisão dupla.

CVI Retorna um resultado inteiro.

CVS Retorna um resultado de precisão simples.

Estas funções são eficazes somente se utilizadas em strings pré-processadas usando-se as funções MKD$, MKI$ ou MKS$. Se outras strings forem usadas, os seus conteúdos serão vistos como dados numéricos compactados, e serão convertidas inadequadamente.

EOF — teste para final de arquivo (F)

EOF (*número_de_arquivo*)

Testa para ver se foi alcançado o fim de arquivo ao ler o arquivo.

número_de_arquivo

O número de arquivo usado quando o arquivo foi aberto.

Para arquivos de acesso randômico, EOF retorna o valor -1 se for tentado um GET# após o último registro no arquivo, portanto o uso desta função não é necessário. EOF será ignorado se ele for parte de um registro.

FIELD# — define buffer de arquivo de acesso randômico (S)

FIELD# *número_de_arquivo,tamanho* AS *var_de_string,...*

Define os campos do buffer usado para ler/gravar um arquivo de acesso randômico.

número_de_arquivo

O número do arquivo usado quando o arquivo foi aberto.

tamanho AS *var_de_string*

Especifica os campos individuais do buffer. *tamanho* é o tamanho do campo da *var_string*. Todos os campos devem ser definidos em uma única linha de programa.

Os conteúdos de um campo especificado na sentença FIELD# não são atribuídos como seqüências de caracteres normais. Em vez disso, LSET e RSET são usados antes que sejam gravados no arquivo com PUT#.

Similarmente, depois que dados tiverem sido lidos do arquivo usando-se GET#, os campos definidos com FIELD podem ser atribuídos a outras variáveis.

A soma dos tamanhos individuais dos campos não pode exceder o tamanho do registro especificado no comando OPEN para aquele arquivo, senão será exibida a mensagem FIELD OVERFLOW. A soma dos tamanhos de arquivos pode ser inferior ao tamanho do registro especificado no comando OPEN para aquele arquivo.

GET# lê dados de um arquivo de acesso randômico (S)

GET# *número_de_arquivo* [,*número_de_registro*]

Lê um registro definido com FIELD do disco para o buffer.

número_de_arquivo

O número de arquivo usado quando o arquivo foi aberto.

número_de_registro

O número do registro a ser lido. Sem *número_do_registro*, GET# lê o registro para o qual o ponteiro de arquivos estiver

apontando. *número_de_registro* pode ter um valor de 1 a 16.777.215.

GET# lê quantos caracteres de um arquivo forem definidos pelo tamanho de registro na sentença OPEN, e armazena os dados nas variáveis definidas por FIELD. Se a soma dos tamanhos dos campos for menor que o tamanho do registro, os caracteres extras serão perdidos.

INPUT# — lê dados de um arquivo (S)

INPUT# *número_de_arquivo,variável,...*

Lê dados de um arquivo e designa-os para uma variável.

número_de_arquivo

O número de arquivo usado quando o arquivo foi aberto.

variável

Uma ou mais variáveis nas quais os dados do arquivo serão armazenados. Elementos de matrizes podem ser especificados.

INPUT lê registros completos do arquivo e designa-os para as variáveis. CR/LF (&H0D/&H0A) é usado como um separador entre os registros. Se PRINT# com os separadores " ou , (aspas ou vírgula) tiverem sido gravados no arquivo, estes também servirão como separadores de registros, e portanto não poderão fazer parte de uma string. Use a sentença LINE INPUT# para ler esses caracteres. Ao se ler dados numéricos, são usados espaços como separadores. INPUT# retorna a mensagem de Type mismatch se você tentar ler dados que não correspondam aos tipos de variáveis.

INPUT$ — lê caracteres de um arquivo (F)

X$ = INPUT$ (*num_caracs* [,*#número_de_arquivo*])

Lê o número especificado de caracteres de um arquivo.

num_caracs

O número de caracteres a serem lidos do arquivo. Já que o destino é uma string, pode ser especificado um valor entre 1 e 255.

número_de_arquivo

O número de arquivo usado quando o arquivo foi aberto.

KILL — apaga um arquivo (C)

KILL *nome_de_arquivo*

Apaga um arquivo.

nome_de_arquivo

Uma especificação de arquivo correspondendo às convenções do MS-DOS, consistindo em drive, caminho, e nome de arquivo.

A especificação de arquivo deve ser colocada entre aspas caso *nome_de_arquivo* seja usado como uma constante. Funções de string não podem ser usadas com o comando KILL, apesar de que a especificação de arquivo pode ser entrada como uma simples variável string. O arquivo a ser apagado deve estar fechado, ou aparecerá a mensagem File already open.

LINE INPUT# lê dados de um arquivo (S)

LINE INPUT# *número_de_arquivo, string*

Lê dados, incluindo separadores, de um arquivo.

número_de_arquivo

Número de referência usado quando o arquivo foi aberto.

string

Nome de uma variável string na qual o dado será armazenado. Essa pode ser um elemento de uma matriz de string.

LINE INPUT# lê os caracteres de um arquivo até que seja encontrado um CR/LF (&H0D/&H0A), ou até que 255 caracteres tenham sido lidos.

LOC determina a posição no arquivo (F)

X = LOC(*número_de_arquivo*)

Retorna a posição do ponteiro do arquivo. Para arquivos seqüenciais, o valor retornado será o número de blocos de 128 bytes que foram lidos ou gravados.

número_de_arquivo

O número de referência usado quando o arquivo foi aberto.

LOF — **determina o tamanho do arquivo (F)**

X = LOF(*número_de_arquivo)*

Determina o tamanho em bytes de um arquivo previamente aberto. O número retornado será um múltiplo de 128.

número_de_arquivo

Número de referência usado quando o arquivo foi aberto.

LSET
RSET — **formata dados de um arquivo de acesso randômico (S)**

LSET *var_CAMPO = var_string*

RSET var_CAMPO = var_string

Formata campos definidos com FIELD para armazenamento em um arquivo de acesso randômico. LSET justifica à esquerda os dados, e RSET justifica à direita os dados.

var_CAMPO

O nome de um campo definido na sentença FIELD.

string

Nome de uma variável de string cujos conteúdos serão atribuídos a *var_CAMPO.*

LSET e RSET também podem formatar variáveis string normais.

MKD$
MKI$ **conversão de números (F)**
MKS$

X$ = MKD$(*expressão*)

X$ = MKI$(*expressão*)

X$ = MKS$(*expressão*)

Converte uma expressão numérica em uma string para armazenamento compactado em arquivos de acesso randômico.

expressão

Pode ser uma constante, uma variável numérica, ou o resultado de uma função ou cálculo.

MKD$

O resultado de uma expressão numérica deve ser do tipo precisão dupla, e será convertido em uma string de 8 bytes.

MKI$

O resultado de uma expressão deve ser do tipo inteiro, e será convertido em uma string de 2 bytes.

MKS$

O resultado de uma expressão numérica deve ser do tipo precisão simples, e será convertido em uma string de 4 bytes.

Observação: Dados compactados com MKD$, MKI$, e MKS$ não poderão ser armazenados em arquivos seqüenciais porque o padrão de bits de um byte poderia corresponder a um separador, ou mesmo à marca de fim de arquivo.

NAME — renomeia um arquivo (C)

NAME *antigo_nome_de_arquivo* TO *novo_nome_de_arquivo*

Renomeia um arquivo.

antigo_nome_de_arquivo

Drive, caminho, e nome de arquivo do arquivo a ser renomeado. Drive e caminho são opcionais, mas o nome de arquivo deve ser dado com a sua extensão. A especificação pode ser uma constante ou uma variável.

novo_nome_de_arquivo

O novo nome do arquivo, dado com a sua extensão. *novo_nome_de_arquivo* pode ser uma constante string ou uma variável.

Quando constantes forem usadas para as especificações de arquivos, elas devem ser colocadas entre aspas. Funções de strings não são permitidas. Se o nome de arquivo especificado por *novo_nome_de_arquivo* já existir, será mostrada a mensagem de erro File already exists. A mensagem de erro Rename across disks será exibida se forem especificados drives diferentes. Será mostrado File already open caso você tente renomear um arquivo aberto.

OPEN — abre um arquivo (S)

OPEN *nome_de_arquivo* FOR *modo* AS # *número_de_arquivo*

[LEN = *tamanho_do_registro*]

Abre um arquivo para leitura/gravação, ou para usar uma função.

nome_de_arquivo

Uma especificação de arquivo correspondendo às convenções do MS-DOS. *nome_de_arquivo* pode especificar um drive, caminho e uma extensão. *nome_de_arquivo* deve ser digitado entre aspas (" ").

Se o drive for omitido, será usado o drive atual. Se o caminho for omitido, será usado o diretório atual. Uma extensão é opcional.

modo

Especifica a operação para o arquivo que está sendo aberto:

INPUT

O arquivo é aberto de forma que possam ser lidos dados dele. O ponteiro de arquivos é colocado no início do arquivo.

OUTPUT

O arquivo é aberto de forma que possam ser gravados dados nele. O ponteiro de arquivos é colocado no início do arquivo. Se o arquivo já existir, os dados presentes serão sobrepostos. Se o arquivo não existir, será criado.

APPEND

O arquivo é aberto no modo OUTPUT, porém o ponteiro de arquivos é colocado no final do arquivo de modo que o arquivo possa ser estendido. Se o arquivo não existir, será criado um.

número_de_arquivo

Um número de referência que é usado subseqüentemente para acesso ao arquivo enquanto aquele arquivo estiver aberto.

LEN=*tamanho_do_registro*

Se LEN for especificado em vez do parâmetro modo, o arquivo será aberto como um arquivo de acesso randômico. *tamanho_do_registro* indica o tamanho do registro em bytes. A faixa de *tamanho_do_registro* vai de 1 a 32767 bytes. LEN não é separado dos outros parâmetros por uma vírgula.

OPEN — abre um arquivo (S)

OPEN "modo", #*número_de_arquivo*, *nome_de_arquivo* [,*tamanho_do_registro*]

Esta é a segunda variante do comando OPEN. Ela é aceita para manter compatibilidade com versões anteriores do GW-BASIC.

número_de_arquivo

Um número de referência que é usado subseqüentemente para acesso ao arquivo enquanto aquele arquivo estiver aberto.

nome_de_arquivo

Uma especificação de arquivo correspondendo às convenções do MS-DOS. *nome_de_arquivo* pode especificar um drive, caminho e uma extensão. *nome_de_arquivo* deve ser digitado entre aspas (" ").

Se o drive for omitido, será usado o drive atual. Se o caminho for omitido, será usado o diretório atual. Uma extensão é opcional.

modo

Especifica a operação para o arquivo que está sendo aberto:

I O arquivo é aberto de forma que possam ser lidos dados dele. O ponteiro de arquivos é colocado no início do arquivo.

O O arquivo é aberto de forma que possam ser gravados dados nele. O ponteiro de arquivos é colocado no início do arquivo. Se o arquivo já existir, os dados presentes serão sobrepostos. Se o arquivo não existir, será criado um.

A O arquivo é aberto no modo OUTPUT, porém o ponteiro de arquivos é colocado no final do arquivo de modo que o arquivo possa ser estendido. Se o arquivo não existir, será criado um.

R O arquivo é aberto para acesso randômico.

tamanho_do_registro

O tamanho do registro deve ser especificado para o modo R. LEN=... é omitido nesta variante. O tamanho do registro é separado de outros parâmetros com uma vírgula. *tamanho_do_registro* pode ir de 1 a 32767.

PRINT# — gravar dados em um arquivo (S)

PRINT# *número_de_arquivo, expressão, ...* [;|,]

Grava dados em um arquivo.

número_de_arquivo

O número de referência usado no comando OPEN para este arquivo.

expressão

Dados a serem gravados no arquivo. Podem ser constantes, variáveis, ou o resultado de funções ou cálculos. Expressões múltiplas podem ser especificadas se elas forem separadas por vírgulas. As expressões podem ser de tipos diferentes.

;|, Após gravar uma expressão no arquivo, PRINT# coloca CR/LF (&H0D/&H0A) como um separador. Esta seqüência pode ser suprimida colocando-se um ponto-e-vírgula (;) após *expressão*. Os dados serão armazenados com separadores, mas não poderão ser lidos com INPUT#.

, Uma tabulação será gravada no arquivo. Oito espaços serão enviados ao arquivo antes que a próxima expressão seja gravada.

O número de arquivo deve ser igual ao número de arquivo previamente usado em um comando OPEN. Se expressões de tipos diferentes forem gravadas, elas devem ser atribuídas a variáveis dos tipos apropriados com INPUT# quando da leitura do arquivo.

PRINT# USING — grava dados formatados (S)

PRINT# *número_de_arquivo* USING "máscara"; *expressão*

Formata a saída de dados para um arquivo.

número_de_arquivo

O número de referência usado quando o arquivo foi aberto.

"máscara"

Especifica as características de formatação para *expressão*. Veja PRINT USING na seção "Saída de Tela" para uma descrição completa da "máscara".

expressão

Dados a serem gravados. Estes podem ser constantes, variáveis, resultados de cálculos ou funções. Os dados podem ser de qualquer tipo, mas a "máscara" deve ser de um formato compatível para tratar com os tipos específicos.

PUT# grava dados em um arquivo de acesso randômico (S)

PUT# *número_de_arquivo* [,*núm_reg*]

Grava em um arquivo em disco registros do buffer definido por FIELD.

número_de_arquivo

O número de referência usado quando o arquivo foi aberto.

núm_reg

O número do registro a ser gravado. Se omitido, PUT# grava o registro para o qual o ponteiro de arquivos estiver apontando no momento. *núm_reg* pode ser um valor indo de 1 a 16.777.215.

PUT# grava tantos caracteres do buffer quanto especificados pelo tamanho do registro no comando OPEN.

Se um número de registro incorreto for especificado, a mensagem Illegal function call será exibida. Se a soma dos tamanhos dos campos na sentença FIELD for inferior ao tamanho do registro, os dados a seguir do buffer na memória serão gravados no arquivo.

WRITE# grava dados em um arquivo (S)

WRITE# *número_de_arquivo*[SPC(*número*)][*expressão*...] [;]

Envia dados para um arquivo.

número_de_arquivo

O número de referência usado no comando OPEN.

SPC(*número*)

Grava o número especificado de espaços no arquivo.

expressão

Os dados a serem gravados. Estes podem ser constantes, variáveis, resultados de cálculos ou funções. Os dados podem ser de qualquer tipo. Se mais de um tipo for especificado, o seguinte separador será usado:

Envia uma tabulação, gravando oito espaços brancos no arquivo antes que o próximo item de dados seja gravado.

A sentença WRITE# corresponde ao comando PRINT#, com as seguintes exceções:

* Todas as expressões são colocadas entre aspas e separadas por vírgulas.
* A formatação com USING não é possível.
* WRITE# não pode ser usado com TAB.
* O ponto-e-vírgula (;) não pode ser usado para suprimir um avanço de linha.

GRÁFICOS

CIRCLE — desenha círculos e elipses (S)

CIRCLE[STEP](*xcentro,ycentro*),*raio*[,*cor*][,*início,fim*] [,*aspecto*]

Desenha um círculo ou uma elipse na tela gráfica nas coordenadas dadas.

xcentro,ycentro

Especifica a origem do círculo ou elipse. *(xcentro,ycentro)* sem STEP especifica as coordenadas absolutas do ponto. STEP *(xcentro,ycentro)* especifica a posição relativa à última posição endereçada na tela. Y pode ir de 0 a 199. X pode ir de 0 a 319 ou de 0 a 639, dependendo da resolução selecionada.

raio

Especifica o raio da elipse ou círculo em pixels. Se *raio* resultar em valores que caiam fora da resolução especificada com SCREEN, apenas as partes visíveis serão desenhadas.

início,fim

Especifica os pontos iniciais e finais de um círculo ou elipse. Os valores são ângulos em radianos. Valores de -2π a $+2\pi$ são válidos. Se os valores forem negativos, a origem será um arco conectado.

COLOR — seleciona cores gráficas (S)

COLOR[*primeiro_plano*] [,*fundo*] [,*cor_de_texto*]

Indica as cores para o modo de alta resolução gráfica (640x200 pontos).

primeiro_plano

Especifica ambas as cores de textos e gráficos. Os valores vão de 0 a 15.

fundo Este modo possui a cor padrão preta. Comandos gráficos que forneçam uma cor como um parâmetro se referem aos valores de primeiro plano e de fundo indicados com COLOR. Devido ao modo de alta resolução ser monocromático e não colorido, alguma conversão será necessária. Os valores das cores de palette são tomados e ajustados para a tela monocromática, de acordo com a seguinte tabela:

valor	cor
0	preto
1	branco
2	preto
3	branco

COLOR **seleciona cores gráficas (S)**

COLOR [*fundo*] [,*paleta*] [,*fundo_gráf*] [,*primeiro_plano_gráf*] [,*cor_de_texto*]

Indica as cores para o modo de média resolução gráfica (320x200 pontos).

fundo

Indica a cor de fundo e de borda para a tela inteira. Os valores podem ir de 0 a 15.

paleta

Especifica qual das duas palettes será usada para desenhar cores em comandos gráficos:

Paleta 0	Cor 0 = cor de fundo
Paleta 0	Cor 1 = verde
Paleta 1	Cor 0 = cor de fundo

fundo_gráf,primeiro_plano_gráf

Especifica os valores padrões para as cores de desenho usadas nos comandos gráficos, dependendo da palette selecionada previamente.

cor_de_texto

Especifica a cor de texto baseada na palette usada. Observe, por favor, que 0 não pode ser usado para *cor_de_texto*.

DRAW — desenha gráficos de variáveis (S)

DRAW *string*

Desenha um gráfico conforme descrito pela string.

string

Pode ser uma constante, ou o resultado de uma expressão de seqüência de caracteres. A posição atual na qual DRAW irá iniciar será ou o centro da tela ou a última posição usada por um comando gráfico. Outras posições deverão ser indicadas

antes de DRAW com um comando tal como PSET. Os seguintes parâmetros podem ser parte da string:

U *núm* — Desenha *núm* pontos para cima.

D *núm* — Desenha *núm* pontos para baixo.

L *núm* — Desenha *núm* pontos para a esquerda.

R *núm* — Desenha *núm* pontos para a direita.

E *núm* — Desenha *núm* pontos diagonalmente para a direita superior.

H *núm* — Desenha *núm* pontos diagonalmente para a esquerda superior.

F núm — Desenha núm pontos diagonalmente para a direita inferior.

G *núm* — Desenha *núm* pontos diagonalmente para a esquerda inferior.

Se o número não é uma constante, o valor numérico deve ser antes convertido para uma string. De outro modo, DRAW deve ser informado que uma variável está sendo usada com o sinal de igual (=) na frente da variável e um ponto-e-vírgula após a mesma:

B *dir* — A "caneta" move um pixel na direção indicada sem desenhar nenhum ponto. *dir* usa os caracteres U, D, L, R, E, F, G ou H (veja acima).

N *dir* — A "caneta" move um pixel na direção indicada, indicando aquele ponto. *dir* usa os caracteres U, D, L, R, E, F, G ou H (veja acima).

M *X,Y* — Se *X,Y* for especificado sem um sinal antecessor de mais ou menos, então uma linha será desenhada a partir do ponto atual até o ponto especificado por *X,Y* (coordenadas absolutas). Se *X,Y* for precedido por + ou -, então *X,Y* será visto como coordenadas relativas.

C *cor* — Indica a cor de desenho, baseada nas palettes disponíveis 0 a 3.

P *F,B* — Preenche uma região delimitada pela cor de borda *B* com uma cor de preenchimento *F*. A posição atual deve estar dentro da área delimitada. P não possui valores padrões, portanto ambos os parâmetros deverão ser especificados. Ambos os valores podem ir de 0 a 3.

A *ângulo* — Indica o ângulo para movimentos subseqüentes. Valores possíveis: 0 = 0 graus, 1 = 90 graus, 2 = 180 graus, 3 = 270 graus. Esta especificação tem precedência sobre a direção de movimento. Por exemplo, se você especificar o comando A3, então L irá para a direita em vez de para a esquerda.

TA *ângulo* — Especifica o ângulo do movimento a seguir. Ao contrário de A, TA pode ter valores que vão de -360 a +360.

S *fator* — Retorna o valor do *fator de fórmula/4* que deverá ser multiplicado com todos os parâmetros numéricos para o movimento a seguir. Isto permite que objetos sejam desenhados em escala com outros. fator pode ir de 0 a 255.

X *var_string* — Armazena procedimentos geralmente repetidos para inclusão posterior conforme necessário. As instruções contidas em *var_string* serão inseridas na posição atual e executadas. Lembre-se de incluir o ponto-e-vírgula logo após o nome da variável.

GET — armazena gráficos (S)

GET[STEP] (*X1,Y1*)-[STEP] (*X2,Y2*), *matriz*

Armazena gráficos em uma matriz, de modo que eles possam ser armazenados em disquete ou movidos na tela.

STEP (*X1,Y1*),STEP (*X2,Y2*)

(X1,Y1) são as coordenadas absolutas do canto inferior esquerdo. *(X2,Y2)* são as coordenadas absolutas do canto superior direito da área a ser armazenada. *Y* pode ir de 0 a 199. X pode ir de 0 a 319 ou de 0 a 639, dependendo da resolução selecionada.

matriz

Nome da *matriz* na qual o gráfico está armazenado. *matriz* deve ter sido dimensionada previamente para o tamanho apropriado com DIM. O tamanho pode ser calculado através da seguinte fórmula :

X = 4 + INT ((pontoX*B+7) /8) * pontoY

B= 1 para resolução 640x200

B= 2 para resolução 320x200

ponto X = largura do gráfico em pixels

ponto Y = comprimento do gráfico em pixels

O número de bytes que cada elemento da matriz pode armazenar depende do tipo de dados da matriz:

Inteiro:	2 bytes
Precisão simples:	4 bytes
Precisão dupla:	8 bytes

Lembre-se de levar isso em conta nos cálculos.

A dimensão X é armazenada no primeiro elemento da *matriz* (0), e a dimensão Y é armazenada no segundo elemento (1). Se a base da *matriz* for indicada como 1 com OPTION BASE 1, então o primeiro elemento será 1 e o segundo elemento será 2. Em seguida, está o mapa de bits para o gráfico. O mapa de bits é lido e armazenado linha por linha através do eixo X. Se os últimos pontos lidos do gráfico não preencherem um byte, o resto do elemento será preenchido com bits 0. Você deverá usar uma matriz do tipo inteira para armazenar o gráfico, porque os dados armazenados em tal matriz poderão ser manipulados mais facilmente.

Antes que o comando GET seja usado, o campo de coordenada deve primeiro ser definido com WINDOW.

LINE — desenha linhas e retângulos (S)

LINE [STEP] [(*X1,Y1*)] - [STEP] (*X2,Y2*) [,*cor*] [,B][F]] [,*estilo*]

Desenha linhas e retângulos.

X1,Y1

Especifica as coordenadas iniciais da linha a ser desenhada. Para um retângulo, elas especificam o canto inferior esquerdo. Se estas coordenadas forem omitidas, serão aplicadas as coordenadas do comando anterior. Se STEP não for especificado, X1,Y1 especificará as coordenadas absolutas do ponto.

STEP X1,Y1 especifica a posição relativa à última posição endereçada na tela. *Y* pode ir de 0 a 199. *X* pode ir de 0 a 319 ou de 0 a 639, dependendo da resolução.

X2,Y2

Especifica o ponto final da linha. Para um retângulo elas especificam o canto superior direito. Ambos os valores devem

ser fornecidos. Se STEP não for especificado, então *X2,Y2* especificará as coordenadas absolutas do ponto. *STEP X2,Y2* especifica a posição relativa à última posição endereçada na tela. *Y* pode ir de 0 a 199. *X* pode ir de 0 a 319 ou de 0 a 639, dependendo da resolução.

cor

A cor da linha. Os valores vão de 0 a 3.

B

Desenha um retângulo.

F

Um F logo a seguir do B preenche o retângulo com a cor especificada em COLOR.

estilo

Especifica a aparência da linha. Normalmente a linha é sólida. Uma máscara de 16 bits pode ser usada para mudar a aparência da linha.

LINE converte coordenadas que caiam fora da resolução máxima definida com SCREEN para valores permitidos a fim de evitar que o programa finalize com uma mensagem de erro.

PAINT preenche shapes (formas gráficas) com cores (S)

PAINT [STEP] (*X,Y*) [,*modo*] [,*cor_da_borda*]

Preenche uma área delimitada com uma determinada cor e/ou padrão.

(*X,Y*)

(*X,Y*) sem STEP especifica as coordenadas absolutas do ponto. STEP (*X,Y*) especifica a posição relativa à última posição

endereçada na tela. *Y* pode ir de 0 a 199. *X* pode ir de 0 a 319 ou de 0 a 639, dependendo da resolução.

modo

Especifica se a área será preenchida com uma cor ou um padrão. Se COLOR possuir um valor de 0 a 3, então ela especificará a cor a ser usada. De outro modo, você pode especificar uma máscara de 8 x 64 bits como o padrão (64 bytes). Este padrão será então desenhado no conjunto de cores padrões indicado por COLOR. O padrão é especificado com CHR$.

cor_da_borda

Especifica a cor da borda cercando a região a ser preenchida. Esta cor de borda é indicada pela borda com um valor que vai de 0 a 3.

Observação: Se pelo menos um único ponto na região não estiver desenhado, a cor ou padrão "vazará" através desse buraco e preencherá outras regiões.

PMAP — converte coordenadas (F)

X= PMAP (*coordenada*),*modo*

Converte as coordenadas modificadas com WINDOW.

coordenada

Especifica uma coordenada X ou Y convertida de acordo com modo.

modo= 0

Coordenada X especificada indicada com WINDOW e convertida para uma coordenada absoluta.

modo= 1

Coordenada Y especificada indicada com WINDOW e convertida para uma coordenada absoluta.

modo= 2

Uma coordenada absoluta X, convertida para as coordenadas de eixo indicadas com WINDOW.

modo= 3

Uma coordenada absoluta Y, convertida para as coordenadas de eixo indicadas com WINDOW.

POINT **determina a cor de um ponto gráfico (F)**

n= POINT(X,Y)

Retorna a cor do ponto nas coordenadas dadas.

(X,Y)

Especifica as coordenadas do ponto sendo testado. O resultado será entre 0 e 3 e indica o número da cor do ponto baseado na palette atual:

n Pode ser 0, 1, 2 ou 3:

0	Retorna a coordenada física X
1	Retorna a coordenada física Y
2	Retorna a coordenada da área X
3	Retorna a coordenada da área Y

PRESET — apaga um ponto gráfico (S)

PRESET[STEP] (*X,Y*) [,*cor*]

Apaga um ponto na tela gráfica nas coordenadas especificadas.

(*X,Y*) e STEP

Com *STEP, (X,Y)* especifica a posição relativa à última posição endereçada na tela. Para PRESET sem STEP, *(X,Y)* são coordenadas absolutas.

Y pode ir de 0 a 199. X pode ir de 0 a 319 ou de 0 a 639 dependendo da resolução.

cor

Especifica a *cor* do ponto. Se cor for omitido, o ponto será colorido com a cor de fundo atual, apagando-o. Os valores podem ir de 0 a 3.

PSET — desenha um ponto gráfico (S)

PSET [STEP] (*X,Y*) [,*cor*]

Desenha um ponto na tela gráfica nas coordenadas especificadas.

STEP

Quando usado com STEP, *(X,Y)* especifica a posição relativa à última posição endereçada na tela.

(X,Y)

Usando-se PSET sem STEP, *(X,Y)* são as coordenadas absolutas. Y pode ir de 0 a 199. X pode ir de 0 a 319 ou de 0 a 639, dependendo da resolução selecionada.

cor

Especifica a cor do ponto. Os valores podem ir de 0 a 3.

PUT — exibe o gráfico armazenado

PUT (*X,Y*), *matriz* [, *modo*]

Exibe um gráfico armazenado em uma matriz com GET.

(X,Y)

Especifica o ponto superior esquerdo do gráfico nas coordenadas *X* e *Y*. *Y* pode ir de 0 a 199. *X* pode ir de 0 a 319 ou de 0 a 639, dependendo da resolução.

matriz

A matriz que contém o gráfico a ser exibido.

modo

Especifica como o gráfico armazenado será exibido:

PSET	A exibição do gráfico será exatamente como ele foi lido.
PRESET	O gráfico será invertido, significando que um ponto armazenado como desenhado será exibido como apagado e um ponto

armazenado como apagado será desenhado na tela.

XOR

O gráfico será exibido ponto por ponto conforme a tabela:

Situação da Tela	Gráfico Armazenado	Resultado
desenhado	desenhado	apagado
desenhado	apagado	desenhado
apagado	desenhado	desenhado
apagado	apagado	apagado

AND

O gráfico será exibido ponto por ponto conforme a tabela:

Situação da Tela	Gráfico Armazenado	Resultado
desenhado	desenhado	desenhado
desenhado	apagado	apagado
apagado	desenhado	apagado
apagado	apagado	desenhado

OR

O gráfico será exibido ponto por ponto conforme a tabela:

Situação da Tela	Gráfico Armazenado	Resultado
desenhado	desenhado	desenhado
desenhado	apagado	desenhado
apagado	desenhado	desenhado
apagado	apagado	apagado

SCREEN — seleciona um modo gráfico (F)

SCREEN [*modo*] [,*cor*] [,*página_ativa*] [,*página_visualizável*]

Muda para o modo gráfico e indica a resolução.

modo Indica o modo gráfico:

0	modo texto
1	modo gráfico de média resolução
2	modo gráfico de alta resolução
100	modo especial para outras placas

cor

Se *cor* = 1 quando SCREEN for chamado, a cor atual será mantida. Se *cor* = 0, as cores padrões (fundo preto e cor de primeiro plano branco) serão indicadas para o modo escolhido.

página_ativa,página_visualizável

Existe apenas uma página no modo gráfico, portanto estes parâmetros são ignorados.

VIEW — define janela para gráficos (S)

VIEW [SCREEN] (*X1,Y1*) - (*X2,Y2*) [,*cor_de_fundo*] [,*cor_de_borda*]

Especifica as dimensões e a cor de uma janela usada para mostrar gráficos.

(*X1,Y1*) - (*X2,Y2*)

Especifica as coordenadas inferior esquerda e superior direita da tela, e define o tamanho da janela.

cor_de_fundo

A cor de fundo da janela. Os valores podem ir de 0 a 3.

cor_de_borda

Especifica a cor da borda em volta da janela. Esta borda será desenhada somente se houver espaço disponível. Se a janela for colocada em algum canto, somente as linhas visíveis dentro da tela serão desenhadas.

Depois de VIEW, todas as coordenadas usadas em comandos gráficos são aplicadas relativamente à borda de WINDOW. Por exemplo, coordenadas de 10,10 são lidas como 10 pixels para baixo e para a direita da borda desenhada pela janela. Se você desejar coordenadas absolutas, use VIEW com a opção SCREEN. Isto dá ao canto superior direito da janela as coordenadas 0,0.

WINDOW — muda coordenadas (S)

WINDOW [SCREEN] (*X1,Y1*) - (*X2,Y2*)

Define o sistema de coordenadas da área para a janela atual. Coordenadas da área têm 0,0 como o centro da tela. Os valores *X* acima do centro são positivos e abaixo do centro são

negativos. Os valores *Y* à direita do centro são positivos, e à esquerda do centro são negativos.

(X1,Y1) - (X2,Y2)

Estes parâmetros se aplicam aos cantos inferior esquerdo e superior direito da janela e especificam os novos valores da borda. A atribuição pode ser mudada com a opção SCREEN. *(X1,Y1)* então referir-se-á ao canto superior esquerdo e *(X2,Y2)* referir-se-á ao canto inferior direito da janela.

COMUNICAÇÕES DE E/S

GET# lê dados no buffer COM (S)

GET# *número_de_arquivo,número_de_caracteres*

Lê um número especificado de caracteres da interface serial para o buffer de comunicações.

número_de_arquivo

Refere-se ao número do arquivo usado em OPEN COM.

número_de_caracteres

O número de caracteres a serem lidos da interface para o buffer. *número_de_caracteres* não pode ser maior do que o tamanho de registro fornecido em OPEN COM.

Após os dados terem sido lidos da interface serial para o buffer, os comandos *INPUT#*, *LINE INPUT#*, e *INPUT$* podem ser usados para posterior processamento.

INP lê dados de uma porta (F)

X=INP (*porta*)

Retorna o valor atual de uma porta de dados ou de controle.

porta

Pode ser um valor entre 0 e 65535.

LOC
LOF — funções OPEN COM (F)

X=LOC(*número_de_arquivo*)

X=LOF(*número_de_arquivo*)

LOC retorna o número de caracteres que tiverem sido lidos no buffer de transferência.

LOF retorna o número total de caracteres que ainda restam no buffer de transferência.

número_de_arquivo

O número do arquivo usado em OPEN COM.

OPEN COM — abre canal de comunicações (S)

OPEN "COM canal: [*velocidade*] [*paridade*] [,*dados*] [,*parada*] [,RS] [,CS[*tempo*]] [,DS[*tempo*]] [,CD[*tempo*]] [,BIN] [,ASC] [,LF]" [FOR *modo*] AS # [*número_de_arquivo*] [,LEN= *tamanho*]

Abre e inicializa a interface serial para comunicações.

canal

Especifica o número da interface. Normalmente há somente uma interface serial presente, portanto você usaria 1.

velocidade

Indica o número de bits por segundo para transferência de dados. Este valor deve ser igual à velocidade do dispositivo através do qual você está tentando se comunicar. Se esta

especificação for omitida, *velocidade* terá o valor padrão de 300. Os valores possíveis são: 75, 110, 150, 300, 600, 1200, 1800, 2400, 4800, 9600.

paridade

Indica a paridade para a transferência de dados. Este valor deve ser igual à indicação no dispositivo periférico. O valor padrão é N (sem paridade). As seguintes abreviações podem ser usadas:

N	sem paridade
E	paridade par
U	paridade ímpar
S	espaço
M	marca

dados

Indica o número de bits por caractere. O valor padrão é 7 bits. Os valores possíveis são 5, 6, 7, e 8 bits por caractere.

parada

Indica o número de bits de parada enviados após cada caractere. Este valor deve ser igual à indicação no dispositivo periférico em questão. Se não for especificado ao contrário, 2 bits de parada serão enviados se a velocidade for menor ou igual a 110. Para todas as outras velocidades, um bit de parada será enviado. Outro valor possível é 1.5.

RS

Suprime o sinal RTS (Pedido para Enviar) durante a transmissão.

CS *tempo*

Espera pelo sinal CTS (Pronto para Enviar) do dispositivo periférico. *tempo* indica o tempo máximo de espera em segundos. Após este tempo uma mensagem Device Timeout será exibida.

DS *tempo*

Espera pelo sinal DSR (Conjunto de Dados Pronto) do dispositivo periférico. *tempo* indica o tempo máximo de espera em milissegundos. Após este tempo uma mensagem Device Timeout será exibida.

CD *tempo*

Espera pelo sinal CD (Detecção de Carro) do dispositivo periférico. *tempo* indica o tempo máximo de espera em milisegundos. Após este tempo uma mensagem Device Timeout será exibida.

BIN

Especifica que os dados recebidos devem ser tratados como dados binários. Todos os caracteres são processados sem mudança. CR ou LF não são interpretados como o fim da linha. O caractere EOF (&H1A) é ignorado.

ASC

Trata os dados recebidos como dados ASCII. O caractere de tabulação (CHR$ (9)) é convertido para espaços (CHR$ (32)). CR ou LF são lidos como um fim de linha. A transferência termina quando EOF (&H1A) for recebido.

LF

Especifica que um LF (&H0A, 10 decimal) deve ser enviado após cada CR (&H0D, 13 decimal). Esta opção de envio serve para impressoras que não executam automaticamente um avanço de linha após um retorno de carro.

modo

Indica o modo de transferência:

INPUT = recebe

OUTPUT = envia

Se você não especificar o modo FOR, a transferência é feita em modo randômico, significando que é possível o recebimento e o envio simultâneos.

número_de_arquivo

Especifica o número do arquivo para entrada/saída. Os valores possíveis vão de 1 a 15.

tamanho

Especifica o tamanho de registro para a transferência de dados. Os valores padrões são 256 bytes para INPUT e 128 bytes para OUTPUT. Se você chamou o GW-BASIC com o parâmetro [/C], o tamanho do registro não pode ser maior do que os valores especificados então.

Os parâmetros *velocidade, paridade, dados* e *parada* devem ser especificados na ordem em que eles foram mostrados acima. Todos os outros parâmetros e opções podem estar em qualquer ordem. O uso de BIN e LF ao mesmo tempo não tem sentido, porque CR e LF são ignorados quando BIN estiver ativo. A transferência será concluída com um CLOSE # número de arquivo.

OUT — grava dados numa porta (S)

OUT *porta,valor*

Grava um valor para uma porta de dados ou de controle.

porta

A porta na qual os dados serão gravados. Os valores possíveis são de 0 a 65535.

valor

Cada porta pode aceitar um único byte de dados, portanto valores de 0 a 255 são permitidos.

Consulte outras fontes a respeito de portas de E/S antes que você use OUT. **Valores inadequados podem causar travamento do sistema.**

PUT# — envia dados do buffer COM (S)

PUT# *número_de_arquivo,número_de_caracteres*

Grava um número específico de caracteres do buffer para a interface serial.

Os parâmetros correspondem àqueles dos comandos GET. Novamente, *número_de_caracteres* não pode ser maior que o tamanho de registro especificado em OPEN COM.

Os dados são gravados no buffer com os comandos PRINT#, WRITE#, e PRINT# USING.

WAIT — espera pela mudança da porta (S)

WAIT *porta, máscara_1, máscara_2*

Pára a execução do programa até que um determinado padrão de bits tenha sido lido de uma porta. O valor lido da porta é combinado com as máscaras da seguinte maneira:

valor_lido XOR *máscara_1* AND *máscara_2*

Se o resultado for zero, a porta será lida até que o resultado seja diferente de zero. O padrão de bits calculado estará então disponível para processamento, e a execução do programa continuará.

porta

Um valor entre 0 e 65535.

máscara_1,máscara_2

Os valores para uso na fórmula.

Observação: WAIT pode criar um loop infinito que não poderá ser interrompido com Ctrl+C ou Ctrl+Break.

WIDTH — indica o tamanho da linha para COM (S)

WIDTH "COM *canal*:", *número_de_caracteres*

Indica o número de caracteres enviados antes que um CR (&H0D) seja enviado.

canal

Especifica a interface serial sendo endereçada. Normalmente apenas uma interface está instalada, portanto este valor tem como padrão o número 1. Os valores podem ir de 1 a 4.

número_de_caracteres

O número de caracteres a enviar antes que um CR (&H0D) seja enviado.

TÉCNICAS DE INTERRUPÇÃO

KEY	programação de interrupções (S)

KEY *núm_tecla*, CHR$ (*código_hexa*) + CHR$ (*código_de_varredura*)

Define uma tecla ou combinação de teclas para detectar interrupções usando ON KEY.

núm_tecla

Um valor de 15 a 20.

código_hexa

Os seguintes valores podem ser usados para especificar teclas específicas:

Tecla Ctrl pressionada	04 ou &H04
Tecla Alt pressionada	08 ou &H08
Tecla NumLock ativada	32 ou &H20
Tecla Shift pressionada	64 ou &H40

Os valores listados acima dependem do teclado e do driver de teclado usado.

código_de_varredura

Os códigos de varredura não são os mesmos que os caracteres ASCII que representam as teclas. Um *código_de_varredura* é enviado pelo teclado; o valor deste código vai depender se a tecla foi pressionada ou liberada. Este valor é processado pelo driver de teclado e convertido

em um valor ASCII. Consulte o manual de operação de seu computador para o código de varredura.

Quando o KEY ( ) STOP for usado, a sub-rotina não será executada. Em vez disso, será indicado um sinal para a próxima sentença KEY ( ) ON para indicar que a condição é verdadeira. Quando KEY ( ) ON for encontrado novamente, a sub-rotina será então executada.

KEY ( ) STOP será então reinicializado para evitar um loop infinito. RETURN ao fim da rotina reabilita a checagem de interrupção de teclas.

Se ON ERROR GOTO estiver ativo, um salto para a rotina de tratamento de erro suspenderá a checagem de interrupção de teclas. RESUME reabilita a checagem. Para suprimir a reabilitação da checagem de interrupção de teclas, coloque um comando KEY ( ) OFF na rotina apropriada.

ON COM — programação de interrupções (S)

ON COM *canal* GOSUB *núm_linha* COM *canal* ON | OFF | STOP

A interface serial é lida durante a interrupção de sistema. Se um byte for recebido, uma sub-rotina será executada.

canal

Especifica o número da interface serial a ser lida. Os valores podem ir de 1 a 4, dependendo do número de interfaces instaladas.

núm_linha

Especifica o número da linha da sub-rotina a ser executada se um byte for recebido da interface serial.

COM *canal* ON

Habilita a checagem de interrupção.

COM *canal* OFF

Finaliza a checagem de interrupção.

COM *canal* STOP

Pára a checagem de interrupção até o próximo COM *canal* ON.

Quando STOP for usado, a sub-rotina não será executada se a condição for verdadeira. Em vez disso, um sinal é indicado para a próxima opção ON mostrando que a condição é verdadeira. Quando COM ON for encontrado novamente, a sub-rotina será então executada. COM STOP será então reinicializado para evitar um loop infinito. RETURN no fim da rotina reabilita a checagem de interrupções. Se ON ERROR GOTO estiver ativo, um salto para a rotina de tratamento de erro suspenderá a checagem de interrupção. RESUME reabilita a checagem de interrupção. Para suprimir a reabilitação da checagem de interrupção causado por RETURN ou RESUME, coloque um comando COM OFF na rotina apropriada.

ON KEY — programação de interrupções (S)

ON KEY (*núm_tecla)* GOSUB *núm_linha* KEY(X) ON | OFF | STOP

Lê as teclas de função e de controle durante a interrupção do sistema. Uma sub-rotina é executada se a tecla especificada for pressionada.

núm_tecla

Os seguintes valores podem ser usados para *núm_tecla*:

Tecla	Valor
F1 a F10	1 a 10
Cursor para cima	11
Cursor para a esquerda	12
Cursor para a direita	13
Cursor para baixo	14

núm_linha

Especifica o número da linha da sub-rotina que será executada quando a tecla especificada for pressionada.

KEY (*núm_tecla*) ON

Habilita a checagem durante a interrupção.

KEY (*núm_tecla*) OFF

Finaliza a checagem de interrupção.

KEY (*núm_tecla*) STOP

Suprime a checagem de interrupção até o próximo KEY (*núm_tecla*) ON.

Observação: Após uma interrupção para uma tecla especificada ter sido executada, o buffer de teclado estará vazio. Portanto, leituras posteriores com INKEY$ não serão possíveis. Quando KEY ( ) STOP for usado, a sub-rotina não será executada se a condição for verdadeira. Em vez disso, será indicado um sinal para que a próxima sentença KEY ( ) ON indique que a condição é verdadeira. Quando KEY ( ) ON for encontrado novamente, a sub-rotina será então executada. KEY ( ) STOP será então reinicializado para evitar um loop

para evitar um loop infinito. RETURN ao fim da rotina reabilita a checagem de interrupção de teclas.

Se ON ERROR GOTO estiver ativo, um salto para a rotina de tratamento de erro suspenderá a checagem de interrupção de teclas. RESUME reabilita a checagem. Para suprimir a reabilitação da checagem de interrupção de teclas, coloque um comando KEY ( ) OFF na rotina apropriada.

ON PLAY — programação de interrupções (S)

ON PLAY (*núm_notas*) GOSUB *núm_linha* PLAY ON | OFF | STOP

Checa o número de notas quando o buffer PLAY for checado. Se o número de notas for menor ou igual a *núm_notas*, a sub-rotina apropriada será executada. Isso pode ser usado para reinicializar PLAY para tocar música de fundo continuamente.

núm_notas

Número de notas restantes no buffer para ainda serem executadas pela sub-rotina.

núm_linha

O número de linha inicial da sub-rotina.

PLAY ON

Habilita a checagem de interrupção.

PLAY OFF

Desabilita a checagem de interrupção.

PLAY STOP

Pára a checagem de interrupção até o próximo comando PLAY ON.

Quando PLAY STOP for usado, a sub-rotina não será executada. Em vez disso, será indicado um sinal para a próxima sentença PLAY ON para indicar que a condição é verdadeira. Quando PLAY ON for encontrado novamente, a sub-rotina será então executada.

PLAY STOP será então reinicializado para evitar um loop infinito. RETURN ao fim da rotina reabilita o tratamento de interrupção de PLAY.

Se ON ERROR GOTO estiver ativo, um salto para a rotina de tratamento de erro suspenderá a checagem de interrupção de PLAY. RESUME reabilita a checagem. Para suprimir a reabilitação da checagem de interrupção de PLAY, coloque um comando PLAY OFF na rotina apropriada.

ON TIMER — programação de interrupções (S)

ON TIMER (*segundos*) GOSUB *núm_linha* TIMER ON | OFF | STOP

O timer (contador de tempo) é lido durante a interrupção do sistema e uma sub-rotina é executada, dependendo do resultado e do tempo especificado.

segundos

Intervalo no qual a sub-rotina em *núm_linha* será executada. Os valores podem ir de 1 a 86400.

núm_linha

Número da linha onde a sub-rotina a ser executada está situada.

TIMER ON

Habilita a checagem do timer durante a interrupção do sistema.

TIMER OFF

Finaliza a checagem do timer.

TIMER STOP

Suspende a checagem até o próximo TIMER ON.

Quando STOP for usado, a sub-rotina não será executada caso a condição seja verdadeira. Em vez disso, será indicado um sinal para a próxima opção TIMER ON para indicar que a condição é verdadeira. Quando TIMER ON for encontrado novamente, a sub-rotina será então executada. TIMER STOP será então reinicializado para evitar um loop infinito. RETURN ao fim da rotina reabilita a checagem de interrupção do timer.

Se ON ERROR GOTO estiver ativo, um salto para a rotina de tratamento de erro suspenderá a checagem de interrupção do TIMER. RESUME reabilita a checagem. Para suprimir a reabilitação da interrupção do timer após RETURN ou RESUME, uma sentença TIMER OFF deve ser colocada na rotina apropriada.

ENTRADA DE TECLADO

INKEY$ — entrada de teclado (F)

X$=INKEY$

Retorna o caractere ASCII da última tecla pressionada.

INKEY$ retorna uma string de dois bytes para teclas de função e de controle. O primeiro byte tem o valor 0. O segundo byte contém o valor ASCII da tecla pressionada. O tamanho da string deve ser determinado com LEN, e a string então tratada apropriadamente.

INPUT — entrada de teclado (S)

INPUT [;] ["*mensagem*"]; | , *variáveis*, ...

Lê dados do teclado. Pode ser mostrada uma mensagem para o usuário. Aspas e vírgulas não podem ser lidas.

;

Se um ponto-e-vírgula vir logo a seguir de INPUT, o avanço de linha será suprimido quando a entrada for finalizada com a tecla Enter.

"mensagem"

O texto pode ser especificado como uma constante de string que mostra uma mensagem pedindo ao usuário a entrada esperada.

;|, O ponto-e-vírgula suprime um avanço de linha após imprimir a mensagem (não implementado em todas as versões). A vírgula suprime o ponto de interrogação.

variáveis

Determina as variáveis nas quais serão armazenados os valores. A entrada deve corresponder aos tipos de variáveis. Diversas variáveis podem ser usadas, separadas por vírgulas. Também quaisquer dados digitados devem ser separados por vírgulas.

INPUT$ — lê caracteres (F)

X$=INPUT$ (*núm_caracs*)

Lê um número específico de caracteres do teclado.

núm_caracs

O número de caracteres a serem lidos.

LINE INPUT — entrada de teclado (S)

LINE INPUT [;] [*"mensagem"*] ;| , *var_string*

Lê uma linha de entrada do teclado. Os separadores usados no INPUT normal (" e ,) são permitidos como entrada.

; Se um ponto-e-vírgula vier logo a seguir de LINE INPUT, nenhum avanço de linha será impresso quando a entrada for terminada com a tecla Enter.

"mensagem"

Texto especificado como uma constante de string que pede ao usuário a entrada esperada.

; | , O segundo ponto-e-vírgula dentro da sentença suprime o envio de um avanço de linha após imprimir-se a mensagem. O envio automático do avanço de linha não está implementado em todas as versões. A vírgula suprime o envio do ponto de interrogação.

var_string

Variável de string na qual a entrada será armazenada. Não é possível a leitura de mais de uma variável de string.

LINGUAGEM DE MÁQUINA

CALL

CALLS chama uma rotina em linguagem de máquina (S)

CALL[S] *var_offset*[,*var_dados*]

Chama uma rotina em linguagem de máquina previamente carregada com BLOAD ou colocada na memória com POKEs.

[S]

CALLS é uma variante da sentença CALL. CALLS posiciona o segmento atual na pilha no formato byte baixo/alto antes do ponteiro de offset de 2 bytes. Compiladores especiais são necessários para passar o endereço segmentado desta forma.

var_offset

O endereço de offset no qual a rotina tem início é atribuído a uma variável inteira. O segmento apropriado deve ser indicado com DEF SEG antes de CALL.

var_dados

Os dados ou valores a serem processados são passados à rotina por variáveis. Um ponteiro de offset de 2 bytes para os conteúdos da variável é colocado na pilha no formato baixo/alto e pode ser processado dali.

DEF USR	(S)
USR	chama linguagem de máquina (F)

DEF USR *número= offset*

USR *número [var_dados]*

DEF USR indica a variável de offset. USR chama uma rotina em linguagem de máquina carregada com BLOAD ou colocada na memória com POKE.

número

Um valor entre 0 e 9. A definição anterior será válida, portanto USR(3), por exemplo, pode ser usado para chamar várias rotinas.

offset

O endereço de offset no qual a rotina começa. O segmento deve ser definido previamente com DEF SEG.

var_dados

As variáveis cujos conteúdos estão para serem processados. O valor não é passado pelo caminho da pilha, mas as informações serão carregadas nos registradores da CPU:

AL contém:

&H02, para variáveis inteiras

&H03, para variáveis string

&H04, para variáveis de precisão simples

&H08, para variáveis de precisão dupla

BX contém:

Um ponteiro de offset para a FAC onde o valor estiver armazenado (para valores numéricos).

DX contém:

Um ponteiro de offset para uma área de 3 bytes que contém o tamanho da string no primeiro byte e o endereço de offset para o texto da string nos dois bytes seguintes na ordem baixo/alto (para variáveis de string).

FUNÇÕES MATEMÁTICAS

Adição

Adição é indicada com o sinal de mais (+):

RESULTADO = *expressão_1* + *expressão_2*

Subtração

Subtração é indicada com o sinal de menos (-):

RESULTADO = *expressão_1* - *expressão_2*

Multiplicação

Multiplicação é indicada com o asterisco (*):

RESULTADO = *expressão_1* * *expressão_2*

Divisão

Divisão é indicada com a barra (/):

RESULTADO = *expressão_1* / *expressão_2*

Uma barra contrária (\) é usada para divisão inteira:

RESULTADO = *expressão_1* \ *expressão_2*

Exponenciação

Exponenciação é indicada com o circunflexo (^):

RESULTADO = *expressão_1* ^ *expressão_2*

Funções matemáticas estendidas

X = ABS(*expressão*)

Retorna o valor absoluto de *expressão*. O grau de precisão corresponde à declaração de tipo para as variáveis usadas na expressão.

X = Y AND Z

Quando dois valores forem combinados com *AND*, os bits serão avaliados de forma que o bit resultante será desligado se o bit correspondente tanto em *Y* ou *Z* estiver desligado, ou um bit será ligado se os bits correspondentes em *ambos Y* e *Z* estiverem ligados. *AND* pode ser usado para limpar bits específicos.

X = ATN(*expressão*)

O ângulo do arco tangente é calculado e retornado em radianos com precisão simples. Se o GW-BASIC foi chamado com o parâmetro [/D], o cálculo será executado em precisão dupla.

X = COS(*expressão*)

O co-seno da expressão é calculado e retornado em radianos com precisão simples. Se o GW-BASIC foi chamado com o parâmetro [/D], o cálculo será executado em precisão dupla.

X = Y EQV Z

Quando dois valores forem combinados com EQV, os bits serão avaliados de forma que o bit resultante seja ligado se os bits correspondentes em Y e Z forem os mesmos. De outro modo, o bit resultante será desligado.

X = EXP(*expressão*)

A constante *e* é elevada à potência de *expressão* e o valor retornado em precisão simples. Se o GW-BASIC foi chamado com o parâmetro [/D], o cálculo será executado em precisão dupla.

X = FIX(*expressão*)

Retorna a porção inteira de *expressão*. Qualquer fração após o ponto decimal será truncada. O resultado terá a mesma precisão

das variáveis na *expressão*. São permitidos valores positivos e negativos.

X = Y IMP Z

Quando combinados dois valores com IMP, os bits serão avaliados de forma que se os bits correspondentes em Y e Z forem os mesmos, o bit resultante será ligado. Se um bit for ligado em Y e desligado em Z, o bit resultante será desligado. Se um bit for desligado em Y e ligado em Z, o bit resultante será ligado.

X = INT(*expressão*)

Retorna o maior valor inteiro da *expressão*. O resultado corresponde às variáveis na expressão. São permitidos valores positivos e negativos.

X = LOG(*expressão*)

Retorna o logaritmo natural da expressão em precisão simples. Se o GW-BASIC foi chamado com o parâmetro [/D], o cálculo será executado em precisão dupla. *expressão* deve ser maior que zero.

X = *expressão_1* MOD *expressão_2*

Retorna o resto de uma divisão inteira (módulo).

X = NOT Y

NOT manipula os bits de modo que cada bit ligado em Y é desligado, e cada bit desligado em Y é ligado (invertido).

X = Y OR Z

Quando dois valores são combinados com OR, os bits são avaliados de forma que o bit resultante é ligado caso os bits correspondentes **tanto** em Y ou em Z forem ligados. OR pode ser usado para ligar bits específicos.

X = SGN(*expressão*)

Retorna o sinal de *expressão*. SGN retorna +1 se o sinal for positivo, -1 se for negativo, e 0 se a expressão for igual a zero.

X = SIN(*expressão*)

O seno de *expressão* em radianos é calculado e retornado (em radianos) com precisão simples. Se o GW-BASIC foi chamado com o parâmetro [/D], o cálculo será executado em precisão dupla.

X = SQR(*expressão*)

Retorna a raiz quadrada de *expressão* com precisão simples. Se o GW-BASIC foi chamado com o parâmetro [/D], o cálculo será executado em precisão dupla. A expressão deve ser maior ou igual a zero.

X = TAN(*expressão*)

A tangente da *expressão* em radianos é calculada e retornada com precisão simples. Se o GW-BASIC foi chamado com o parâmetro [/D], o cálculo será executado em precisão dupla.

X = Y XOR Z

Quando dois valores forem combinados com XOR, os bits serão avaliados de forma que o bit resultante será desligado caso os bits correspondentes em **ambos** Y e Z forem iguais. De outro modo, o bit resultante será ligado.

ACESSO À MEMÓRIA

BLOAD — carrega dados na RAM

BLOAD *nome_de_arquivo, offset*

Carrega um arquivo em uma determinada área da memória.

nome_de_arquivo

Especificação do arquivo a ser carregado.

offset

Especifica o endereço do segmento que foi indicado por último com DEF SEG, no qual o arquivo será carregado. Os dados serão lidos do arquivo até que seja lido um caractere EOF (&H1A).

BSAVE — grava dados de uma área da RAM (C)

BSAVE *nome_de_arquivo, offset, núm_bytes*

Grava uma determinada área da memória no disco.

nome_de_arquivo

Uma especificação de arquivo correspondendo às convenções do MS-DOS, consistindo em drive, caminho e nome do arquivo.

offset

Especifica o endereço dos dados a serem gravados, indicados por DEF SEG.

núm_bytes

Especifica o número de bytes a serem gravados do offset.

DEF SEG — indica o endereço de segmento (S)

DEF SEG = *ender_segmento*

Indica o endereço de segmento usado para BSAVE, BLOAD, PEEK, POKE, VARPTR, e VARPTR$.

ender_segmento

Especifica o segmento referente às funções e comandos. Os valores possíveis vão de 0 a 65535. Se você não especificar este parâmetro, o segmento de dados (DS) atualmente ocupado pelo GW-BASIC será indicado como segmento-padrão.

GW-BASIC não verifica se o segmento selecionado está ocupado por ele mesmo, por outros programas ou por dados.

PEEK — lê uma localização de memória (F)

X = PEEK(*offset*)

Retorna o valor de 8 bits do endereço de memória especificado no segmento atual.

offset

Endereço dentro do segmento (0 a 65535).

POKE grava um valor em uma localização de memória (F)

POKE *offset,valor*

Grava um valor em uma localização de memória no segmento atual.

offset

O endereço da localização de memória dentro do segmento. Valores podem ir de 0 a 65535.

valor

O valor a ser gravado na localização. Os valores podem ir de 0 a 255.

VARPTR acesso à variável de memória (F)

X=VARPTR(*variável*) ou X=VARPTR(*# núm_arquivo*)

Determina o deslocamento de uma variável no segmento de dados (DS).

variável

Qualquer tipo de variável suportada pelo GW-BASIC pode ser usada aqui, incluindo matrizes. À variável deve ser atribuído um valor antes que esta função seja chamada.

núm_arquivo

A segunda variante da função VARPTR retorna o endereço do primeiro byte da FCB (Bloco de Controle de Arquivos) para arquivos seqüenciais. Para arquivos de acesso randômico ela retorna o endereço do primeiro byte do buffer

FIELD. *núm_arquivo* corresponde ao número de arquivo usado quando o arquivo foi aberto.

O comando DEF SEG não tem efeito sobre esta função.

Observação: Os endereços das variáveis são modificados pela liberação de memória. O resultado da função não pode ser usado durante todo o programa - o endereço deve ser recalculado toda vez que for necessário.

VARPTR$ acesso ao armazenamento de variáveis (F)

X$=VARPTR$(*variável*)

Determina o tipo de variável e endereço de deslocamento de uma variável no segmento de dados (DS). O primeiro byte da string retornada contém informações sobre o tipo da variável:

CHR$ (2) = inteiro

CHR$ (3) = string

CHR$ (4) = precisão simples

CHR$ (8) = precisão dupla

O segundo e terceiro bytes contêm o endereço do primeiro byte da variável, no formato byte baixo/byte alto:

endereço = byte_baixo + 256*byte_alto

variável

Quaisquer dos tipos de variáveis suportados pelo GW-BASIC podem ser usados aqui, incluindo matrizes. À variável deve ser atribuído um valor antes que esta função seja chamada.

A função VARPTR$ não pode ser usada em FCBs ou buffers de acesso randômico! O comando DEF SEG não tem efeito sobre VARPTR$.

Observação: A liberação de memória modifica os endereços das variáveis.

MS-DOS e GW-BASIC

CHDIR **MKDIR** **RMDIR**	**acesso a diretórios (C)**

CHDIR *caminho*

MKDIR *nome_de_diretório*

RMDIR *nome_de_diretório*

Esses comandos operam como os seus equivalentes do MS-DOS.

nome_de_diretório,caminho

Estes parâmetros devem ser colocados entre aspas, ao contrário do MS-DOS, onde elas não são necessárias aos equivalentes de sistema destas palavras-chaves. Variáveis strings podem ser usadas, mas não são permitidas operações com as strings.

As abreviações MD, CD e RD são permitidas no MS-DOS, mas não o são no GW-BASIC.

ENVIRON	**acesso à tabela do ambiente (S)**

ENVIRON *entrada = atribuição*

MS-DOS armazena informações sobre o ambiente do sistema, tais como o caminho de busca ou a mensagem de prompt na tabela de ambiente. Esta tabela pode ser acessada do GW-BASIC.

entrada

Esta é uma entrada válida na tabela como PATH ou COMSPEC ou um parâmetro especificado por SET.

atribuição

Este é o novo parâmetro atribuído à *entrada*:

```
ENVIRON "PATH = C:\WORDSTAR"

ENVIRON "PROMPT = $p$_$n:"
```

A nova atribuição não pode ocupar mais caracteres do que a entrada atual.

ENVIRON$ — acesso à tabela de ambiente (F)

ENVIRON$ ("*entrada*") ou (*número*)

MS-DOS armazena informações sobre o ambiente do sistema (como o caminho de busca ou a mensagem do prompt) em uma tabela de ambiente que pode ser acessada do GW-BASIC.

entrada

Deve ser uma entrada de variável válida tal como PATH, PROMPT, ou COMSPEC. ENVIRON$ retorna a atribuição atual desta entrada:

```
PRINT ENVIRON$ ("PATH")

C:\WORDSTAR
```

ENVIRON$ também pode ser usado para acessar uma determinada entrada pelo seu número na tabela:

```
PRINT ENVIRON$ (1)
C:\WORDSTAR
```

IOCTL — acesso aos drivers do MS-DOS (S)

IOCTL # *núm_arquivo, seqüência_de_controle*

Envia uma seqüência de caracteres de controle para um driver de dispositivo (arquivo de controle do dispositivo) previamente aberto com OPEN. Por exemplo, ele pode ser usado para passar dados de formatação ou valores de inicialização. Os drivers padrões do MS-DOS não aceitam seqüências de caracteres IOCTL.

núm_arquivo

O número de referência usado quando o arquivo foi aberto.

seqüência_de_controle

Esta seqüência de caracteres contém os dados a serem avaliados pelo driver. Ela pode conter um máximo de 255 caracteres.

IOCTL$ — acesso aos drivers do MS-DOS (F)

X$ = IOCTL$ (*# núm_arquivo*)

Retorna a resposta de um driver de dispositivo a uma seqüência de caracteres de controle enviada por IOCTL.

núm_arquivo

O número de referência usado quando o arquivo foi aberto.

Os drivers padrões do MS-DOS não aceitam seqüências de caracteres IOCTL.

SHELL — executa outros programas do MS-DOS (C)

SHELL *nome_de_arquivo*

Chama um programa .COM, .EXE, ou .BAT do GW-BASIC.

nome_de_arquivo

Especifica o programa a ser executado. A especificação deve ser colocada entre aspas e é a mesma que a entrada no nível de comandos do MS-DOS. É possível que se use uma variável string, mas não são permitidas operações com strings após SHELL.

COMMAND.COM deve estar no disco no drive atual porque o MS-DOS precisa dele para todas as operações SHELL. Senão, GW-BASIC retorna a mensagem File not found e continua o programa. Por outro lado, GW-BASIC não exibe uma mensagem de erro se ele não conseguir encontrar o programa a ser executado. Após executar o comando SHELL, GW-BASIC continua o programa no comando seguinte ao comando SHELL. Você também pode chamar o COMMAND.COM através do SHELL e trabalhar no nível de comandos do MS-DOS. Após EXIT você será retornado ao GW-BASIC. Se *nome_de_arquivo* for omitido, COMMAND.COM será automaticamente carregado.

TIMER — acessa o relógio do sistema (F)

X = TIMER

Esta função retorna o tempo decorrido desde 0:00 horas (meia-noite) no formato *segundos:centésimos*.

Algumas versões retornam o tempo decorrido desde que o PC foi ligado, independentemente do tempo indicado.

SAÍDA-IMPRESSORA

LCOPY — cria uma cópia em papel (C)

LCOPY(*número*)

Imprime o conteúdo atual da tela na impressora. Não incluído em todas as versões do GW-BASIC.

número

Este número depende da versão de GW-BASIC que você tenha (geralmente 0, que imprime a tela de texto).

Observação: Como a maioria das versões do GW-BASIC, LCOPY trabalha somente em modo texto. Alguns discos de sistema PC são equipados com um programa chamado GRAPHICS.COM. Após ter chamado este programa, você pode imprimir uma cópia gráfica em papel pressionando Shift+PrtSc.

LPOS — determina a posição no buffer de impressora (F)

X=LPOS(*número_da_impressora*)

Retorna a posição do ponteiro de buffer no buffer de impressora.

número_da_impressora

O número da interface conectada à impressora. Pode ser 1, 2 ou 3. O valor padrão geralmente é 1.

LPRINT — envia dados à impressora (S)

LPRINT[TAB(*coluna*)] [SPC(*número*)] [*expressão...*] [; / ,]

Envia dados à impressora.

coluna

Indica uma determinada tabulação à cabeça de impressão. Posições de tabulação de impressora devem ser pré-indicadas para se usar corretamente esta opção.

número

Especifica o número de espaços a serem pulados.

expressão

Os dados a serem impressos. *expressão* pode conter constantes, variáveis, ou os resultados de funções ou cálculos. Os dados podem ser de qualquer tipo, e tipos diferentes podem ser combinados. Os dados são separados com um dos seguintes caracteres:

; Um ponto-e-vírgula suprime um avanço de linha após a saída. Isto permite que expressões múltiplas sejam impressas na mesma linha.

, Uma vírgula força uma tabulação durante a saída para a impressora. As tabulações são indicadas a cada 8 caracteres.

A impressora também pode ser aberta como um arquivo para a saída:

```
OPEN "LPTx:" FOR OUTPUT AS #1
PRINT#1, A$
```

X especifica o número de dispositivo que é o destino da saída. Os valores possíveis são 1, 2, ou 3. PRINT#, PRINT# USING, e WRITE# podem ser usados como sentenças para a saída. O arquivo deve estar fechado quando do fim da saída.

LPRINT USING — imprime dados formatados (S)

LPRINT USING "máscara"; *expressão*

Formata a saída de dados para a impressora.

"máscara"

Indica as características de formatação para *expressão*. Maiores detalhes podem ser encontrados na descrição de PRINT USING na seção "Saída-Tela" logo a seguir.

expressão

Especifica os dados a serem impressos. Estes podem ser constantes, variáveis, ou os resultados de funções ou cálculos. Os dados podem ser de qualquer tipo. Eles podem ser de tipos diferentes, mas "máscara" deve estar apto a acomodá-los.

WIDTH — indica caracteres por linha de impressão (S)

WIDTH LPRINT *núm_caracs*

WIDTH "LPT *núm_da_impressora*;",*núm_caracs*

Indica o número de caracteres por linha a serem impressos.

núm_caracs

Especifica o número de caracteres enviados à impressora antes que uma seqüência CR/LF seja enviada. O número de caracteres pode ir de 0 a 255.

núm_da_impressora

O número da interface conectada à impressora. Pode ser 1, 2, ou 3.

O valor padrão para a saída na impressora é de 80 caracteres por linha em algumas versões do GW-BASIC.

SAÍDA-TELA

PRINT	saída para a tela (S)

PRINT [TAB(*coluna*)] [SPC(*número*)] [*expressão...*] [;] [,]

Exibe dados de qualquer tipo na tela.

coluna

Move o cursor para uma coluna da tela na linha de saída atual. *coluna* pode ter um valor de 1 a 80, ou de 1 a 40, dependendo de WIDTH.

número

Pula o número especificado de espaços na linha de saída atual.

expressão

Especifica os dados a serem impressos. *expressão* pode conter constantes, variáveis, e expressões numéricas ou strings. Os dados podem ser de qualquer tipo, e os tipos podem ser misturados. Eles são separados por um dos seguintes separadores:

; Começa a imprimir o próximo valor na próxima coluna de tela enquanto suprime o retorno de carro. Isto permite que expressões múltiplas sejam impressas na mesma linha.

, Começa a imprimir o próximo valor na próxima parada de tabulação (a cada 8 colunas).

Se a expressão não couber na linha atual, a saída será automaticamente continuada na linha seguinte. Quando imprimindo valores numéricos, a primeira posição representa o sinal do número. Para valores positivos será pulado um espaço. Para valores negativos será impresso um sinal de menos (-).

O teclado também pode abrir arquivos para saída de tela:

```
OPEN"CONS:" FOR OUTPUT AS #1

PRINT#1,PRINTER OUTPUT$
```

PRINT#, PRINT# USING e WRITE# são também sentenças de saída. Após a saída, o arquivo deve ser fechado com a sentença CLOSE.

PRINT USING — saída formatada (S)

PRINT USING *máscara;expressão*

Exibe dados formatados na tela.

máscara

Uma constante de string ou variável que indica as características de formatação para *expressão*:

Máscaras para Dados Numéricos

"#####"

Esta máscara formata inteiros. Cada # significa um dígito no número a ser impresso. Se for necessário um sinal, você deve colocar mais um #. Números reais são arredondados antes que eles sejam impressos.

"#####.##"

Esta máscara formata números decimais. Os caracteres # antes e depois do ponto decimal determinam o número de casas a serem impressas antes e depois do ponto decimal. Se você não fornecer caracteres # suficientes após o decimal, o valor será arredondado para o número de casas especificadas.

"+#####.##" [ou] "#####.##+"

Um sinal de mais (+) no início ou final da máscara exibe o sinal do número antes ou depois do valor.

"#####.##-"

Um sinal de menos (-) no final da máscara exibe um sinal negativo no final do valor. Valores positivos são impressos sem um sinal de mais (+). O sinal de menos pode aparecer somente no final da máscara.

"#####.##^^^^"

Exibe um valor em notação exponencial (notação científica). Os caracteres ^^^^ significam E ou D, o sinal (+/-) e dois dígitos que representam as potências de dez.

"**#####.##"

Se dois asteriscos forem colocados no início da máscara, os espaços que são normalmente impressos na máscara caso o

valor seja menor que a largura da máscara serão substituídos por asteriscos.

"$$#####.##"

O sinal de dólar ($) no início da máscara faz que o mesmo sinal seja colocado antes de cada valor. Não é possível usá-lo com a notação científica ^^^^. Quando usado em combinação com a máscara de asteriscos do exemplo anterior, somente um sinal de dólar é especificado (**$#####.##).

"#######,.##"

Uma vírgula na máscara imprime os dígitos do valor em grupos de três, separados por vírgulas (por exemplo, 1,254,267.80). Use este formato com números que tenham quatro ou mais dígitos.

Máscaras para Strings e Caracteres

"\ \"

Esta máscara especifica o número de caracteres a serem impressos. Os caracteres de barra invertida (\) indicam um caractere cada. Os espaços entre os caracteres \ representam caracteres adicionais. Por exemplo, se você colocar cinco espaços entre os caracteres \, um total de 7 caracteres serão impressos.

"&" O E comercial imprime a string como não-formatada.

"!" O ponto de exclamação imprime somente a primeira letra da string.

"_" [sublinhado]

Se PRINT USING encontrar o caractere sublinhado na máscara, os caracteres que vierem logo a seguir a este serão impressos sem formatação. Isto lhe permite incluir caracteres que normalmente não são permitidos na máscara.

"#####.## *texto*"

Texto dentro da máscara é impresso na posição relativa dentro da máscara.

;

O ponto-e-vírgula é usado como um separador entre a máscara e a expressão.

expressão

Estes são os dados a serem impressos. *expressão* pode conter constantes, variáveis, resultados de cálculos, ou resultados de funções. Os dados podem ser do mesmo tipo ou de tipos diferentes, mas eles devem estar aptos a serem formatados pela máscara.

Se o valor especificado for maior do que a máscara pode formatar, PRINT USING indica isto colocando um sinal de porcentagem (%) precedendo a saída. Isto pode destruir qualquer formatação subseqüente. Se o número máximo de espaços imprimíveis não puder ser determinado exatamente com antecedência, uma máscara maior deve ser selecionada, ou você deve incluir um teste para tamanho de valores dentro do seu programa.

WRITE — saída para a tela (S)

WRITE [SPC(*número*)] [*expressão*...] [,]

Envia dados para a tela.

SPC(*número*)

Pula o número de espaços especificados por *número* na linha de saída atual.

expressão

Os dados a serem impressos. *expressão* pode conter constantes, variáveis, resultados de cálculos, ou resultados de funções. Os dados podem ser de tipos misturados. O seguinte separador é usado para separar os dados:

, Uma vírgula move o GW-BASIC para a próxima tabulação (a cada 8 caracteres) antes de imprimir o próximo item de dados.

O comando WRITE é similar ao comando PRINT, com as seguintes diferenças:

* Todas as expressões devem ser colocadas entre aspas e separadas por vírgulas.
* Formatação usando-se USING não é possível.
* A função TAB não pode ser usada com WRITE.
* O ponto-e-vírgula (;) não pode ser usado para suprimir o avanço de linha.

CONTROLE DE TELA

CLS	limpa a tela (S)

CLS

Limpa a tela e move o cursor para o canto superior esquerdo.

COLOR	indica a cor de texto/fundo (S)

COLOR [*cor_de_texto*] [,*cor_de_fundo*]

Indica as cores de texto e de fundo da tela.

cor_de_texto

Indica a cor na qual as letras e caracteres serão exibidos na tela. Use um valor inteiro entre 0 e 15 de acordo com a tabela de cores na página seguinte.

cor_de_fundo

Indica a cor do fundo no qual o texto será exibido. Use um valor inteiro de 0 a 7 de acordo com a tabela de cores na página seguinte.

Monitor Colorido

Se você estiver usando um monitor colorido e uma placa normal CGA, as cores de fundo são limitadas de 0 a 7. Se estiver usando uma placa EGA com um monitor compatível, você poderá selecionar uma cor de fundo entre 0 e 15.

Monitor Composto

Aqui as cores selecionadas são exibidas como padrões de estilo ou níveis de cinza, dependendo da qualidade do monitor. Algumas cores são portanto invisíveis.

Monitor Monocromático

Aqui não há cores, somente os atributos:

Intensidade	Atributo
Normal	(COLOR 7,0)
Intensidade Dupla	(COLOR 15,0)
Sublinhado	(COLOR 1,0)
Vídeo Reverso	(COLOR 0,7)
Intermitente	(COLOR 31,0)

Tabela de Cores

Valor da Cor	Cor	Valor da Cor	Cor
0	Preto	8	Cinza-escuro
1	Azul	9	Azul-claro
2	Verde	10	Verde-claro
3	Ciano	11	Ciano-claro
4	Vermelho	12	Vermelho-claro
5	Magenta	13	Magenta-claro
6	Marron	14	Amarelo
7	Cinza-claro	15	Branco

Colocando o valor 16 para *cor_de_texto*, o texto fica intermitente.

CSRLIN — determina a localização do cursor (F)

X=CRSLIN

Retorna a localização atual do cursor.

Se as extremidades da tela forem alteradas com VIEW PRINT, CRSLIN retornará a linha atual dentro da janela.

LOCATE — posiciona o cursor (S)

LOCATE [*linha*] [,*coluna*] [,*liga_desliga*] [,*da_linha*] [,*até_linha*]

Posiciona o cursor na tela, liga-o ou desliga-o, e indica a forma do cursor.

linha, coluna

A linha da tela e a coluna dentro daquela linha onde o cursor estiver posicionado. A *linha* pode ter um valor entre 1 e 25. O valor de *coluna* depende do limite indicado com WIDTH e pode ir de 1 a 80, ou 1 a 40.

liga_desliga

Este parâmetro determina se o cursor estará ligado (1) ou desligado (0).

da_linha, até_linha

O cursor pode ser exibido de diversas formas. Dependendo da placa de vídeo usada, o cursor será composto de uma série

de linhas de estilo. A linha do topo é 0, e a linha máxima inferior é 31. O fator determinante aqui é a matriz que a placa de vídeo usa para exibir os caracteres:

Monitor colorido e composto: 0 a 7; matriz = 8x8

Monitor monocromático: 0 a 31; matriz varia dependendo da placa.

POS(0) — determina a posição do cursor (F)

X=POS(0)

Determina a posição atual da coluna do cursor.

SCREEN — seleciona a página de vídeo (S)

SCREEN [*modo*] [,*cor*] [,*página_de_saída*] [,*página_de_exibição*]

Indica o modo de texto e a cor de texto, seleciona as páginas de saída e de exibição.

modo

Indica o modo de operação da tela:

0	Modo texto	80x25 ou 40x25
1	Modo gráfico	320x200 pixels
2	Modo gráfico	640x200 pixels
x	Modo gráfico	640x400 pixels
	↑(somente versão 2)	↑(somente placas mono)

cor

Para 1, a cor indicada com COLOR permanece a mesma. Para 0, a cor de texto é definida para o valor 7 (cinza-claro) e o fundo para 0 (preto).

página_de_saída

Especifica a página para a qual será enviado o texto. Uma tela de 80x25 possui quatro páginas (0 a 3) disponíveis, e uma tela de 40x25, oito páginas.

página_de_exibição

Especifica qual página será exibida. Uma tela de 80x25 possui quatro páginas (0 a 3) disponíveis, e uma tela de 40x25, oito páginas.

A sua opção de páginas de exibição e de saída depende de a sua placa gráfica suportar esse modo. Como regra geral, as placas gráficas coloridas suportam-no. Consulte a sua documentação de hardware e verifique se a sua placa gráfica monocromática suporta esse modo.

SCREEN — determina caractere e atributo (F)

X=SCREEN (*linha,coluna* [,0|1])

Retorna o código ASCII ou atributo de um caractere exibido na tela.

linha,coluna

As coordenadas do caractere na tela. Dependendo de WIDTH, a coluna pode ter um valor entre 1 e 80, ou 1 e 40.

0 | 1

Se você especificar 0 aqui, SCREEN retornará o código ASCII do caractere em questão. Para 1, o atributo de exibição do caractere será retornado.

VIEW PRINT **indica a janela de texto (S)**

VIEW PRINT [*da_linha*] [TO] [*até_linha*]

Define uma janela de texto.

da_linha,até_linha

Apenas faixas compostas de linhas completas podem ser definidas como janelas de texto. Digite a primeira linha da faixa como *da_linha*, e a última linha da faixa como até_linha.

WIDTH **indica o número de caracteres/linha (S)**

WIDTH *núm_caracs*

Especifica o número de caracteres por linha para a saída em tela.

núm_caracs

Especifica 40 ou 80 caracteres por linha. Digite 40 ou 80.

WIDTH automaticamente limpa a tela inteira antes de indicar a largura. Somente um modo é possível por vez.

SENTENÇAS DE SOM

BEEP	emite um tom simples (S)

BEEP

Emite um tom simples.

PLAY	programa de seqüências de tons (S)

PLAY *string*

Emite o conjunto de tons definidos na string.

string

Pode conter as seguintes especificações para definir os tons a serem criados:

MF (modo de primeiro plano)

Gera tons em primeiro plano. Um nova sentença PLAY não poderá ser executada até que a atual tenha sido completada.

MB (modo de fundo)

Gera tons de fundo. O programa continua a ser executado. Um máximo de 32 caracteres para geração de tons podem ser passados.

MN (modo normal)

Gera tons de duração normal (7/8 da duração especificada por L).

ML (modo de tons contínuos)

Gera tons *legato* (contínuos), conforme especificado na seqüência de caracteres por L.

MS (modo de tons curtos)

Gera tons *staccato* (curtos), 3/4 da duração especificada com L.

Ox (oitava)

Especifica a oitava desejada com x. O valor de x pode ir de 0 a 6.

> Aumenta o tom em uma oitava, até um máximo de 6.

< Diminui o tom em uma oitava, até um mínimo de 0.

A - G[-] [+| #]

Uma nota musical de A até G. O sinal de número (#) ou sinal de mais (+) aumenta a nota um meio-tom (sustenido), enquanto o sinal de menos (-) diminui a nota um meio-tom (bemol).

N *número_da_nota*

Toca uma das possíveis 84 notas dentro das sete oitavas disponíveis. Os valores podem ir de 0 a 84. 0 produz uma pausa.

P *duração_da_nota*

Produz uma pausa. A duração da pausa corresponde à *duração_da_nota*, que pode ser um valor de 1 a 64. *duração_da_nota* =4 provoca uma pausa de um compasso.

L *duração_da_nota*

Indica a duração das notas subseqüentes. *duração_da_nota* pode ser especificada na faixa de 1 a 64. Se L com um parâmetro vier logo após uma nota, L se aplicará somente a esta nota.

T *tempo*

Indica o tempo para a geração de tons - o número de compassos por minuto. *Tempo* pode ir de 32 a 255.

O ponto faz uma geração de tom durar uma vez e meia (3/2) a mais do que os valores indicados por L e T. Múltiplos pontos aumentam o valor correspondentemente.

X*var_string;*

Executa *var_string*. Isto corresponde, grosso modo, a uma "sub-rotina musical". Lembre-se de incluir o ponto-e-vírgula após a variável.

As variáveis também podem ser usadas para parâmetros numéricos. Isto pode ser indicado na string que se segue:

"T=Tempo;"

Observe que (em contraste à X*var_seqüência*) um sinal de igual (=) deve ser usado.

SOUND — emite tom variável (S)

SOUND *freqüência, duração*

Emite um tom de freqüência e duração variáveis.

freqüência

A freqüência do tom a ser gerado. O valor para *freqüência* pode ir de 37 a 32767.

duração

Duração do tom resultante medido em ciclos horários (18.2 por segundo). Os valores possíveis vão de 0 a 65535.

Tratamento de Seqüência de Caracteres

+	concatena strings(F)

X$ = *string_1* + *string_2* + *string...*

Concatena strings.

string_x

Constante de string, variável, ou o resultado de uma função de string.

O tamanho total da string não pode exceder 255 caracteres. Espaços não são inseridos antes ou depois da string.

DATE$	**data do sistema (F,S)**
TIME$	**hora do sistema (F,S)**

X$ = DATE$ ou DATE$ = X$

X$ = TIME$ ou TIME$ = X$

Retorna ou indica a hora e data do sistema do MS-DOS.

Dependendo da configuração do sistema operacional, DATE$ possui um dos seguintes formatos:

```
10 DATE$ = "mês-dia-ano"
```

(ou)

```
10 DATE$ = "dia-mês-ano"
```

TIME$ possui o seguinte formato:

10 TIME$ = "hrs:min:seg"

(ou)

10 TIME$ = "hrs:min:seg:cent"

DEF FN	**(S)**
FN	**funções definidas pelo usuário (F)**

DEF FN *nome_da_string (parâmetro...) = função*

FN *nome_da_string (parâmetro...)*

DEF FN permite ao usuário definir uma função para operações com strings. FN chama essas funções definidas pelo usuário.

nome_da_string

Especifica o nome da função que será usado quando chamada com FN. Este deve ser um nome válido de variável string. Matrizes não podem ser usadas.

parâmetro

Especifica um ou mais nomes de variáveis de string. Estas variáveis são definidas localmente, significando que elas se aplicam somente a esta função. O número de variáveis aqui deve ser igual ao número de parâmetros quando da chamada.

função

As operações executadas nos parâmetros. Todas as funções de strings suportadas pelo GW-BASIC podem ser usadas.

Uma função definida desta maneira não pode chamar a si própria. A função, que deve ser definida com DEF FN antes da

primeira chamada, não pode ser maior do que uma linha de programa. Definições não são permitidas no modo direto.

FRE("") limpa o armazenamento de strings (F)

FRE("")

Força uma liberação para limpar a área de armazenamento de strings.

INSTR busca por caractere na string (F)

INSTR ([*da_posição*], *string*, *carac_de_busca*)

Retorna a posição de um caractere na string.

da_posição

Especifica a posição na string onde terá início a busca. Se omitido, a busca terá início no primeiro caractere.

string

A string a ser vasculhada. Esta também pode ser um elemento de uma matriz de string.

carac_de_busca

O caractere a ser encontrado na string. Este pode ser uma constante, uma variável, ou o resultado de uma função de string.

O caractere de busca também pode ser uma string, mas INSTR somente buscará pelo primeiro caractere, não a string inteira.

LEFT$ **secciona uma string (F)**

X$ = LEFT$ (*string, núm_caracs*)

Retorna o número de caracteres à esquerda de uma string.

string

A string a ser seccionada. Elementos de matrizes podem ser especificados.

núm_caracs

O número de caracteres (começando na posição inicial) a serem retornados da *string*.

LEN **informa o tamanho de uma string (F)**

X = LEN(*string*)

Retorna o tamanho de uma string.

string

A string cujo tamanho será informado. Elementos de matrizes podem ser especificados.

MID$ **disseca uma seqüência de caracteres (F)**

X$ = MID$(string, *primeiro_carac* [,*núm_caracs*])

Retorna o número especificado de caracteres começando em uma determinada posição dentro da string.

string

A string a ser seccionada. Elementos de matriz podem ser usados.

primeiro_carac

A posição dentro da string na qual a operação terá início.

núm_caracs

O número de caracteres a ser retornado. Se este parâmetro não for especificado, o restante da string será retornado.

MID$ — modifica partes da string (S)

MID$ (*string_dest, primeiro_carac* [,*núm_caracs*]) = *string_fonte*

Substitui uma parte da string por outra.

string_dest

Especifica a string a ser substituída. Elementos de matrizes podem ser especificados.

primeiro_carac

Indica o caractere na string de destino no qual a string-fonte será inserida.

núm_caracs

Especifica o número de caracteres da string de destino a serem substituídos. Se este parâmetro não for dado, todos os caracteres após *primeiro_carac* serão substituídos.

string_fonte

Especifica a string que substituirá os caracteres da string-fonte começando em *primeiro_carac*. Elementos de matrizes podem ser especificados.

RIGHT$ **secciona uma string (F)**

X$ = RIGHT$ (*string, núm_caracs*)

Retorna o *núm_caracs* à direita na string.

string

Especifica a string a ser seccionada. Elementos de matrizes podem ser especificados.

núm_caracs

Número de caracteres (começando na posição mais à direita da seqüência) a serem retornados.

SPACE$ **retorna espaços (F)**

X$ = SPACE$(*número*)

Retorna uma string consistindo em um número especificado de espaços.

número

Especifica o número de espaços a serem retornados.

STRING$ preenche uma string com caracteres (F)

X$ = STRING$(*número, carac*)

Retorna uma string consistindo em um número especificado de um único caractere repetido.

número

Especifica o número de caracteres a serem repetidos.

carac

O caractere a ser repetido. *carac* pode ser uma constante, variável, ou o resultado de uma função de string.

SENTENÇAS DE ESTRUTURA

FOR...NEXT	**loop (faixa) de programa (S)**

FOR *var_índice* = *valor_inicial* TO *valor_final* [STEP passo]*sentença(s)*
NEXT [*var_índice*]

Executa sentenças dentro do loop delimitado por FOR e NEXT. As sentenças especificadas são executadas até que a variável de índice alcance o valor final.

var_índice

A variável de índice de loop (inteira).

valor_inicial, valor_final

Determinam os valores iniciais e finais do índice de loop. Podem ser constantes, variáveis, resultados de cálculos, ou de funções. O resultado será do tipo inteiro.

passo

O valor de índice é normalmente incrementado de um valor de +1. Isto pode ser modificado para qualquer passo acrescentando-se STEP à sentença. Se *passo* for negativo, FOR...NEXT conta para trás a partir do valor inicial.

sentença

Pode incluir qualquer um dos comandos e funções suportadas pelo GW-BASIC, e pode consistir em quantas linhas de programa forem necessárias.

NEXT [*var_índice*]

Designa o fim do loop. GW-BASIC reconhece todas as sentenças entre FOR e NEXT como sendo parte deste loop.

A variável de índice precisa ser especificada somente em loops FOR...NEXT encadeados.

Os loops FOR...NEXT não podem ser aninhados a menos que seus elementos estejam na ordem apropriada. O FOR mais interno deve ser concluído com um NEXT antes dos outros loops, seguido de um NEXT correspondente ao segundo loop definido mais recentemente etc. Se isto não for feito corretamente, GW-BASIC retornará a mensagem de erro *NEXT without FOR*. O valor de índice de loop pode ser modificado dentro do loop. Entretanto, isto poderia causar um loop infinito. O loop pode ser cancelado a qualquer hora com GOTO. Para a sua segurança, indique a variável de índice ao valor final de modo que a sentença NEXT finalize apropriadamente o loop. Entretanto, isto poderá resultar na mensagem de erro *Out of memory*.

GOSUB...RETURN — sub-rotina (S)

GOSUB *número_de_linha* RETURN [*número_de_linha*]

Chama e executa uma sub-rotina, e retorna à sentença logo a seguir do GOSUB.

número_de_linha

A linha de programa na qual a sub-rotina começa, ou a linha para onde saltar quando da finalização da sub-rotina.

Uma sub-rotina chamada com GOSUB também pode ser finalizada com GOTO.

Entretanto, isto pode resultar na mensagem de erro *Out of memory*.

GOTO — salto incondicional (S)

GOTO *número_de_linha*

Salta incondicionalmente dentro do programa.

número_de_linha

O número de linha no programa para onde será dado o salto.

Loops de programa (FOR...NEXT, WHILE...WEND) e sub-rotinas chamadas por GOSUB não devem ser finalizadas usando-se GOTO, porque os dados armazenados na pilha não são removidos apropriadamente.

IF...THEN...ELSE — salto condicional (S)

IF *condição* THEN *sentença_1* [ELSE *sentença_2*]

Salta no programa ou executa sentenças baseado nos resultados de um condição ou série de condições.

condição

Uma condição ou comparação suportada pelo GW-BASIC.

sentença_1, sentença_2

Quaisquer comandos ou funções suportadas pelo GW-BASIC podem ser usadas aqui. Múltiplas sentenças podem ser separadas por dois pontos, mas o tamanho total não pode exceder uma linha de programa.

ELSE é opcional. Se ELSE for omitido e a condição não for preenchida, a execução continuará com a próxima linha. Se ELSE for especificado, ele deve estar na mesma linha que IF e THEN. Para

saltos absolutos com GOTO, a sentença GOTO não precisa ser especificada logo a seguir de THEN e ELSE. GOSUB, por outro lado, deve sempre ser especificado.

ON X GOTO/GOSUB **salto condicional (S)**

ON *índice* GOTO *lista_de_números_de_linhas*

(ou) ON *índice* GOSUB *lista_de_números_de_linhas*

Salta condicionalmente para subprogramas ou sub-rotinas através de índice.

índice

Uma variável numérica, função, ou cálculo cujo resultado é do tipo inteiro.

lista_de_números_de_linhas

Uma lista de números de linhas (separados por vírgulas) para os quais a execução saltará como sendo o índice.

Se X=0 ou X é maior do que o número de números de linhas especificado, o programa continuará na linha logo a seguir da linha ON X GOTO/GOSUB. Os números de linha não precisam estar em ordem crescente. Uma linha pode aparecer na lista mais de uma vez.

WHILE...WEND **loop de programa (S)**

WHILE *condição sentença*

WEND

Executa sentenças enquanto uma condição for verdadeira.

condição

Uma expressão lógica (comparação de dados etc.) suportada pelo GW-BASIC.

sentença

Todos os comandos e funções suportados pelo GW-BASIC podem ser usados aqui. *sentença* pode estender-se sobre múltiplas linhas de programa.

Observe que o loop será executado enquanto a condição for verdadeira. Um loop infinito ocorrerá caso a condição nunca seja verdadeira devido a um erro ou a dados incorretos.

Múltiplas condições agrupadas com AND ou OR podem ser especificadas, uma das quais pode provocar um "final forçado" para o loop. Este pode ser finalizado a qualquer hora com GOTO. Entretanto, isto pode resultar na mensagem de erro Out of memory.

MENSAGENS DE ERRO

Os números de erros e de mensagens do GW-BASIC são divididos nos seguintes grupos:

Erro 01 a erro 30	Erros de programa e de sintaxe
Erro 50 a erro 76	Gerenciamento de arquivos e periféricos

Os seguintes números de erro geram Unprintable error: 21, 28, 56, 59, 60, 65, e 78 a 255.

Os seguintes números de erro não são usados e exibem mensagens variadas, dependendo da versão de GW-BASIC:

Erro 73	geralmente Função Avançada
Erro 77	geralmente Deadlock (Travamento do sistema)

As seguintes duas mensagens de erro não possuem números de erro:

Can't continue after SHELL

Esta mensagem é enviada para a área de programa ou de dados se o GW-BASIC for destruído depois que um programa SHELL tiver sido executado. O PC responderá com o prompt do MS-DOS.

Can't run BASIC as a child from BASIC

Esta mensagem é produzida por algumas versões se você tentar chamar o interpretador novamente com SHELL.

01 NEXT without FOR

Durante a execução de programa foi descoberto um NEXT não precedido por um FOR. Este erro ocorre geralmente em loops FOR...NEXT aninhados do qual FOR foi apagado.

02 Sintax error

O interpretador não consegue identificar a instrução que ele deve executar. Você ou entrou com a palavra-chave incorretamente, ou com um parâmetro ilegal, esqueceu caracteres especiais como parênteses, vírgulas, ou ponto-e-vírgulas, ou entrou com caracteres demais em uma linha de programa.

03 RETURN without GOSUB

O interpretador encontrou um RETURN sem primeiro ter executado um GOSUB. Este erro ocorre geralmente em seqüências encadeadas GOSUB/RETURN.

04 Out of DATA

READ tentou ler mais dados do que estavam disponíveis nas sentenças DATA. Verifique essas linhas. RESTORE pode ter sido especificado com uma linha incorreta.

05 Illegal function call

Este erro possui uma grande variedade de causas:

* O número de registro do GET#/PUT# para um arquivo de acesso randômico é 0 ou negativo.
* O comando USR sem um DEF USR anterior.
* O valor de uma operação matemática foi inválido.
* Um parâmetro inválido foi passado para uma função.
* Você tentou indexar uma matriz com um número negativo.

06 Overflow

O resultado de um cálculo é muito grande. Verifique os argumentos.

07 Out of memory

O programa é muito grande, portanto não há mais memória suficiente para as variáveis e para a pilha, ou as instruções podem ter sido encadeadas muito profundamente e/ou são muito complexas, significando que você tem muitos FOR / NEXTs, WHILE / WENDs, e/ou GOSUB / RETURNs abertos de uma só vez. Tenha a certeza de que as sub-rotinas não tenham sido finalizadas usando-se GOTO, que os loops FOR...NEXT e WHILE...WEND estejam finalizados, e que os cálculos e expressões sejam tão simples quanto possíveis. Libere a memória regularmente com FRE(" ") quando você usar diversas operações de strings de uma só vez.

08 Undefined line number

Você está tentando fazer referência a um número de linha não existente em uma sentença tal como GOTO ou

GOSUB. Verifique o número de linha que está sendo chamado.

09 Subscript out of range

Você tentou endereçar um elemento de matriz que está ou além da dimensão indicada com DIM, ou menor do que a área inferior de índice indicada por OPTION BASE. Verifique os índices.

10 Duplicate definition

Uma matriz foi redefinida com DIM, ou uma matriz sendo usada com o dimensionamento padrão foi redimensionada com DIM. Também é possível que OPTION BASE tenha sido usado após as matrizes terem sido referenciadas. Para redimensionar uma matriz, ela deve ser primeiramente apagada com ERASE. OPTION BASE deve ser usado no início do programa antes de todos os DIMs.

11 Division by zero

Você tentou dividir um valor por 0. Verifique o valor antes da divisão para ver se ele é zero.

12 Illegal direct

Você tentou executar um comando em modo direto que somente pode ser permitido no modo de programa. Use o comando somente dentro de um programa.

13 Type mismatch

Os tipos de variáveis usados em uma operação de string ou operação matemática não batem. Verifique os tipos de variável.

14 Out of string space

A memória de string irá ser excedida se muitas operações de string e liberação de memória insuficientes forem usadas. Mantenha as operações de string para um mínimo, e execute liberação de memória geralmente com FRE(" ").

15 String too long

Uma string maior do que 255 caracteres foi criada. Verifique a operação e os conteúdos das variáveis usadas.

16 String formula too complex

A operação de string que você tentou executar é complexa demais para o interpretador. Divida a operação em diversas operações menores.

17 Can't continue

Você usou um CONT sem um STOP anterior, ou modificou o programa ou os conteúdos de variáveis após um STOP.

18 Undefined user function

Uma função foi chamada dentro do programa com FN ou USR que não foi definida com DEF FN ou DEF USR.

19 No RESUME

Depois de um ERROR GOTO, o interpretador descobriu que a sua rotina de tratamento de erro não continha um RESUME.

20 RESUME without ERROR

Um RESUME foi encontrado dentro do programa mas um erro não aconteceu. Verifique a rotina.

22 Missing operand

Um operando dentro de uma operação está faltando.

23 Line buffer overflow

Você digitou uma linha em um modo direto com mais de 255 caracteres. Encurte a linha ou escreva um programa para executar a linha.

24 Device timeout

A resposta esperada não foi recebida de um dispositivo periférico dentro do tempo especificado. Veja se o dispositivo está presente e ligue-o. Para operações COM você pode especificar intervalos de tempo maiores para CS tempo, DS tempo e CD tempo. Você também pode enganar o erro com ON ERROR GOTO e IF ERR=24 THEN RESUME. Isto gera um loop infinito na pior das hipóteses.

25 Device fault

O dispositivo endereçado não existe. Verifique a chamada.

26 FOR without NEXT

Não há nenhum NEXT para que o FOR seja executado, ou os loops não estão aninhados corretamente.

27 Out of paper

A impressora enviou um código de erro "sem papel".

29 WHILE without WEND

Um WHILE ocorreu no programa sem um WEND correspondente.

30 WEND without WHILE

Um WEND foi encontrado sem que um WHILE tenha sido executado.

50 FIELD overflow

Você tentou fazer o campo maior do que o tamanho de registro especificado em OPEN. Verifique os tamanhos especificados de variáveis ou o parâmetro LEN=*tamanho_de_registro*.

51 Internal error

Um erro ocorreu dentro do interpretador provocado por uma falha do interpretador.

52 Bad file number

Você tentou acessar um arquivo que não foi aberto com OPEN, ou o número do arquivo cai fora da faixa de 1 a 15, ou o número máximo de arquivos abertos já foi atingido. Verifique a sua entrada ou o parâmetro /F ao chamar GW-BASIC ou a entrada no CONFIG.SYS.

53 File not found

O arquivo endereçado não pode ser encontrado. Verifique o drive, caminho, nome do arquivo e extensão.

54 Bade file mode

Você tentou usar um arquivo de uma maneira que não corresponde ao seu modo. Em outras palavras, você tentou usar PUT# ou GET# em um arquivo seqüencial ou tentou ler algo da impressora.

55 File already open

Você tentou usar um número de arquivo que já está em uso em outro comando OPEN. Escolha um número de arquivo diferente ou feche o outro arquivo. O mesmo

erro será exibido caso você tente dar um KILL em um arquivo aberto ou modificar o seu nome com NAME. Nestes casos o arquivo deve ser fechado primeiro.

57 Device I/O error

Um erro ocorreu durante a saída para o disco ou impressora. É possível que o disquete não esteja formatado, esteja protegido contra gravação, ou não possa ser gravado por algum outro motivo. Se este erro ocorrer em um disco rígido, pode haver um problema com o próprio disco. Para uma impressora, ou o dispositivo de impressão não está ligado ou não está em linha, ou enviou um código de erro que não pode ser interpretado. Verifique o dispositivo de impressão.

58 File already exists

Quando você modificou o nome de um arquivo com NAME, você deu a especificação de um arquivo existente em *novo_nome_de_arquivo*. Escolha um nome diferente.

61 Disk full

O seu disco está cheio. Insira um novo disco ou limpe o espaço em disco.

62 Input past end

Você tentou ler após o fim de um arquivo seqüencial. Use EOF para verificar o final do arquivo.

63 Bad record number

O número de registro especificado é 0, negativo, ou maior do que o número máximo de registro (16,777,215).

64 Bad file name

O nome de arquivo usado possui caracteres inválidos ou possui caracteres demais. Veja o manual do MS-DOS para caracteres válidos, ou encurte o nome.

66 Direct statement in file

Uma sentença sem um número de linha foi descoberta enquanto se carregava um arquivo de programa. O processo de carga foi finalizado. Verifique o programa e insira números de linha conforme necessário.

67 Too many files

Não há espaço suficiente no diretório no disco para armazenar a entrada do arquivo. Use um disquete ou subdiretório diferente.

68 Device unavailable

Foi feita uma tentativa de acessar um dispositivo não existente com OPEN.

69 Communication buffer overflow

O buffer de recepção para a interface COM excedeu a sua capacidade. Ou indique um buffer maior com /C: buffer quando chamar o GW-BASIC, selecione uma velocidade mais baixa caso os dados não possam ser processados suficientemente rápido, ou use ON ERROR GOTO.

70 Disk write protected

Foi descoberto durante a saída do arquivo que o disquete está protegido contra a gravação. Remova o selo de proteção contra a gravação, ou use outro disquete.

72 Disk media error

A controladora de disco descobriu irregularidades durante a transferência de dados. Pode haver diversas razões para isso:

* Existe algo fisicamente errado com o disquete.
* A cabeça de leitura/gravação não pode ser posicionada corretamente.
* O disquete contém setores defeituosos.
* A cabeça de leitura/gravação está suja.

Verifique o disco com CHKDSK.COM, e copie os arquivos para um outro disco. Se você não conseguir encontrar o motivo do problema, verifique com a manutenção se é possível fazer uma checagem no(s) drive(s).

74 Rename across disks

Você especificou drives diferentes nas especificações do arquivo a ser renomeado.

75 Path/file acess error

A especificação dada de caminho/arquivo contém um erro.

76 Path not found

O caminho especificado para o acesso a um arquivo não está correto.

ÍNDICE ANALÍTICO

ÍNDICE DE COMANDOS

Este índice lista todos os comandos do GW – BASIC em ordem alfabética, para referência rápida:

ÍNDICE REMISSIVO

Este índice agrupa os comandos do GW-BASIC conforme os seus efeitos. Use-o para encontrar rapidamente a palavra-chave para a operação específica que você deseja executar:

Comparação de Dados

Combinar comparações	AND	9
Combinar comparações	OR	9
Combinar dados quanto à igualdade	=	7
Comparar dados quanto à desigualdade	<>	7
Comparar dados quanto à desigualdade	<	8
Comparar dados quanto à desigualdade	>	8
Inverter o resultado da comparação	NOT	9

Comunicações de E/S

Abrir canal COM	OPEN COM	65
Esperar até que a porta mude	WAIT	69
Enviar dados ao buffer COM	PUT#	69
Funções de buffer	LOC / LOF	65
Gravar valor em uma porta	OUT	68
Indicar tamanho da linha para COM	WIDTH	70
Ler dados do buffer COM	GET#	64
Ler valor da porta atual	INP	64

Constantes e variáveis

Apagar o dimensionamento	ERASE	4
Definir tipo de dados global	DEFDBL	3
Definir tipo de dados global	DEFINT	3

Ler timer durante a interrupção	ON TIMER	76
Verificar buffer PLAY	ON PLAY	75

Tratamento de Erros

Continuar programa após erro	RESUME	30
Enganar o erro	ON ERROR GOTO	29
Gerar erro	ERROR	29
Indicar dispositivo causador do erro	ERDEV$	28
Indicar número da linha incorreta	ERL	28
Indicar número do código do erro	ERR	29
Indicar número do erro	ERDEV	28

Tratamento de Strings

Buscar por caractere na string	INSTR	120
Concatenar strings	+	117
Data do sistema/hora do sistema	DATE$ / TIME$	118
Definir/chamar funções de strings	DEF FN / FN	119
Determinar tamanho de uma string	LEN	121
Seccionar string (esquerda)	LEFT$	121
Seccionar string (direita)	RIGHT$	123
Seccionar string	MID$	121
Limpar armazenamento de string	FRE(" ")	120

CADASTRO PARA MALA DIRETA

Favor preencher todos os campos

NOME (não abreviar):

END. RESIDENCIAL:

CIDADE: UF: CEP: FONE:

EMPRESA:

SEXO: F( ) M( ) NASCIMENTO:____/____ MÊS ANO

1. Cargo: ( )1. Digitador ( )2. Programador ( )3. Analista ( )4. Supervisor ( )5. Gerente ( )6. Diretor ( )7. Outros (especificar):_____

2. Escolaridade: ( )1. 1º grau ( )2. 2º grau ( )3. 3º grau

3. Eu geralmente, compro livros: ( )1. Em livrarias ( )2. Por telefone ( )3. Em feiras ( )4. Por Reembolso Postal ( )5. Outros (especificar):_____

4. Costumo comprar em média, a seguinte quantidade de livros por ano: ( )1. 01 ( )2. 03 ( )3. 05 ( )4. mais

5. Estou interessado em livros das seguintes áreas:
( ) 1. Informática ( ) 2. Negócios ( ) 3. Controle Qualidade
( ) 4. Informática na Educação ( ) 5. Ensino Inglês ( ) 6. Psicologia
( ) 7. Sociologia ( ) 8. Metodologia ( ) 9. Medicina
( ) 10. Eng. Civil ( ) 11. Eng. Elétrica/ Eletrônica ( ) 12. Eng. Mecânica
( ) 13. Eng. Química ( ) 14. Física ( ) 15. Matemática
( ) 16. Outras (especificar):_____

6. Utiliza microcomputador: ( )Sim ( )Não

7. Qual? ( )1. PC ( )2. Apple ( )3. MSX ( )4. Outros (especificar):_____

8. Softwares mais utilizados: ( )1. Banco de Dados ( )2. Planilha ( )3. Proc. Texto ( )4. Outros (especificar):_____

9. Linguagens: ( )1. C ( )2. Cobol ( )3. Clipper/dBase ( )4. Assembler ( )5. Outras (especificar):_____

Obra: GW – Basic – Guia de Referência Básica
Autor: H. F. Bomanns

MAKRON *Books*

MAKRON *Books* do Brasil Editora Ltda.
Nova denominação da
Editora McGRAW-HILL Ltda.
"Padrão em Livros de Qualidade"
Rua Tabapuã, 1105 – Itaim Bibi – SP
04533 – Tels.: 820.6622/820.8528

McGraw Hill

OUTROS LIVROS NA AREA

Meirelles — *Informática: Novas Aplicações com Microcomputadores*

Weber — *Introdução ao Basic para o IBM PC*

Mania de Cultura
8,00
0-07-460534-8