ENCICLOPÉDIA DA LINGUAGEM BASIC

Comandos, Instruções e Funções dos Equipamentos das Diversas Linhas

CLÓVIS PEREIRA
ROSSANA B. ALCANTARA

EDITORA CAMPUS

TÍTULOS DE INTERESSE CORRELATO

ADA®: UMA INTRODUÇÃO – *H. Ledgard*
ALÉM DO BASIC – *N. Santos*
BASIC BÁSICO, 5ª ed. – *J. C. Pereira Filho*
BASIC COM ESTILO – *P. Nagin e H. F. Ledgard*
BASIC PARA APLICAÇÕES COMERCIAIS – *D. Hergert*
BASIC PARA MICROS PESSOAIS, 2ª ed. – *J. C. Pereira Filho*
BASIC RÁPIDO: ALÉM DO BASIC TRS-80 – *G. A. Gratzer e T. G. Gratzer*
BASIC SINCLAIR – *R. U. Christmann*
COBOL COM ESTILO – *L. J. Chmura e H. F. Ledgard*
COBOL PARA ESTUDANTES, 4ª ed. – *A. Parkin*
COBOL: REGRAS PARA PROGRAMADORES – *G. Ledin Jr. et al.*
CONCEITOS DE LINGUAGENS DE PROGRAMAÇÃO – *C. Ghezzi e M. Jazayeri*
FORTRAN PARA MICROS – *G. Marshall*
GUIA DE LINGUAGENS DE COMPUTADORES – *H. L. Helms Jr.*
INTRODUÇÃO À PROGRAMAÇÃO FORTRAN – *J. C. Pereira Filho*
JCL SISTEMA/370, 4ª ed. – *G. D. Brown*
LINGUAGEM DE PROGRAMAÇÃO ALGOL, 2ª ed. – *L. M. Segre*
LINGUAGEM PASCAL, 2ª ed. – *V. L. Strube de Lima*
LOGO – *P. Goodyear*
MANUAL DE COBOL ESTRUTURADO – *D. D. McCracken*
MANUAL DE LINGUAGEM C – *L. Hancock e M. Krieger*
PROGRAMAÇÃO EM ASSEMBLER E LINGUAGEM DE MÁQUINA, 2ª ed. – *D. C. Alexander*
PROGRAMAÇÃO EM BASIC – *J. G. Kemeny e T. Kurtz*
RPG II – *J. C. Pereira Filho*

Conheça toda a linha de Informática da Editora Campus, com títulos nas áreas de: Introdução à Computação; Teoria, Organização e Processamento de Dados; Linguagens; Programação; Programas e Aplicativos; Sistemas Operacionais e Compiladores; Arquitetura e Equipamentos; Interesse Especial.

ENCICLOPÉDIA DA LINGUAGEM BASIC

ENCICLOPÉDIA DA LINGUAGEM BASIC

Comandos, Instruções e Funções dos Equipamentos das Diversas Linhas

CLÓVIS PEREIRA

Bacharel em Matemática
Analista de Sistemas
Diretor Executivo do Instituto Sullivan
Diretor-Geral da Sullivan Informática e Tecnologia

ROSSANA B. ALCANTARA

Bacharel em Ciências Sociais
Programadora em BASIC e COBOL

EDITORA CAMPUS LTDA.
Rio de Janeiro

Capa
Otavio Studart

Paginação e revisão
Editora Campus Ltda.
Rua Barão de Itapagipe 55 Rio Comprido
Tel.: (021) 284 8443
20261 Rio de Janeiro RJ Brasil
Endereço telegráfico: CAMPUSRIO

ISBN 85-7001-235-7

Ficha Catalográfica
CIP-Brasil. Catalogação-na-fonte
Sindicato Nacional dos Editores de Livros, RJ.

P49e

Pereira, Clóvis.
Enciclopédia da linguagem BASIC : comandos, instruções e funções dos equipamentos das diversas linhas / Clóvis Pereira e Rossana Benvenuti Alcantara. — Rio de Janeiro : Campus, 1985.

ISBN 85-7001-235-7

1. BASIC (Linguagem de programação para computadores) I. Alcantara, Rossana Benvenuti II. Título

85-0156 CDD — 001.6424

Introdução

0.1 O MICROCOMPUTADOR

O Microcomputador, definido nos termos mais simples, é uma "máquina" que realiza, sob comando, cálculos e operações lógicas. De acordo com essa definição, no entanto, até a pessoa que acione um calculador pessoal poderia ser considerada como se fosse um computador.

O computador, seja ele pequeno ou muito complexo, deve, na realidade, ter capacidade de realizar mais três funções além da de calcular, isto é, deve poder conter ou armazenar os dados com os quais esta efetuando os cálculos. Deve poder decidir quais os cálculos a efetuar em seguida, e deve ter meios de comunicação com o mundo externo, em primeiro lugar para receber os dados sobre o problema e, em seguida, para transmitir as respostas ou os resultados. A primeira dessas funções e realizada por "memórias" que consistem primordialmente dé semicondutores de óxidos metálicos, suplementadas geralmente por dispositivos de maior capacidade tais como discos e fitas magneticas.

A última função se denomina freqüentemente de "entrada/saída" (E/S), sendo levada a cabo por elementos tais como discos (disquetes), fitas magneticas, impressoras e diversos terminais para interação pessoal.

0.2 O PROGRAMA

Em virtude da elevada velocidade do computador, é preciso que lhe seja proporcionado de antemão um plano de realização dos cálculos, inclusive no que diz respeito à maneira como deve checar um determinado numero de decisões simples que têm de ser tomadas.

Torna-se, portanto, necessário fornecer ao computador de antemão um conjunto completo de instruções quanto aos cálculos a serem efetuados e às decisões lógicas a serem tomadas. Esse conjunto de instruções se denomina programa. Dito de outra maneira, o programa é um plano ou uma "receita" que faculte passar dos dados do problema para as respostas desejadas.

Uma característica importante de um programa é ter que ser preparado numa linguagem que seja "compreensível" ao mecanismo do computador. Em geral as regras de gramática exigidas por essas linguagens são muito detalhadas, devendo ser seguidas com exatidão. A tarefa de elaborar, sob forma precisa, os programas destinados à realização de tarefas pelo computador, se denomina programação, e os praticantes dessa especialidade se chamam programadores. O programador mais bem sucedido é aquele que sabe adotar critérios sensatos a respeito de abordagens alternativas de determinado problema, selecionando métodos que sejam eficientes, em termos tanto das pessoas que vão utilizar o programa como da utilização dos recursos do computador.

Sendo de tal modo especializada a programação em linguagem de máquina, as pessoas mais familiarizadas com outros setores acharam difícil anteriormente traduzir seus problemas para fins de equacionamento pelo computador. Daí que se elaboraram diversas linguagens tais como: COBOL, FORTRAN, BASIC, linguagens essas cujas regras de gramática são mais fáceis de

aprender e aplicar do que as da linguagem de máquina. A utilização dessas linguagens, que são orientadas mais em termos dos usuários dos computadores, veio fomentar enormemente a aplicação dos computadores e da tecnologia da computação para muitas áreas do comércio, da engenharia e das pesquisas acadêmicas.

0.3 A LINGUAGEM "BASIC"

Embora o advento de linguagens orientadas em termos de usuário tenha sido de grande benefício para o uso do computador, a maioria dessas linguagens foram idealizadas ou para a utilização especial de certos grupos de peritos, ou numa época em que relativamente pouco se sabia a respeito de sua tradução. Daí que a maioria dessas linguagens seja difícil de aprender e, em seguida, um tanto difícil de aplicar. Diante da percepção de que muitos usuários de computadores em potencial poderiam achar as próprias linguagens orientadas em termos de usuário uma barreira para o uso inteligente dos computadores, ficou decidido que havia necessidade de uma linguagem de feição simples,que deveria possuir regras gramaticais bastante singelas, podendo ser aprendida em prazo muito reduzido. Foi assim que apareceu a linguagem "BASIC". BASIC é uma sigla, que representa "Beginner's All-Purpose Symbolic Instruction Code". (Código Simbólico de Instruções para todos os fins). Foi desenvolvida pelos professores John Kemeney e Thomas Kurtz no Dartmouth College na década de 60. BASIC foi elaborada na qualidade de linguagem de instrução, pelo que é relativamente simples e fácil de utilizar. E uma linguagem interativa, podendo ser utilizada na área comercial, científica, educacional e de lazer. Embora seja fácil de aprender, pode ser aplicada com rapidez e facilidade para a maioria dos problemas de computação (tanto os não-numéricos como os numéricos). A finalidade desta publicação é, portanto, dar ao leitor os conhecimentos necessários para que possa **aprender a programar** os MICROCOMPUTADORES, mediante a utilização da linguagem "BASIC", linguagem esta adotada para todos os MICROS existentes, e em seguida desenvolver uma ampla gama de aplicações.

Convém saber que existem e se encontram em uso generalizado muitos dialetos e versões da linguagem "BASIC". As partes elementares da maioria das versões são idênticas ou bastante semelhantes tanto à versão utilizada aqui como à Norma Americana Nacional para "BASIC" em termos Mínimos. Diferem mais radicalmente as partes mais adiantadas dos diversos dialetos de BASIC. A maioria deles proporciona, contudo, capacidades gerais muito semelhantes às utilizadas nesta publicação.

0.4 COMO SE UTILIZA O MICROCOMPUTADOR

Deve-se fornecer ao microcomputador um programa e um conjunto de dados. Isto se faz, via de regra, através de um teclado. (Existem tantos tipos de teclado que seria inútil tratar de descrevê-los todos. A característica que compartilham entre eles é possuírem teclado semelhante ao da máquina de escrever). Dá-se também entrada geralmente aos dados através de um teclado. Uma vez "dentro" do computador, o programa e os dados são armazenados em discos ou em fitas cassetes ou em algum tipo de memória externa, não tendo na realidade conseqüências substanciais para o usuário. O fato importante é que o programa pode ser chamado e posto em operação pelo usuário com pouco

esforço adicional.

Os sistemas de microcomputadores que "falam" BASIC variam desde os mais pequenos, de uso pessoal e de baixo custo, até as unidades de porte maior, de uso profissional e de altos custos. Nas máquinas de menor tamanho, os programas são armazenados muitas vezes em fitas cassetes comuns de boa qualidade, ou em disquetes. Nas demais máquinas, os programas e dados do usuário ficam armazenados no sistema do computador onde, uma vez lançados, permanecem até não mais haver necessidade deles. Em muitas máquinas, por outro lado, uma série de usuários podem compartilhar tempo; em outras palavras, cada programa executa uma pequena parte da computação, passando a ceder lugar em seguida ao próximo programa. Os programas não terminados na primeira vez voltarão posteriormente para computação adicional.

A maneira **como** os sistemas de computação encontram erros nos programas BASIC, e **como** processam esses programas apresenta variações de caso para caso. Alguns sistemas detectam os erros ao se teclar cada linha, ao passo que outros esperam até que se tenha dado entrada em todo o programa, e quando executado os erros serão detectados pelo interpretador BASIC. Alguns sistemas de BASIC fazem uso de interpretadores, em que cada instrução BASIC é traduzida e executada (realizada) cada vez que se encontre no decorrer do processamento do programa. Outros sistemas BASIC utilizam compiladores, que traduzem o programa inteiro de BASIC para a própria linguagem de máquina interna de seu computador, após o que a versão traduzida é executada. Ninguém, a não ser o usuário com muito discernimento, poderá detectar a diferença entre o sistema BASIC equipado com compilador e aquele que possui interpretador, uma vez que, nas demais características, os dois sistemas são idênticos.

Ilustram-se nos capítulos seguintes diversos COMANDOS, INSTRUÇÕES, FUNÇÕES e OPERADORES elementares na linguagem BASIC.

Os comandos se diferenciam das instruções pelo fato de que aqueles dizem ao computador o que fazer com o programa em si, ao passo que estas, por fazerem parte do programa em si, indicam ao computador quais os cálculos e as demais operações a realizar.

A diferença principal entre um COMANDO e uma INSTRUÇÃO reside nas suas execuções. Um COMANDO ordena que o computador execute imediatamente o que foi pedido, enquanto que uma INSTRUÇÃO necessita de uma outra ação para que o BASIC a execute. Consideramos COMANDO toda ordem dada ao computador no MODO DIRETO ou IMEDIATO e consideramos INSTRUÇÃO toda ordem dada ao computador no MODO PROGRAMA ou MODO INDIRETO. A INSTRUÇÃO é muitas vezes considerada como um COMANDO INDIRETO.

```
EXEMPLO 1:          EXEMPLO 2:
PRINT 51            10 PRINT 51
```

No EXEMPLO 1, PRINT é um comando e será executado assim que apertar a tecla de entrada (RETURN, CR, ENTER, NEW LINE ou equivalente).

No EXEMPLO 2, PRINT é uma INSTRUÇÃO e somente será executada quando o programa for rodado (RUN).

Todo PROGRAMA obrigatoriamente após o número da Linha terá que ter uma INSTRUÇÃO.

FUNÇÕES são pré-programas já gravados na ROM e que quando executadas obedecem os procedimentos necessários para atingir o objetivo. Existem certas PALAVRAS RESERVADAS usadas em BASIC para chamar as FUNÇÕES, algumas que lidam com dados numéricos e outras

com "STRINGS" (valores alfanuméricos).

As funções devem vir sempre precedidas de uma INSTRUÇÃO, geralmente LET ou PRINT.

```
EXEMPLO 1:      10 X = 100
                20 PRINT SQR(X)
```

A FUNÇÃO SQR(X) é usada para produzir a raiz quadrada do valor de X.

```
EXEMPLO 2:      10 X$ = "SULLIVAN"
                20 PRINT LEFT$(X$,3)
```

A FUNÇÃO LEFT$ quando for executada irá caracterizar o valor alfanumérico de X$ da esquerda para a direita; apenas as 3 primeiras letras e a linha 20 imprimirá SUL.

Os COMANDOS, INSTRUÇÕES e as FUNÇÕES são representados no BASIC por PALAVRAS-CHAVE, reconhecidas pelo INTERPRETADOR.

Os OPERADORES são representados por certos caracteres especiais pelo INTERPRETADOR BASIC e cada categoria de OPERADORES define um tipo de EXPRESSÃO. Existem expressões aritméticas, expressões alfanuméricas, expressões relacionais e expressões Booleanas.

Convenções de nomenclatura e formato adotadas

Adotamos um padrão para apresentar a sintaxe geral, comando, instrução, função, declaração ou operador.
Apresentamos as pontuações e outras convenções mecânicas que usamos.

{} Chaves indicam uma escolha de itens. Um dos itens do grupo deve estar presente; as chaves aparecem em uma declaração real.

[] Colchetes indicam que o parâmetro interno e opcional; os colchetes não aparecem em uma declaração real.

NÚMERO DE LINHA Um número de linha inicial é sempre aplicado para declaração em modo programado.

OUTRAS PONTUAÇÕES Todas as outras marcas de pontuação - vírgulas, dois pontos, aspas e parênteses - devem aparecer como mostrado.

MAIÚSCULAS As palavras em maiúscula devem aparecer exatamente como mostrado.

MINÚSCULAS Termos genéricos em minúsculo. O programador informará a palavra ou valor exato de acordo com a forma abaixo:
Os termos genéricos minúsculos são usados em definição de declarações e funções.
Qualquer termo minúsculo apresentado aqui é particular àquela função em que aparece, e lá é descrito.

col Número de coluna para gráfico de baixa resolução: uma expressão numérica.

colh Número de coluna de gráfico de alta resolução; uma expressão numérica.

const Qualquer constante numérica ou alfanumérica.

Dn Um número de drive de disco deve ser especificado com D1 ou D2 para linha APPLE II ou D0 ou D1 para linha TRS-80.

expr Uma expressão numérica, constante relacional ou Booleana (apenas em Applesoft), variável ou expressão; qualquer combinação destas.

expr$ Qualquer constante alfanumérica variável ou expressão.

exprnm Qualquer constante numérica, variável ou expressão.

nomearq Qualquer nome do arquivo.

linha Qualquer número de linha de programa BASIC.

linha; Um dos vários números de linha do programa BASIC.

endmem Uma expressão numérica, variável ou constante que se relaciona a um endereço de memória. Os endereços de memória variam de -32767 a 32767, ou em Applesoft, de 0 a 65535, onde é equivalente a 1,-65534 a 2 etc.

posmem Uma posição de memória especificada por um inteiro constante entre 0 e 65535 (decimal) ou $0 a $FFF (hexadecimal). As constantes hexadecimais são identificadas pela sinal de dólar ($) à frente.

mensagem Qualquer alfanumérico de texto entre aspas.

linha Número de linha em gráfico de baixa resolução; uma expressão numérica com valor entre 0 e 47.

linhah Número de linha em gráfico de alta resolução; uma expressão numérica com valor entre 0 e 191.

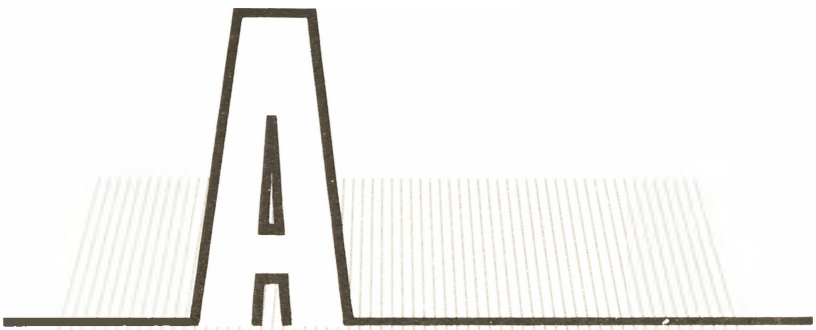
Sn Número do slot para entrada e saída; deve ser S0,S1,S2,S3,S4,S5,S6 ou S7, para computadores que seguem a linha APPLE II.

var Uma variável numérica ou alfanumérica. Em Applesoft do APPLE, qualquer variável numérica, inteira ou cordão.

varnm O nome de uma variável numérica.

var(sub) O nome de uma variável numérica subscrita. Em Applesoft, qualquer variável inteira, numérica ou alfanumérica.

Vn Qualquer número de volume para identificação de disco (entre V0 e V255), para linha APPLE.



ABS • A. • ACS • AC. • ACSD
ACSG • ARCOS • ARCCOS
AND • A • APPEND • ASC
ASCII • ASN • ASNG • ASND
ARCSIN • AT • A. • ATN • ATAN
ATND • ATNG • ARCTAN
ATTRIB • ATRIB • AUTO

Função **ABS • A.**

SINTAXE GERAL

ABS(argumento numérico)

ABS é utilizado apenas como função, devendo ser precedido de uma instrução, geralmente PRINT ou LET.
Esta função é reconhecida pela maioria dos interpretadores BASIC e válida em INTERGER e FLOATING POINT-BASIC.
ABS fornece o valor absoluto ou módulo do argumento. O argumento deve ser um valor numérico dentro do parênteses com o sinal (+) positivo ou (-) negativo. O valor contido no argumento pode ser representado por uma constante numérica ou por uma expressão. Entretanto, o valor absoluto será sempre um valor sem o sinal + ou -.

EXEMPLO 1

```
100 REM USANDO A FUNCAO ABS
110 FOR X = -4 TO 4
120 PRINT ABS(X);
130 NEXT X
135 PRINT
140 PRINT "ESTES FORAM OS VALORES ABSOLUTOS DO ARGUMENTO X"
150 END
```

EXECUÇÃO DO PROGRAMA

```
RUN
4 3 2 1 0 1 2 3 4
ESTES FORAM OS VALORES ABSOLUTOS DO ARGUMENTO X
```

OBS.: ABS(X) fornece o valor absoluto de X, ou seja, o seu módulo.

NOTA:
Dentro das limitações do interpretador do computador a função ABS tem capacidade para lidar com qualquer número. Entretanto, na maioria dos interpretadores, ABS pode ser usado dentro de operações aritméticas. Deve estar contida entre parentêses toda operação matemática que segue ABS.

EXEMPLO 2

```
200 REM USANDO A FUNCAO ABS EM OPERACAO MATEMATICA
205 PRINT "O VALOR ABSOLUTO DE (4*(-5)) E' ";
210 PRINT ABS(4*(-5))
220 X = 20
230 Y = 40
240 PRINT "E O VALOR ABSOLUTO DE  ("X"-"Y")/2 E' " ABS((X-Y)/2)
250 END
```

EXECUÇÃO DO PROGRAMA

```
RUN
O VALOR ABSOLUTO DE (4*(-5)) E' 20
```

E O VALOR ABSOLUTO DE (20-40)/2 É' 10

OBS.: Em computadores que não tenham a função ABS, poderá ser facilmente criada a seguinte sub-rotina:

```
EXEMPLO 3
300 REM USANDO A FUNCAO ABS COMO SUBROTINA
310 PRINT "DIGITE UM NUMERO NEGATIVO ";
320 INPUT Z
325 IF Z > 0 GOTO 310
330 GOSUB 500
340 PRINT "O VALOR ABSOLUTO DE";Z;"E' ";U
350 END
500 REM * ABS(Z) SUBROTINA *
600 U = Z
700 IF Z>=0 THEN 900
800 U = -Z
900 RETURN
```

EXECUÇÃO DO PROGRAMA

```
RUN
DIGITE UM NUMERO NEGATIVO
? -9     (se digitar -9)
O VALOR ABSOLUTO DE -9 E' 9
```

ORTOGRAFIA ALTERNATIVA

Certos computadores aceitam A. como abreviatura de ABS.

VARIAÇÕES DE USO

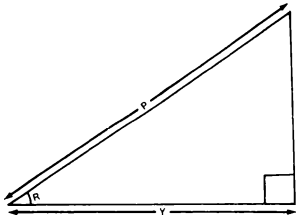
Não se tem conhecimento de nenhuma.

Função ACS • AC. • ACSD ACSG • ARCOS • ARCCOS

SINTAXE GERAL

Número de linha LET variável = ACS valor do co-seno (de -1 para 1)

ACS e uma das funções trigonométricas e geralmente trabalha em radianos, não em graus. Em termos matemáticos uma função é uma regra para dar um número (o resultado) em troca de outro (o argumento ou operando) e é realmente uma operação unitária.
A função ACS(S) é utilizada em determinadas linguagens BASIC para calcular o ARC(COS) da relação n em radianos (não em graus), sendo que um radiano equivale a aproximadamente 57 graus.



ArCcos (ACS) é definido como sendo o ângulo (R) do triângulo reto formado pela hipotenusa (de extensão P) e um dos lados (com extensão Y).

R = ACS(Y/HXP)

O contrário de ACS é COS (co-seno). O co-seno de um ângulo (que mede Y radianos) é a extensão do lado adjacente ao ângulo dividido pela hipotenusa do ângulo reto.

COS (R) = Y/P

EXEMPLO 1

```
100 REM USANDO ACS
110 PRINT "ENTRE COM O VALOR DO CO-SENO (DE -1 A 1)";
120 INPUT C
130 LET Y = ACS(C)
140 PRINT "O ANGULO COM A RELACAO Y/P DE "; C;" E DE "; Y; "
RADIANOS"
3999 END
```

EXECUÇÃO DO PROGRAMA

```
RUN
ENTRE COM O VALOR DO CO-SENO (DE -1 A 1)
? 1
O ANGULO COM A RELACAO Y/P DE 1 É 3.67372923E-06 RADIANOS
```

```
RUN
ENTRE COM O VALOR DO CO-SENO (DE -1 A 1)
? -1
O ANGULO COM A RELACAO Y/P DE -1 É 3.14159633 RADIANOS
```

Para fazer a conversão dos valores de radianos para graus, multiplique a medida do ângulo (em radianos) por 57.29578.
Existem certos computadores que calculam o ângulo em graus ou grados (100 grados = 90 graus). Esses computadores utilizam a função ACSD para graus e ACSG para grados. Substituindo-se na linha 130 e utilizando-se 0 na execução do programa, devem-se conseguir 90 graus e 100 grados.

ORTOGRAFIA ALTERNATIVA
Computadores que seguem a linha do SINCLAIR (Inglês), como o NE-8000, CP-200, TK-82C, TK-85, utilizam ARCCOS ao passo que o computador de bolso Sharp 1211(TRS-80) utiliza AC.

Se O SEU COMPUTADOR NÁO TIVER ESTA FUNÇÃO
Se o seu computador não aceita a linha 130 no EXEMPLO 1, mas reconhece ATN (ARCTANGENT) e SQR (RAIZ QUADRADA), substitua:

```
130 LET Y = 1.5708 - 2*ATN(C/(1+SQR(1-C*C)))
```

VARIAÇÕES DE USO
Não se tem conhecimento de nenhuma.

VEJA TAMBÉM
COS, ASN, ATN, SQR, SIN, TAN

Operador Lógico **AND • A**

SINTAXE GERAL

Número de linha IF expressão AND expressão...THEN...

AND é utilizado como operador lógico podendo fazer comparações lógicas entre valores numéricos ou entre valores alfanuméricos (Strings).
AND é um operador cuja sintaxe é a mesma na maioria dos equipamentos.
O operador AND fará com que o resultado de uma expressão seja verdadeiro somente se os dois operandos sobre os quais se age forem de valor também verdadeiro. Caso o valor de um dos operandos seja falso, o resultado será falso. Existe, porém, uma diferença entre o AND nos microcomputadores que seguem a família APPLE e o AND nos computadores que seguem as famílias TRS-80 ou SINCLAIR, quando esta instrução é utilizada numa expressão numérica. Suponha a operação:

```
C = A AND B
```

Na família APPLE, o resultado em C será igual a 1 se A e B forem diferentes de zero, e igual a zero se pelo menos um dos dois for igual a zero. No TRS-80 e SINCLAIR o resultado em C será igual ao menor dos dois operandos (em termos absolutos) se ambos forem diferentes de zero, e zero se um dos dois, ou ambos, forem iguais a zero.
Os operadores lógicos fazem comparações lógicas e são normalmente usados em instruções IF/THEN para fazer um teste lógico entre duas ou mais relações.
EX.

```
IF A = 1 AND C = 2 THEN PRINT A+C
```

EXEMPLO 1

```
100 REM USANDO AND OPERADOR LOGICO MATEMATICO
110 LET A = 5
120 PRINT "A = "A
130 PRINT "ENTRE COM UM VALOR PARA B "
140 INPUT B
150 PRINT "ENTRE COM UM VALOR PARA C "
160 INPUT C
170 IF A = B AND A = C THEN GOTO 200
180 PRINT "OS TRES VALORES NAO SAO TODOS IGUAIS, VEJA: "A;B;C
190 GOTO 210
200 PRINT "OS TRES VALORES A,B,C SAO IGUAIS "
210 END
```

EXECUÇÃO DO PROGRAMA

```
RUN

A = 5
ENTRE COM UM VALOR PARA B
```

```
? 5
ENTRE COM UM VALOR PARA C
? 5
OS TRES VALORES A,B,C SAO IGUAIS
```

O operador AND é utilizado por alguns microcomputadores para computar o AND lógico binário de dois números, utilizando-se a Álgebra Booleana.
AND é utilizado como porta ou circuito lógico elementar cuja saída só tem o valor lógico UM (1) quando todas as entradas (duas ou mais) também forem iguais ao UM (1) lógico.
Como se pode observar pela tabela abaixo, a saída do circuito AND só é 1 quando todas as entradas são 1 (o que em notação algébrica poderia ser representado por um produto).

ENTRADAS		SAÍDAS
X	Y	S
0	0	0
0	1	0
1	0	0
1	1	1

Como se pode observar pela tabela da verdade, a saída do circuito AND só é 1 quando todas as entradas são 1, o que em notação algébrica pode ser representado por um produto.

Se X=0 e Y=0 então S=0.0=0
Se X=0 e Y=1 então S=0.1=0
Se X=1 e Y=0 então S=1.0=0
Se X=1 e Y=1 então S=1.1=1

Assim sendo, quando o computador utiliza o operador lógico AND para comparar um número binário com outro, o valor de BIT de cada número é submetido ao processso de AND lógico, com o valor de BIT do outro número, produzindo um terceiro número.
EXEMPLO:

	DECIMAL	BINÁRIO
	3	0011
	(AND lógico)	
	5	0101
=	1	0001

Expressões lógicas ou de relação assumem valor 1(um) se são verdadeiras e 0(zero) se são falsas. AND pode também ser utilizado para comparar strings.
EX:

```
200 IF A >3 AND A < 7 THEN PRINT "A E  MAIOR QUE 3 E MENOR QUE 7"
```

ORTOGRAFIA ALTERNATIVA
Alguns computadores de fabricação inglesa aceitam A. por AND

VARIAÇÕES DE USO
AND(Operador E) operador lógico que, aplicado a duas proposições

P e Q, tem como resultado uma proposição R que só é verdadeira se P e Q também o forem, em algumas famílias de microcomputadores.
A instrução AND(p.q) é utilizada pelo computador WANG 2200B para computar o AND binário lógico de dois valores hexadecimais ou dois strings de caracteres. O primeiro valor, P, deve ser uma variável de string e o segundo ou uma variável de string ou uma constante hexadecimal com dois dígitos. O valor resultante substitui o primeiro dos dois valores.
Se X$ = número hexadecimal AA e Y$ = número hexadecimal OF, por exemplo, AND (X$,Y$) faz X$= número hexadecimal OA.

HEXADECIMAL	BINÁRIO
X$ = AA	10101010
AND	
Y$ = OF	00001111
X$ = OA	00001010

Sendo X$ um string de caracteres e Y uma constante hexadecimal, cada carácter no string é convertido para seu valor ASCII e AND"-ado" ao constante hexadecimal. Os resultados são convertidos de volta para forma de caracteres e armazenados em X$. Assim sendo, "EFG"(string) AND(-ado) com hexadecimal 43 resulta em string "ABC"

VEJA TAMBÉM
OR, XOR, NOT, *, +, =, <, >, <>, <=, >=

Comando/DOS **APPEND**

APPEND é utilizado como comando de manipulação de arquivo, ele deve ser seqüencial.
APPEND é um comando para conjugar um programa vindo de armazenamento externo tal como Disquete ou Fita Magnética, com outro já existente na memória. Este comando APPEND apresenta-se com sintaxes diferentes dependendo do computador (Consulte o manual de instruções do seu computador).

SINTAXE GERAL (para computadores que seguem a família APPLE como AP II - Unitron, MICRO-ENGENHO, MAX, etc)

Número da linha PRINT CHR$(4);"APPEND nome do Arquivo de Dados" {,Dn}{,Sn} {,Vn}
(verifique no manual os opcionais Dn,Sn,Vn).
O drive Dn, slot Sn e volume Vn podem ser especificados em qualquer ordem. Se Dn ou Sn for omitido, o drive ou slot referenciado por último será usado.
APPEND é um comando do Sistema Operacional com Disquete - podendo ser utilizado somente quando já se tem carregado na memória RAM do computador o DOS (DOS 3.3 DO APPLE II),requerendo PRINT e CTRL-D no modo programa.
APPEND permite adicionar dados no final de um arquivo seqüencial. Ele é bastante útil nos casos em que se deseja estender a informação dentro de um arquivo seqüencial.
APPEND abre um arquivo seqüencial e posiciona o apontador do arquivo no final.
APPEND não pode ser utilizado no modo direto.

EXEMPLO
Primeiramente criamos um arquivo chamado FICHÁRIO, que contém duas cadeias de caracteres "FICHA 001" e "FICHA 002".

Armazene este programa em um disquete e execute o mesmo.

```
100 REM ARMAZENE COM NOME CRIA FICHARIO
110 D$=CHR$(4)
120 PRINT D$;"OPEN FICHARIO"
130 PRINT D$;"DELETE FICHARIO"
140 PRINT D$;"OPEN FICHARIO"
150 PRINT D$;"WRITE FICHARIO"
160 PRINT "FICHA 001"
170 PRINT "FICHA 002"
180 PRINT D$;"CLOSE FICHARIO"
190 END
```

No próximo programa usamos APPEND para adicionar as cadeias de caracteres "FICHA 003" "FICHA 004" "FICHA 005" no arquivo fichário.

```
100 REM USANDO O COMANDO APPEND
115 D$ = CHR$(4)
110 PRINT D$;"APPEND FICHARIO"
120 PRINT D$;"WRITE FICHARIO"
```

```
130 PRINT "FICHA 003"
140 PRINT "FICHA 004"
150 PRINT "FICHA 005"
160 PRINT D$;"CLOSE FICHARIO"
170 END
```

OBS.: O comando OPEN sempre posiciona o ponteiro "posição dentro do arquivo" para o byte 0, isto é, para o primeiro carácter no arquivo. O comando APPEND, por sua vez, executa automaticamente um OPEN para o arquivo selecionado e, então, posiciona o ponteiro "posição dentro do arquivo" para o próximo byte depois de último carácter do arquivo aberto. Sendo executado o programa que contém o comando APPEND, o arquivo de dados seqüencial conterá agora:
FICHA 001 FICHA 002 FICHA 003 FICHA 004 FICHA 005.
O comando APPEND deve vir seguido de um comando WRITE. A tentativa de usar READ ocasionará a mensagem:

FIM DE DADOS

SINTAXE GERAL (para computadores que seguem a linha do TRS-80 Mod I e III como CP-500)

APPEND Arquivo-fonte Arquivo-destino

O comando APPEND copia o conteúdo do arquivo-fonte no final do arquivo-destino. O arquivo-fonte não é alterado, enquanto que o arquivo-destino é ampliado para incluir o arquivo-fonte.
OBS.: Arquivo-fonte é a especificação do arquivo que será copiada (anexada) no final de um outro. Arquivo-destino é a especificação de Arquivo que recebeu o acréscimo (o anexo).

NOTA Tanto o arquivo-fonte quanto o arquivo-destino devem possuir o formato ASCII (arquivos de dados ou programas de BASIC armazenados <SAVE> com a opção A) e os comprimentos de gravação lógica devem combinar.

EXEMPLO
Suponha que você tem dois arquivos de dados: FOLHAPAG/A e FOLHAPAG/B

FOLHAPAG/A	FOLHAPAG/B
Pereira,C.	Alcantara, P.M.
Souza,J.R.	Benvenuti,R.A
Garcia,C.N.	Cardoso, A.
Nogueira,A.	Burchuetti,D.

Você pode combinar os dois arquivos com o comando:

```
APPEND FOLHAPAG/B FOLHAPAG/A
```

A folha de pagamento A se apresenta assim:

```
FOLHAPAG/A
Pereira,C.
Souza,J.R.
Garcia,C.N.
Nogueira,A.
Alcantara,P.M.
Benvenuti,R.A.
Cardoso,A.
Burchuetti,D.
```

A FOLHAPAB/B será inalterada. Para observar o arquivo gerado, digite LIST FOLHAPAG/B(ASCII).
OBS.: Não carregue um comando em BASIC depois de um comando APPEND.

CONSIDERAÇÕES GERAIS
Sendo o comando APPEND utilizado para conjugar um programa vindo de armazenamento externo, os números de linha do programa que está sendo trazido "de fora" devem ser superiores ao último número de linha do programa que já se encontra na memória.
APPEND PROG 3, por exemplo, fará com que o programa seja trazido de fora para APPEND no final do programa já residente.

EXEMPLO PRÁTICO
Vamos admitir que o material está armazenado em fita cassete. Armazene este curto programa na fita cassete como prog3. (Para informações a respeito, veja CSAVE):

```
2000 PRINT "ESTAS LINHAS PERTENCEM AO "
2010 PRINT "PROG3"
2020 END
```

Tecle em seguida NEW para apagar o programa e dê entrada no PROG1:

```
100 REM USANDO APPEND, PROG1
110 PRINT "ESTAS LINHAS PERTENCEM AO "
120 PRINT "PROG1"
130 PRINT "    MAS..."
```

Tecle em seguida APPEND PROG3
Depois que tiver sido carregado o PROG3, RUN (rodar) (utilizando o método empregado pelo seu computador para SAVE PROG3 e APPEND PROG3, isto é, aspas, nomes de letra única etc.)

EXECUÇÃO DO PROGRAMA
```
ESTAS LINHAS PERTENCEM AO
PROG1
    MAS...
ESTAS LINHAS PERTENCEM AO
PROG3
```

APPEND é utilizado muitas vezes para carregar num programa extensos arquivos de dados. Pode também ser utilizado para ANEXAR uma sub-rotina freqüentemente utilizada a um programa já existente.

VARIAÇÕES DE USO
Se seu computador não responder favoravelmente a APPEND, experimente MERGE, TAPPEND ou WEAVE. Se nenhum desses comandos surtir efeito, pode-se fazer com que alguns computadores anexem um programa se for possível encontrar os "indicadores" do programa atual.
Isto se torna um tanto dificultoso. Consulte, portanto, o manual de instruções do seu computador. Com o TRS-80 e utilizado o seguinte procedimento:
Localize o indicador que der o endereço final do programa

residente atualmente na memória. No caso do TRS-80, Modelo I, o referido número se acha armazenado em 16333 e 16334. Utilize PEEK para pegar os valores que estão armazenados ali. Subtraia 2 do primeiro número(ou aquele que estiver armazenado em 16333). Sendo negativa a diferença, adicione-lhe 256 e subtraia 1 do segundo número.
Em qualquer dos casos POKE esses dois números para dentro de 16548 e 16549, sem alterações adicionais.
CLOAD agora o programa, a partir de fita magnética. Restabeleça os valores em 16548-49 com POKEs de 233 para 16548 e 66 para 16549.
O segundo programa fica agora APPENDado ao primeiro.

VEJA TAMBÉM
PEEK, POKE, CLOAD, CSAVE, DATA, TAPPEND

Função **ASC • ASCII**

SINTAXE GERAL

ASC("X") ou ASC(X$)

ASC(valor alfanumérico entre aspas ou variável alfanumérica)
ASC("X") é uma função usada, geralmente, no modo programa devendo ser precedida de uma instrução LET ou PRINT.
Funciona de modo idêntico nos micros da linha Americana devolvendo o equivalente decimal do código ASCII standard. Em Integer-BASIC o "X" pode ser visto ou não.
ASC(alfanumérico) mostra o código ASCII (na forma decimal) para o primeiro carácter do alfanumérico especificado.
A função ASC converte um carácter ou uma variável string para seu número decimal ASCII correspondente. O valor decimal do código ASCII varia de 0 a 255. Se o argumento for numérico deverá também estar entre aspas dentro do parênteses. Ex.: PRINT ASC("2"). Se o argumento dado for um alfanumérico nulo, ocorrerá um erro.
PRINT ASC ("A"), por exemplo, imprime 65, que é o número de código ASCII da letra A. PRINT ASC("AB") imprimirá também 65 que é o código do primeiro carácter do alfanumérico especificado.
Certos computadores que incluem a função ASC podem aceitar strings caracteres mais extensos do que de um carácter, mas se avalia e se converte para código ASCII tão-somente o primeiro carácter.
O argumento pode ser uma expressão envolvendo operadores e funções de alfanuméricos.

EXEMPLO 1

```
100 REM VERIFICANDO APENAS O CODIGO ASCII
110 LET C$ = "CODIGO ASCII "
120 LET L = 40
130 PRINT
140 PRINT TAB((L - LEN(S$))/2); C$
150 PRINT
160 FOR I = 33 TO 53
170 PRINT I; " "; CHR$(I),
180 PRINT I + 21; " "; CHR$(I + 21),
190 PRINT I + 42; " "; CHR$(I + 42)
200 NEXT I
210 END
```

EXECUÇÃO DO PROGRAMA

```
RUN

                    CODIGO ASCII

33 !            54 6            75 K
34 "            55 7            76 L
35 #            56 8            77 M
36 $            57 9            78 N
```

37 %	58 :	79 O
38 &	59 ;	80 P
39 '	60 <	81 Q
40 (	61 =	82 R
41)	62 >	83 S
42 *	63 ?	84 T
43 +	64 @	85 U
44 ,	65 A	86 V
45 -	66 B	87 W
46 .	67 C	88 X
47 /	68 D	89 Y
48 0	69 E	90 Z
49 1	70 F	91 [
50 2	71 G	92 \
51 3	72 H	93]
52 4	73 I	94 ^
53 5	74 J	95 -

Veja a tabela de código ASCII.
Note que o código ASCII para uma letra minúscula é igual ao código ASCII das letras maiúsculas, mais 32.
Então, ASC pode ser usado para converter valores maiúsculos para valores minúsculos.

EXEMPLO 2

```
200 REM USANDO A FUNCAO ASC
210 INPUT C$
220 PRINT "O CODIGO ASCII PARA ";C$;
230 PRINT " E' "; ASC(C$)
240 PRINT
250 GOTO 210
```

EXECUÇÃO DO PROGRAMA
OBS: Na execução desse programa não deverá ser usada a , (virgula) nem " (aspas)

```
RUN
?A
 O CODIGO ASCII PARA A E' 65

?B
 O CODIGO ASCII PARA B E' 66

?Z
 O CODIGO ASCII PARA Z E' 90

?1
 O CODIGO ASCII PARA 1 E' 49

?/
 O CODIGO ASCII PARA / E' 47

?*
 O CODIGO ASCII PARA * E' 42
```

A função ASC pode ser usada também para criar processos de codificação/descodificação.

ORTOGRAFIA ALTERNATIVA

Alguns computadores Americanos e Europeus utilizam ASCII ao invéz de ASC. Computadores que seguem a linha SINCLAIR como TK-82, NE-8000, CP-200 etc. utilizam a função CODE que é o código do primeiro carácter da "string" (ou 0 se a "string" for vazia).
ASC("X") é igual a CODE("X") nos microcomputadores que seguem a linha Sinclair e o valor retornado não obedece ao padrão ASCII e sim ao padrão interno da máquina.

VARIAÇÕES DE USO
Certos interpretadores Basic (assim como MAXBASIC) utilizam a configuração ASC(B$,N), que imprime os números de códig ASCII dos primeiros N caracteres contidos em B$.

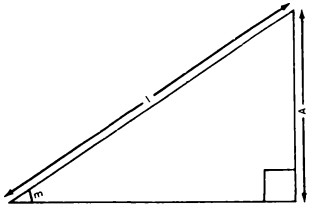
VEJA TAMBÉM
CODE, CHR$ e a tabela do código ASCII no Manual do seu Computador.

Função **ASN • ASNG ASND • ARCSIN**

SINTAXE GERAL

Número da linha LET = ASN(argumento)

ASN(X) é uma função usada geralmente no modo programa devendo ser precedida de uma instrução como LET ou PRINT.
ASN(X) é uma das funções trigonométricas que trabalham em radianos, não em graus, sendo reconhecida por apenas alguns interpretadores BASIC. Em termos matemáticos uma função é uma regra para dar um número (o resultado) em troca de outro (o argumento ou operando) e é realmente uma operação unitária.
A função ASN(X) é utilizada em determinadas versões da linguagem BASIC para calcular o arco-seno da relação X em radianos (não em graus), sendo que um radiano tem aproximadamente 57.2958 graus.
Arco-seno (ASN) é definido como sendo o ângulo (M) criado para determinada relação entre o comprimento do lado que lhe fica oposto (A) para com o comprimento da hipotenusa (I) do triângulo de ângulo de reto.



M = ASN (A/I)
O contrário de ASN e seno (SIN). O seno de um ângulo é a relação do comprimento do lado frente ao ângulo para com o comprimento da hipotenusa do triângulo de ângulo reto.
SIN(M) = A/I

EXEMPLO 1

```
100 REM USANDO A FUNCAO ASN
110 PRINT "DE ENTRADA NUMA RELACAO OU NUM VALOR DE SENO";
120 INPUT A
130 P = ASN(A)
140 PRINT " O ANGULO COM RELACAO A/I DE "; A; " E "; P; "
RADIANOS"
3999 END
```

EXECUÇÃO DO PROGRAMA

```
RUN
DE ENTRADA NUMA RELACAO OU NUM VALOR DE SENO
? .7
O ANGULO COM RELACAO A/I DE .7 E .927408106 RADIANOS
```

Certos computadores calculam o ângulo em graus ou em grados (100 grados = 90 graus). Esses computadores utilizam as funções ASND

por graus e ASNG por grados. Para converter em graus valores expressos em radianos, multiplique o ângulo (em radianos) por 57.2958.

EXEMPLO:
```
R = ASN(A) * 57.2958
```

Para se converterem em radianos valores expressos em graus, multiplique por 0.0174533 o ângulo (em graus).

EXEMPLO:
```
G = A(ângulo expresso em graus) * .0174533
```

ORTOGRAFIA ALTERNATIVA
Computadores que seguem a linha do SINCLAIR (Inglês), como o TK-82, NE-8000, TK-85 utilizam ARCSIN no lugar de ASN. Experimente ARCSIN no lugar de ASN para verificar se seu computador permite seu uso.

SE SEU COMPUTADOR NAO TIVER ESTA FUNÇÃO
Se seu interpretador tiver a função ATN (ARCTANGENT) e SQR (raiz quadrada), mas não tiver ASN, substitua a fórmula:
```
2 * ATN(A/(1+SQR(1-A*A))) para ASN
```
Se seu interpretador não tiver as funções ASN, ATN ou SQR, podem-se substituir as seguintes linhas no Exemplo 1.
```
125 S = A
130 GOSUB 30530
```
e utilize a seguinte sub-rotina:

```
3000 END
3530 REM * ARCOSENO SUBROTINA * INPUT S, OUTPUT B,P
3535 REM B E EM GRAUS, P E EM RADIANOS
3540 REM USANDO TAMBEM VARIAVEIS A, B INTERNAMENTE
3550 A = S: IF ABS(S)<=.707107 THEN 3610
3560 A = 1-S*S: IF A<0 THEN PRINT S;"ESTA FORA DE FILEIRA": STOP
3565 IF A = 0 THEN P = 90/57.29577951: GOTO 3630
3570 P = A/2: C = 0
3580  B  = (A/P-P)/2: IF (ABS(B)<.1E-8) AND (B=C) THEN A = P: GOTO
3610
3600 P = P+B: C=B: GOTO 3580
3610 B = A+A*A*A/6+A*A*A*A*A*.075+A*A*A*A*A*A*A*4.464286E-2
3620 P = B+A*A*A*A*A*A*A*A*A*3.038194E-2
3625 IF ABS(S)>.707107 THEN P = 1.570796-P
3630 B = P* 57.29577951: RETURN
```

Para utilizar esta sub-rotina no Exemplo 1, efetue as seguintes alterações:
```
125 S = A
130 GOSUB 30530
```

VARIAÇÕES DE USO:
Não se tem conhecimento de nenhuma.

VEJA TAMBÉM
ACS, ATN, COS, SIN, SQR, TAN

Função AT • A.

SINTAXE GERAL

AT linha da tela, coluna

AT é utilizado apenas como função, devendo ser precedido de uma instrução, geralmente PRINT.
A função AT é utilizada com PRINT no BASIC Nível I dos computadores da linha TRS-80 e em computadores que seguem a linha SINCLAIR como TK-82, NE-80, NE-8000, TK-85 e CP-200.
AT move a posição PRINT (a posição onde o próximo item vai ser escrito) para a linha e a coluna especificadas. As linhas estão numeradas de 0 (parte superior) a 21 (parte inferior), e as colunas de 0 (à esquerda) a 31 (à direita). O número de linhas e colunas da tela de vídeo varia de computador para computador.
Pode-se utilizar AT para pôr a posição PRINT onde já está algo escrito e o texto será escrito por cima do antigo.

```
EXEMPLO 1
100 REM USANDO A FUNCAO AT
110 FOR N = 21 TO 0 STEP -1
120 PRINT AT 21 -N,N; CHR$ (CODE "0" + N);
130 NEXT N
```

EXECUÇÃO DO PROGRAMA
RUN
Este exemplo poderá ser executado em computadores que seguem a linha do SINCLAIR - TK-82, CP-200, ZE8000 etc
Aparecerá na tela em diagonal L,K,J,I,H,G,F,E,D,C,B,A 9,8,7,6,5,4,3,2,1,0

A função AT é utilizada com instruções PRINT (em BASIC TRS-80 nivel I) para especificar o ponto inicial da instrução PRINT. O valor da função AT pode ser um número, uma variável numérica ou uma operação matemática. Devem ser inseridos entre o valor AT e o string uma vírgula ou um ponto-e-vírgula.

```
EXEMPLO 2
400 REM USANDO A FUNCAO AT
410 CLS
500 FOR X = 1 TO 13
510 PRINT X
515 NEXT X
520 PRINT AT 452, "SE ESTA FOR A LINHA N.8 SEU MICRO ACEITA AT"
525 FOR X = 9 TO 13: PRINT X: NEXT X
530 PRINT AT 800,
540 END
```

EXECUÇÃO DO PROGRAMA
RUN
1

2
3
4
5
6
7
8 SE ESTA FOR A LINHA N.8 SEU MICRO ACEITA AT
9
10
11
12
13

ORTOGRAFIA ALTERNATIVA
O operador @(arroba) é utilizado em alguns microcomputadores que seguem a linha TRS-80 (USA), CP-500, DGT-100, FENIX, etc. ao invés da função AT. Os computadores que utilizam Tiny BASIC permitem também a utilização de A. ao invés de AT.

NOTA: A função AT ou o modificador @ especificam exatamente onde a impressão deve começar, e a localização especificada deve ser um número entre 0 e 1023. Consulte no manual do seu computador o mapa de vídeo para ver a posição exata de cada locaização 0 a 1023.
Sempre que se usar PRINT @, na linha inferior do vídeo, existe uma linha de alimentação automática fazendo com que tudo que foi mostrado mova-se uma linha para cima. Para suprir isto, use um ponto-e-vírgula no final da instrução.

EXEMPLO: PRINT @ 1000,1000;

Você também poderá usar um ponto-e-vírgula (;) toda vez que quiser suprir uma linha de alimentação.

VEJA TAMBÉM
PRINT, @, TAB, HLIN, VLIN, DRAW, XDRAW

Função ATN • ATAN
ATND • ATNG • ARCTAN

SINTAXE GERAL

Número da linha LET variável = ATN(argumento)
ou
ATN(expressão)

ATN é uma função usada geralmente no modo programa devendo ser precedida de uma instrução como LET ou PRINT.
ATN(X) Esta função é semelhante nos equipamentos das famílias APPLE, TRS-80 e SINCLAIR e retorna o arco tangente de um número ou expressão X, expresso em radianos. O ângulo desenvolvido está na faixa entre -II/2 até II/2.
Não é disponível em Integer BASIC.
A função ATN(X) fornece o arco tangente (em radianos não em graus) do argumento, isto é, o ângulo cuja tangente é X.
ATN(X) tem como ação devolver o arco tangente de X em radianos. A expressão X pode ser qualquer tipo numérico, mas a avaliação de ATN sempre é com precisão simples na maioria dos computadores.

EXEMPLO 1
```
100 REM USANDO A FUNÇÃO ATN(X)
110 INPUT X
120 PRINT ATN(X)
```

EXECUÇÃO DO PROGRAMA
```
RUN
? 3
 1.24904577
```

Caso se deseje obter o resultado em graus, basta multiplicar ATN(argumento) por 57.2958, isto porque um radiano corresponde aproximadamente a 57.2958 graus.

EXEMPLO 2
```
200 REM USANDO A FUNCAO ATN PARA RESULTADOS EM GRAUS
210 PRINT "ARCO TANGENTE EM GRAUS"
220 DEF FND(X) = X * 57.2958
230 PRINT
240 PRINT "DE O VALOR PARA TANGENTE"
250 INPUT T
260 PRINT "ARCO TANGENTE = "; FND(ATN(T))
270 END
```

EXECUÇÃO DO PROGRAMA
```
RUN
ARCO TANGENTE EM GRAUS
```

```
DE O VALOR PARA TANGENTE
? 1
ARCO TANGENTE = 45.0000161
```

NOTA: Para converter os valores de graus para radianos, multiplique o ângulo (em graus) por 0.0174533.
EXEMPLO:
Radianos = Ângulo(em graus) * 0.014533

NOTA IMPORTANTE:
A maioria dos computadores tem somente ATN como sua "função trigonométrica inversa", encontrando-se raramente ARCCOS e ARCSIN. Isto deixa ATN como a única "janela" através da qual todos os ângulos possam ser calculados e devolvidos para o "exterior".
Evidentemente, porém, se vai ser utilizado ATN, o TAN deve ser conhecido ou ser capaz de ser determinado.
As fórmulas abaixo relacionadas permitem converter qualquer relação para TAN, e daí para o proprio ângulo em si, mediante o uso de ATN.

$TAN = 1/COT$ $\quad TAN = \sqrt{\frac{1-COS^2}{COS^2}}$ $\quad TAN = \frac{1}{\sqrt{\frac{1-SEN^2}{SEN^2}}}$

$TAN = \frac{1}{\sqrt{CSC^2-1}}$ $\quad TAN = \sqrt{SEC^2-1}$

Essas fórmulas utilizam as relações entre as funções trigonométricas que nos proporcionam meios para calcular cada uma das funções inversas.
Cada uma das funções inversas codificadas em BASIC se apresenta da seguinte forma:

```
ARCCOS(X) = 1.5708 - 2*ATN(X/(1+SQR(1-X*X)))
ARCCOT(X) = ATN(1/X)
ARCCSC(X) = ATN(1/SQR(X*X-1)
ARCSEC(X) = ATN(SQR(X*X-1))
ARSSIN(X) = 2*ATN(X/(1+SQR(1-X*X)))
```

ORTOGRAFIA ALTERNATIVA
Determinados computadores utilizam para a função Arco tangente ATAN, ATND ou ATNG. Alguns computadores que seguem a linha SINCLAIR utilizam ARCTAN no lugar de ATN.

VEJA TAMBÉM
ACS, ASN, COS, SIN, TAN

Comando/DOS **ATTRIB • ATRIB**

SINTAXE GERAL

ATTRIB arquivo(visibilidade,ACC = nome, UPD = nome, PROT = nível)

ATTRIB Este comando é utilizado no DOS (Sistema Operacional de Disco) em computadores que seguem a linha do TRS-80 Mod I e III, assim como o CP-500 da PROLOGICA (DOS/500).

Arquivo é a especificação de arquivo.

Visibilidade deve ser I ou N. Diz ao DOS (Sistema Operacional de Disco) se o arquivo é invisível (I) ou não invisível (N), se omitido a visibilidade será inalterada.

ACC = nome diz ao DOS a senha de acesso.

Se omitida ela será inalterada. Se apenas o nome for omitido, (ACC=,) a senha de acesso será representada por espaços em branco (inexistentes).

UPD = diz ao DOS a senha para atualização do arquivo, se omitido, não será alterado. Se apenas o nóme for omitido (ACC=,) a senha de atualização será representada por espaços em branco (inexistentes).

PROT = nível informa ao DOS (Sistema operacional de disco) a senha de acesso. Caso seja omitido, o "nível" permanecerá o mesmo.

NIVEL grau de acesso dado à palavra de acesso.

FULL acesso total, nenhuma proteção.

KILL supressão (KILL), alteração de nome, leitura, execução e gravação (permite acesso total, isto é, o menos protegido).

NAME alteração de nome, leitura, execução e gravação.

WRITE leitura, execução e gravação

READ leitura e execução.

EXEC somente execução

O comando ATTRIB permite a mudança das senhas para um arquivo existente e o torna invísivel ou não. As senhas são determinadas quando o arquivo é criado. Nesse momento, as palavras atual e de acesso são igualadas a um mesmo valor (por especificação ou por omissão)

EXEMPLOS

ATTRIB ARQUIVO (I,ACC = 14/JULHO,UPD = RATO, PROT = READ)

Esse comando torna o arquivo invisível e determina as senhas de acesso (JUNHO/14) e a atualização (RATO). A utilização da senha de acesso possibilitará somente a leitura e a execução do arquivo.

ATTRIB FOLHAPAG/BAS.SECRETO (N,ACC=,)

Neste caso, a senha de acesso é constituída de espaços em branco. O arquivo é não-invisível e o nível de proteção determinado para a senha não é alterado:

ATTRIB VELHA/DAT.MACAS (UPD =,)

Neste caso, a senha de atualização consiste em espaços em branco.

ATTRIB FOLHAPAG./BAS.SN (PROT=EXEC)

Aqui, as senhas de atualização e de acesso ficam inalteradas, mas modifica-se o nível de acesso.

EXEMPLOS PRÁTICOS

Suponha que tem um arquivo de dados FOLHAPAG (folha de pagamento) e deseja que seu subordinado o utilize na preparação de cheques de pagamentos, e que seja capaz de lê-lo, mas não de modificá-lo. Então, deve usar esse comando.

ATTRIB FOLHAPAG (I,ACC = DIAPAG, UPD = ABACATE, PROT = READ)

Agora, diga-lhe para usar a senha DIAPAG (dia de pagamento)(que só permite leitura), enquanto só você sabe a senha (abacate), que dá acesso total ao arquivo.

PROTEÇÃO DE PROGRAMA DO BASIC

Você pode dar uma proteção de somente-execução para programas em BASIC, utilizando o comando ATTRIB. Por exemplo, suponha que o programa é denominado TESTE (sem senha)

Na condição
DOS500 ATIVO,
execute este comando:
ATTRIB TESTE (ACC=,UPD=VALE,PROT=EXEC)

Agora, TESTE tem uma senha de acesso em branco, uma atual (VALE) e um nível de acesso de somente-execução. Só existe um meio de se executar o programa sem usar a senha atual e é este:

1- Acione o BASIC
2- Digite: RUN "TESTE"

(Este é o único meio de se ter acesso ao programa. Se o operador tentar carregá-lo, o BASIC apagará o programa da memória antes de retornar à condição READY).

Após a execução do comando RUN "TESTE", o BASIC carregará e executará o programa. Se o operador pressionar a tecla BREAK ou o programa finalizar normalmente, o BASIC apagará o programa antes de retornar com a mensagem READY. Isto impossibilita a obtenção de uma listagem do programa, a não ser que a senha atual seja usada. Neste caso, você terá acesso completo ao programa.

ORTOGRAFIAS ALTERNATIVAS

Alguns computadores (tais como LABO em seu Sistema Operacional SOL 8221) utilizam ATRIB no lugar de ATTRIB, sendo utilizado como descrito abaixo:

ATRIB exibe, marca e elimina os atributos de discos e arquivos; exibe o código do usuário corrente e dos demais usuários do disco.

ATRIB exibe o quadro dos próprios comandos.

ATRIB: exibe o estado de todos os discos registrados e sua área disponível.

ATRIB # : exibe o estado do disco especificado e sua área disponível.

ATRIB /U exibe o código do usuário corrente e o código dos demais usuários do disco.

ATRIB /+L protege o disco corrente contra gravação.

ATRIB # : /+L protege o disco da unidade especificada contra gravação.

ATRIB /-L libera o disco corrente para gravação.

ATRIB # : /-L libera o disco da unidade especificada para gravação.

```
                    + L
ATRIB ARQ. FAM/     + E
                    + L + E
                    + E + L
```

protege o arquivo especificado contra gravação e/ou eliminação.

```
                    - L
                    - E
ATRIB ARQ.FAM/      - L - E
                    - E - L
```

libera o arquivo especificado para gravação e/ou eliminação.

```
                    - L + E
                    - E + L
ATRIB ARQ.FAM/      + L - E
                    + E - L
```

protege e/ou libera o arquivo especificado para (contra) gravação e/ou eliminação.

código de uma determinada unidade.
ARQ.FAM nome de um determinado arquivo
\+ marcar um atributo (proteger contra)
\- desmarcar um atributo (liberar para)
E eliminação
L leitura

VARIAÇÕES DE USO
Não se tem conhecimento de nenhuma.

Comando **AUTO**

SINTAXE GERAL

AUTO
ou
AUTO número da linha [,incremento]

AUTO é utilizado como comando no modo imediato.

AUTO tem por finalidade gerar uma numeracao automática de linha em Integer BASIC, toda vez que se digita a tecla RETURN (em determinados teclados ENTER, CR, EOF etc).

Este comando ativa uma função automática de numeração de linhas para a entrada conveniente de programas - tudo que se tem a fazer é introduzir as instruções reais do programa. Pode ser especificado no comando AUTO o número da linha inicial e o valor do incremento entre as linhas.

Só pode ser usado em modo imediato, não é disponível em APPLESOFT.

AUTO XX funciona de maneira idêntica nos computadores que aceitam este comando produzindo uma numeração automática das linhas de um programa com XX especificando o início. Se não houver especificação de incremento, a máquina saltará de 10 em 10.

AUTO XX,YY mesma função porém YY é o incremento que determina o aumento no número de linhas por degraus.

EXEMPLO:

GERA A NUM.AUTOM.DAS LINHAS

AUTO	10,20,30...
AUTO 10,5	10,15,20,25...
AUTO 1,1	1,2,3,4...
AUTO 200	200,210,220...
AUTO 1,100	1,101,201,301...

OBS.: Não sendo especificado no comando AUTO o número da linha inicial, o valor do incremento será considerado pelo computador o valor 10 para os dois parâmetros. Neste caso, a numeração de linhas começará em 10 e usará acréscimos de 10 a cada entrada da linha.

EXEMPLO:
AUTO
Gera os números de linha 10,20,30,40...

OBS.: Se o comando AUTO vier seguido de uma vírgula e do incremento, o computador deverá gerar a linha inicial 0 com acréscimos de acordo com o incremento.

EXEMPLO:
AUTO, 3
Gera os números de linha 0,3,6,9,12...

NOTA: Se o comando AUTO gerar um número de linha que já esteja sendo usado, um asterisco será impresso após o número para avisar ao programador que qualquer entrada substituirá a linha existente na memória e gerará o número da linha seguinte.

Para desligar a função AUTO, alguns computadores requerem que se aperte a tecla BREAK, ao passo que em outros é preciso teclar CTRL-C ou ESC.

EXEMPLO 1
Tecle AUTO 100,5 e dê entrada neste programa:

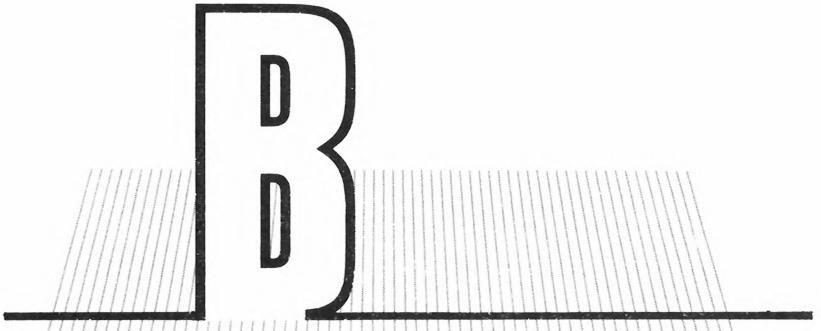
```
100 REM USANDO O COMANDO AUTO
105 PRINT "O PROXIMO NUMERO DE LINHA SERA' AUMENTADO DE 5"
110 PRINT "APERTE A TECLA BREAK PARA DETER O COMANDO AUTO"
500 END
```

Uma vez desligado o comando AUTO mediante a tecla BREAK, o BASIC retornará ao nível de comando podendo ser lançados números de linha fora de seqüência (tais como 500).

Dê entrada novamente em AUTO 100,5 ao que a linha 100 deve ser impressa, seguida de um (*) asterisco, sinal de que as informações já estão armazenadas numa linha 100; sendo pressionada a tecla ENTER (ou equivalente), será apagada a mensagem. Entretanto, podemos manter as informações originais usando a tecla BREAK e repetindo o comando AUTO seguido do número da próxima linha e do incremento.

VARIAÇÕES DE USO
Não se tem conhecimento de nenhuma.

VEJA TAMBÉM
BREAK, LIST, MAN, CTRL-C



BASE • BLOAD • BREAK
BRUN • BSAVE • BUILD • BYE
GOODBYE

Instrução **BASE**

SINTAXE GERAL

Número da linha BASE valor

BASE é utilizada no modo programa.
BASE é utilizada em determinados computadores (tais como os que funcionam com a Versão 3 do BASIC da Control Data) para definir como 0 ou 1 o valor do elemento do arranjo variável de BASE (mais baixo).

```
EXEMPLO 1
100 REM USANDO A INSTRUCAO BASE
110 BASE 0
120 DIM A(3)
```

A instrução BASE 0 define esse arranjo como sendo de quatro elementos A(0) até A(3) .
Muitos computadores estabelecem automaticamente os elementos de arranjo de 0 a 10 (11 elementos) sem DIMemsionamento prévio. A instrução BASE permite mudar essa faixa dos 11 elementos normais (0 a 10) para 10 (1 até 10) e de volta.
Pode-se utilizar normalmente num programa apenas uma instrução BASE, devendo ser executada a mesma antes de as instruções DIM e de as variáveis de arranjo serem manipuladas.

```
EXEMPLO 2
200 REM USANDO A INSTRUCAO BASE
210 BASE 0
220 DIM A(3)
230 FOR R = 0 TO 3
240 A(R) = R
250 NEXT R
260 FOR R = 0 TO 3
270 PRINT A(R);
280 NEXT R
290 PRINT "A INSTRUCAO BASE APROVOU"
300 END

EXECUÇÃO DO PROGRAMA
RUN
0 1 2 3  A INSTRUCAO BASE APROVOU
```

Alguns computadores (tais como os que utilizam MAXBASIC) permitem mais de uma instrução BASE num programa e permitem que o valor BASE seja definido como qualquer valor de um número inteiro.

```
EXEMPLO 3
300 REM INSTRUCAO BASE
310 BASE 6
320 DIM A(10)
```

A instrução BASE 6 define esse arranjo de sete elementos

A(6) até A(10).

```
EXEMPLO 4
400 REM USANDO A INSTRUCAO BASE
410 BASE 3
420 DIM A(5)
430 FOR R = 3 TO 5
440 A(R) = R
450 NEXT R
460 BASE 0
470 FOR R = 0 TO 2
480 A(R) = R
490 NEXT R
500 FOR R = 0 TO 5
510 PRINT A(R);
520 NEXT R
530 PRINT "A INSTRUCAO BASE APROVOU"
540 END
```

EXECUÇÃO DO PROGRAMA

```
RUN
0 1 2 3 4 5 A INSTRUCAO BASE APROVOU
```

VARIAÇÕES DE USO

A linguagem ANSI BASIC inclui a instrução OPTION

VEJA TAMBÉM

DIM, OPTION

Comando/DOS **BLOAD**

SINTAXE GERAL

BLOAD nomearq [,Aposmem] [,Dn] [,Sn] [,Vn]

BLOAD é um comando do DOS nos computadores que seguem a linha APPLE II; para ser utilizado no modo programa é necessário utilizar PRINT e CTRL-D.

Se o parâmetro A está ausente, o arquivo especificado é colocado na memória a partir da posição de onde foi tirado na gravação. Se o parâmetro A está presente, o arquivo vai para a memória para posmem.

O BLOAD deve ser usado com cuidado. Aquilo que estiver na área de memória onde ele coloca o arquivo (como um programa, o Applesof,DOS etc.) será destruído.

Se o arquivo não existe no drive Dn no slot Sn, aparece a mensagem FILE NOT FOUND.

Se o disco do drive Dn do slot Sn não tiver o volume Vn, resulta no erro VOLUME MISMATCH.

Dn,Sn,e Vn podem ser especificados em qualquer ordem. Se Dn ou Sn são omitidos, o drive ou slot referenciados por último são assumidos. Usa-se VO se Vn está ausente.

Este é um comando do DOS, requer PRINT e CTRL-D em modo programado.

VEJA TAMBÉM
BSAVE, LOAD, CLOAD

Tecla/Instrução **BREAK**

SINTAXE GERAL

Número da linha BREAK número da linha

BREAK pode ser utilizado em alguns computadores como instrução no modo programa ou simplesmente, em outros, usando a própria tecla BREAK.

A instrução BREAK pode ser utilizada para fazer com que se detenha a execução do programa em qualquer número (ou quaisquer números) de linha, por colocar o(s) número(s) de linha (separados por vírgulas) após a instrução BREAK.

Como tecla, BREAK interrompe o processamento que está sendo executado ou a lista que está sendo fornecida e prepara o computador para outro comando pelo teclado.

Nos computadores que possuem a tecla BREAK, quando pressionada, o computador ignora a linha que você digitou e retorna ao nível de comando.

EXEMPLO:

100 PRINT "1235" (pressione a tecla BREAK)

No vídeo aparecerá:

100 PRINT "1235" porém a linha não será armazenada na memória do computador.

Como tecla BREAK também é utilizada para interromper um programa, quando este tiver entrado em um anel fechado (loop) no qual ficaria indefinidamente. É preciso então interromper a execução através de um comando que tenha prioridade sobre a seqüência de instruções do programa, e BREAK tem esta capacidade, pois todos os outros comandos estarão inoperantes durante a execução do programa.

Nos microcomputadores que seguem a linha APPLE, a tecla BREAK equivale ao acionamento conjunto das teclas CONTROL e C que fará com que interrompa a execução do programa.

BREAK é utilizada como instrução em alguns computadores para mandar que uma ou mais linhas do programa parem a execução, colocando o computador no modo de monitoragem ou direto, funcionando de maneira semelhante a uma instrução STOP.

```
EXEMPLO 1
100 REM USANDO A INSTRUCAO BREAK
110 BREAK 120, 140-160
120 PRINT "O COMPUTADOR PAROU A EXECUCAO NA LINHA 120"
130 REM TECLE O COMANDO CONT PARA CONTINUAR
140 PRINT "LINHA 140"
150 PRINT "LINHA 150"
160 PRINT "E LINHA 160"
170 END

EXECUÇÃO DO PROGRAMA
RUN
O COMPUTADOR PAROU A EXECUÇÃO NA LINHA 120
LINHA 140
LINHA 150
E LINHA 160
```

A execução do programa prossegue depois de cada BREAK, ao se teclar CONT (CONTinue). BREAK aceita também uma série de número de linhas, ao se colocar um tracinho (-) entre o primeiro e o último número da série.

No exemplo 1: 110 BREAK ...., 140-160 para a execução do programa no final de cada linha de 140 até 160.

OBS.: Ao contrário da instrução END, que (em alguns computadores apenas) faz restabelecer em ZERO todas as variáveis, ao se executar a instrução BREAK os valores armazenados nos endereços de variáveis são mantidos.

VARIAÇÕES DE USO

Muitos dos teclados de computadores possuem a tecla BREAK para permitir a interrupção manual do programa.

VEJA TAMBÉM

STOP, CONT, END

Comando/DOS **BRUN**

SINTAXE GERAL

BRUN nome arquivo [,Aposmem] [,Dn] [,Sn] [,Vn]

BRUN é um comando do DOS (sistema operacional com disquete) utilizado e reconhecido nos microcomputadores que seguem a linha APPLE II.

BRUN deve ser precedido de PRINT e CTRL-D no modo programa.

Pega um arquivo binário (que poderia ser um programa em linguagem de máquina), guarda-o numa posição específica da memória, e executa um salto em linguagem de máquina (JMP no código Assembler do 6502) para a posição inicial de memória.

Se o parâmetro A está ausente, o arquivo especificado é colocado na memória iniciando na posição de onde ele foi tirado para gravar (veja BSAVE). Se o parâmetro A estiver presente, o arquivo é colocado a partir da posição posmem.

Um programa em linguagem de máquina pode operar bem em qualquer posição de memória.

Cuidado com as instruções que são dependentes de endereço antes de carregar o programa para uma área diferente de memória.

O BRUN destrói tudo que estiver na área de memória onde ele coloca o arquivo; isso pode destruir o APPLESOFT ou DOS no processo.

Se o arquivo não existe no drive Dn, aparece a mensagem de erro FILE NOT FOUND.

Se o disco no drive Dn e slot Sn não tem o volume Vn, aparece a mensagem de erro VOLUME MISMATCH.

Dn,Sn e Vn podem ser especificados em qualquer ordem. Se Dn e Sn são omitidos, o drive e slot referenciados por último são usados. Usa-se V0 quando Vn está ausente.

OBSERVAÇÃO: Aposmem = Aposição da memória.

VEJA TAMBÉM
RUN

Comando/DOS **BSAVE**

SINTAXE GERAL

BSAVE nomearq, Aposmem, Lcompr, [,Dn] [Sn] [,Vn]

BSAVE é um comando do DOS nos microcomputadores que seguem a linha APPLE II utilizado para copiar ARQUIVO em disquete e armazena este arquivo em BINARIO. O parâmetro A especifica o endereço inicial de memória em binário a ser armazenada. O parâmetro L especifica o comprimento da imagem binária a ser armazenada.

O comprimento é expresso em números de bytes, devendo ser um número inteiro entre 0 e 32767 (decimal), podendo também ser um número hexadecimal precedido pelo sinal de dólar ($).

Se o disco no drive Dn do slot Sn não tem o volume Vn, ocorre o erro VOLUME MISMATCH.

Dn,Sn e Vn podem ser especificados em qualquer ordem. Se Dn ou Sn é omitido, assume-se o último drive ou slot que foi acessado. Usa-se V0 se Vn está ausente. Também o n pode estar ausente, e no caso é adotado D0, S0 e V0.

Este é um comando DOS, requer PRINT e CTRL-D quando usado em modo programado.

VEJA TAMBÉM
SAVE

Comando/DOS **BUILD**

SINTAXE GERAL

BUILD nomearq

Cria um arquivo de introdução automática de comando; nos microcomputadores que seguem a linha TRS-80 modelo I e III é utilizado com o sistema operacional, com disquete, assim como o CP-500 fabricado pela Prologica (DOS 500).

nomearq arquivo é uma especificação de arquivo que não pode conter uma extensão.

Este comando possibilita a criação de um arquivo de introdução automática de comandos que não pode ser executada através do comando DOS.

O arquivo deve conter dados que normalmente seriam digitados através do teclado no modo DOS500 ATIVO.

O objetivo do comando BUILD é transferir as linhas de comando para o DOS500 exatamente como foram digitadas na condição DOS500 ATIVO.

Ao se introduzir o comando BUILD, este cria um arquivo e imediatamente solicita a inserção de linhas. Ao completar uma linha, pressione ENTER.

Você pode utilizar teclas normais de controle do cursor para apagar e fazer correções.

Para finalizar o arquivo BUILD, pressione simplesmente a tecla BREAK no começo da linha.

Primeiro digite: BUILD ARQUIVO

Então será solicitado a digitar o texto do comando, que poderá conter até 63 caracteres e logo após deverá pressionar a tecla ENTER.

Pode introduzir tantas linhas quantas desejar.

Pressione BREAK para terminar e retornar à condição DOS500 ATIVO.

EXEMPLO DE BUILD-ARQUIVO

Aqui apresentamos um comando hipotético BUILD-arquivo que inicializa a interface serial e o drive de impressora:

```
SETCOM (BAUD=1200;WAIT)
FORMS (WIDTH=80)
PAUSE INICIALIZADAS IMPRESSORA E INTERFACE RS-232-C
```

Comando/Instrução **BYE GOODBYE**

SINTAXE GERAL

BYE é reconhecida como comando direto ou como instrução por apenas alguns interpretadores ou compiladores BASIC.

BYE como comando é utilizado para dar fim ao programa com o BASIC. A maioria dos computadores de grande porte aceita BYE como comando de desligamento, terminando com ele a tarefa do usuário.

Em diversos computadores de tamanho pequeno (MICROCOMPUTADORES tais como: ATARI) se dá a entrada no nível do MONITOR ou no Sistema de Operação do Disco, quando teclamos a instrução BYE.

ORTOGRAFIAS ALTERNATIVAS

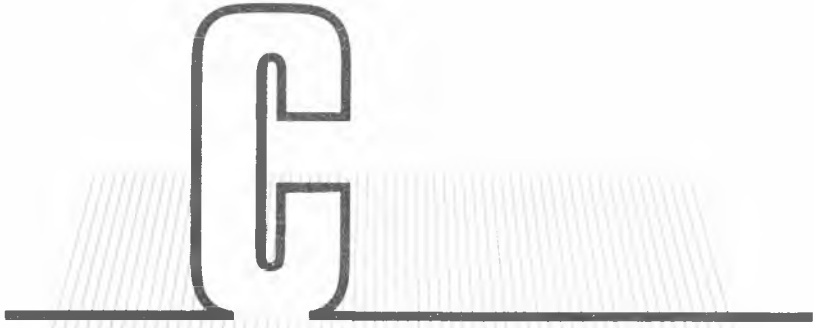
Os sistemas de tempo compartilhado aceitam muitas vezes GOODBYE em vez de BYE, ao se desligar o usuário do computador. Examine o seu manual para verificar o procedimento de desligamento de sua instalação.

VARIAÇÕES DE USO

Não se tem conhecimento de nenhuma.

VEJA TAMBÉM

SYSTEM



CALL • CATALOG
CDBL • CH • CHAIN • CHANGE
CHR$ • CHAR • CHAR$ • CHR
CINT • CLEAR M • CLEAR • CLR
CLG • CLOG • CLK$ • CLK • CLOAD
CLOCK • CLOSE • $CLOSE • CLRDOT
CLS • CODE • COLOR • COMMON
COM • CON • CONT • CONT • CON
CO • C. • CODE • COLOR • COMMON
COM • CON • CONT • CONT • CON
CO • C. • COPY • COPYA • COPI • COS
COSD • COSG • COSH • CSH • COUNT
CREATE • CSAVE • CSNG
CVI • CVS • CVD • CUR

Comando/Instrução **CALL**

SINTAXE GERAL

Número da linha CALL endmen

CALL pode ser utilizado como comando no modo direto ou como instrução no modo programa em microcomputadores que seguem a linha APPLE II (americano). Em outros microcomputadores USR(0) equivale ao CALL. Não há equivalente no nível I. No nível II a função pode ser aproximadamente escrita da seguinte forma:

POKE 16536,Y: POKE 16527,Z: X=USR(0)

onde:

Y designa o byte mais significativo e o Z o byte menos significativo em um micro da linha APPLE II.

CALL X permite transferir o fluxo do programa para uma sub-rotina em linguagem de máquina na posição de memória a partir do endereço decimal especificado pelo argumento X.

O valor do argumento X deve estar entre -65535 e 65535. Se X não estiver nesta faixa, o computador emitirá uma mensagem de erro: VALOR ILEGAL.

EXEMPLOS:

CALL -151 é uma chamada para o endereço FF69, mas o BASIC não reconhece números hexadecimais, só reconhecendo o decimal equivalente ao FF69 neste comando.

Uma vez digitado CALL -151 seguido da tecla RETURN aparece um asterisco, seguido do cursor; neste ponto já se está no Programa Monitor (Linguagem Binária)

CALL-936

Limpa todos os caracteres sobre a janela de texto, e move o cursor para o topo na posição mais à esquerda da janela. Este comando é equivalente a ESC @ e ao comando HOME.

CALL -958

Apaga todos os caracteres escritos no texto desde a posição atual do cursor até a linha inferior. É equivalente a ESC F.

CALL -922

"Alimenta" uma linha. É equivalente a CTRL J (Control J).

CALL -912

Rola o texto para cima, uma linha, isto é, move cada linha de texto uma posição para cima. Caracteres fora da janela definida não são afetados.

Ex.: X = PEEK(-16384)

Lê o teclado. Se X > 127 indica que uma tecla foi pressionada, e X é o valor ASCII da tecla pressionada.

CALL -868

Apaga a linha atual desde o cursor até a margem direita. É

equivalente a ESC E.

CALL -1994

Limpa as 20 linhas superiores da página de texto 1 para colocar o sinal @. Se você está na página 1 modo gráfico de baixa resolução, as 40 linhas superiores da tela gráfica serão limpas.

CALL -1998

Limpa inteiramente a página de texto 1 para o sinal @. Se está na página 1 modo gráfico em baixa resolução em tela completa, a tela inteira será limpa.

CALL 62450

Limpa a atual tela em alta resolução(o computador lembra qual a última tela usada).

CALL 62454

Limpa a atual tela em alta resolução (o computador lembra qual a última tela usada).

A instrução CLL(n) transfere o controle do programa para uma rotina de linguagem de máquina que tem um ponto de entrada no ponto N da memória. A rotina pode ser parte do software do sistema do computador ou poderá ser escrita pelo usuário. Programas escritos pelo usuário em linguagem de máquina podem ser introduzidos a partir do teclado mediante instruções POKE, por um programa BASIC ou pela utilização de um programa "monitor/editor" ao nível de sistema.

EXEMPLO:

CALL 18624 fará com que comece a ser executado um programa em linguagem de máquina armazenado no endereço decimal 18624. Quando se encontra no programa de linguagem de máquina uma instrução RETURN, estamos de volta na linguagem BASIC, sendo executada a instrução que segue a instrução CALL. Para testar a instrução CALL, se deve carregar no computador uma rotina de linguagem de máquina ou localizar o ponto de entrada de uma sub-rotina residente. Consulte o manual de seu computador para informações quanto à maneira de fazer isto.

VARIAÇÕES DE USO

Alguns computadores utilizam a instrução CALL para desviar a determinada sub-rotina BASIC específica. Nesses computadores, CALL pode ser utilizado assim como GOSUB, exceção feita ao se utilizar um nome de sub-rotina em vez de um número de linha. Utilizando-se a instrução CALL dessa maneira, a sub-rotina começa com uma instrução SUB que contém o nome e termina com SUBEND.

```
EXEMPLO 1
100 REM USANDO A INSTRUCAO CALL PARA DESVIO DA SUB-ROTINA
110 CALL TESTE
120 PRINT "CALL DESVIOU PARA SUB-ROTINA "; A$
130 GOTO 170
140 SUB TESTE
150 A$ = "TESTE"
160 SUBEND
170 END
```

EXECUÇÃO DO PROGRAMA
RUN
CALL DESVIOU PARA SUB-ROTINA TESTE

NOTAS:
No computador CP-500 caso se deseje transferir o USR(N) para sua rotina, devemos incluir a instrução CALL abaixo de sua rotina USR.
EXEMPLO: CALL 0A7FH
Esta instrução carrega o argumento N no par de registros HL como um inteiro de 2 bytes com sinal.

Na versão BASIC 8220 do minicomputador LABO permite o uso de sub-rotinas escritas em assembler através do comando CALL. A interface provida permite que o controle seja passado do programa BASIC para a sub-rotina e que variáveis do programa principal sejam consultadas e atualizadas por ela.

SINTAXE GERAL

CALL <sub-rotina> (lista de argumentos)

CALL tem a finalidade de chamar uma sub-rotina em assembler compilada separadamente. Sendo que o comando CALL é utilizado para transferir o fluxo do programa para uma sub-rotina em assembler, com passagem de argumentos.
<sub-rotina> é o nome da sub-rotina para a qual se deseja transferir o controle do programa.
(lista de argumentos) contém as variáveis que serão passadas para a sub-rotina, como para retornar os resultados da operação.
A sintaxe geral do comando CALL é igual para o interpretador ou compilador, no entanto, existem diferenças operacionais entre o ambiente do programa interpretado e do compilado, cujo reflexo no uso de sub-rotinas será mostrado abaixo.
A diferenca do CALL nos dois ambientes consiste no modo de incorporar a sub-rotina ao programa e como declarar o ponto de entrada na sub-rotina.
O método de passagem de argumento é o mesmo, não exigindo duas versões da mesma sub-rotina.
O controle é passado para a sub-rotina através da instrução "CALL" da "UCP" e os argumentos são passados por referências com uso dos registradores "BC", "DE" e "HL". O retorno ao programa pode ser efetuado por uma instrução "RET".

CALL NO COMPILADOR

O compilador inclui um programa objeto de uma referência externa com nome da sub-rotina citada no comando CALL, que deverá ser satisfeita no momento da ligação do programa.
A sub-rotina deverá então declarar o ponto de entrada como um nome global(pseudo-instrução ENTRY do montador), igual ao usado no comando CALL.
EXEMPLOS: Seja uma sub-rotina "COMPAC" que compacta uma cadeia de dígitos decimais em outra cadeia em que cada dígito é armazenado em meio byte.
Um programa BASIC utiliza esta sub-rotina para compactar a variável A$ de 10 bytes na variável B$ de 5 bytes.

PROGRAMA BASIC "PRGI"

```
10 B$ = STRING$(5,CHR$(0)) ALOCA VARIAVEL B$
300 A$ = ...
310 CALL COMPAC(A$,B$)
```

Sub-rotina "COMPAC" em assembler:

```
TITLE Sub-rotina COMPAC
ENTRY COMPAC; declara COMPAC como global.

COMPAC:     ; ponto de entrada
   .
   .
   .

RET         ; retorna ao programa
END
```

A montagem da sub-rotina e a compilação e ligação do programa podem ser feitas por uma seqüência de comandos do SOL 8221 como:

```
ASM    = COMPAC    ; monta sub-rotina
CBASIC = PRG       ; compila programa
LIG PRG1, COMPAC,PRG1/N/F ; liga programa

CALL NO INTERPRETADOR
```

Para serem usadas com o interpretador BASIC, as sub-rotinas devem ser ligadas junto com um módulo que defina o ponto de carga do arquivo com sub-rotinas e os pontos de entrada de cada sub-rotina. Este módulo deve ser redigido em assembler e ter a seguinte estrutura:

```
   TITLE repertório de sub-rotinas
   ENTRY $MEMRY
   DW    $              ; define endereço de carga
$MEMRY:DS 2             ; endereço último byte
                        ; preenchido pelo ligador
;
; Lista das Sub-rotinas
;
     DC      'SUBRO1'   ; nome da sub-rotina 1
     DW      ROT1       ; ponto de entrada dela
     DC      'SUBRO2
     DW      ROT2
     .
     .
     .
     DB      -1,-1,-1   ; Tres bytes -1 definem
                        ; o fim da lista
     EXTRN ROT1, ROT2,.....
     END
```

A montagem do módulo e a criação do arquivo de sub-rotinas são feitas com o montador e ligador do SOL 8221.

```
EXEMPLO:
     ASM  =  SUBRINT  ; monta módulo
```

LIG /C:5E00, SUBINT, SUBRO1...,SUBINT/N/U/F

A chave "/C:" especifica o ponto de carga do arquivo que deve ser um endereço maior que o último byte ocupado pelo interpretador. Esse valor está sujeito a variações a cada distribuição do BASIC. Consulte o boletim de liberação do software para conhecer o valor correto para a versão do interpretador em uso.

A chave "/U" indica ao ligador que o formato do arquivo de saída é "USR".

Veja manual de utilitários para maiores informações sobre o montador e ligador do SOL 8221.

A carga das sub-rotinas para uso pelos programas pode ser feita através da opção "/S:" na inicialização do BASIC ou pelo comando DEFSUB. Neste último caso, o espaço para sub-rotinas deve ter sido reservado pela opção "/M".

EXEMPLO

```
BASIC  /S: SUBINT
BASIC  /M: 2500
DEFSUB "ROTINAS"
```

PASSAGEM DE ARGUMENTOS

Para cada variável na lista de argumentos do comando CALL é passado um endereço para a sub-rotina. O endereço é o byte menos significativo da variável se numérica, ou do descritor da variável se do tipo cadeia de caracteres.

O descritor é composto de três bytes. O primeiro contém o tamanho da cadeia e os dois seguintes formam uma palavra com o endereço do primeiro byte da cadeia.

Os endereços são passados nos registradores segundo o seguinte critério:

1. Se o número de argumentos for menor ou igual a 3, seus endereços são passados nos registradores "HL","DE" e "BC" nesta ordem.

2. Se o número de argumentos for maior que 3, então os endereços dos dois primeiros são passados em "HL" e "DE". Os endereços dos demais são passados em um bloco de palavras contíguas, cada uma contendo o endereço dos argumentos 3,4....etc. O registrador "BC" contém o endereço do primeiro byte deste bloco.

Observe que, com esse esquema, a sub-rotina deve saber quantos parâmetros esperar, a fim de encontrá-los. Inversamente, o programa que estiver chamando é responsável pela passagem do número correto de parâmetros.

Não há verificação do número correto ou tipo dos argumentos.

Se a sub-rotina esperar mais de três parâmetros e precisar transferi-los para uma área de dados local, há uma sub-rotina na biblioteca do BASIC que executa esta função. Essa rotina se chama ".TARG" e é chamada com HL apontado para a área local, BC apontando para o bloco de parâmetros e A contendo o número de parâmetros a transferir. A sub-rotina é responsável por salvar os dois primeiros parâmetros antes de chamar .TARG.

Por exemplo, uma sub-rotina que esperar cinco parâmetros, poderia ser redigida assim:

```
SUBR:
        LD      (P1),HL     ; salva parâmetro 1
        LD      (P2),DE     ; salva parâmetro 2
        LD      A,3         ; número de parâmetros restantes
        LD      HL,P345     ; aponta área local
        CALL .TARG          ; transfere parâmetro 3,4,5
        .
        .
        .
        RET                 ; retorna ao programa
P1:     DS      2           ; espaço para parâmetro 1
P2:     DS      2           ; parâmetro 2
P345    DS      6           ; parâmetro 3 a 5
        .
        .
        .
        End
```

NOTAS

1. Ao ter acesso aos parâmetros numa sub-rotina, não devemos esquecer que eles são apontadores dos argumentos reais passados.

2. Compete inteiramente ao programador verificar se os argumentos do programa que estiver chamando combinam em número, tipo e tamanho com os esperados pela sub-rotina.

3. O descritor de variável tipo cadeia não pode ser alterado. Portanto, se uma sub-rotina devolver um resultado numa variável desse tipo, o espaço para ela deve ser alocado previamente no programa principal.

VEJA TAMBÉM
USR, POKE, SYSTEM, GOSUB, RETURN

Comando/DOS **CATALOG**

SINTAXE GERAL

CATALOG [,Dn] [,Sn]
Mostra uma lista com todos os arquivos especificados.

Este comando é utilizado no DOS 3.3 dos microcomputadores que seguem a linha APPLE II para interrogar o diretório e listar os nomes dos programas arquivados em um disquete.
CATALOG é um comando do DOS e no modo programa requer PRINT e CTRL-D podendo também se indicar Dn (Drive usada), Sn (Número do Slot) ou Vn (Número do volume)

EXEMPLO Carregue o DOS e EXECUTE:
CATALOG
e observe na tela do vídeo a listagem índice dos programas armazenados no disquete.

```
DISK VOLUME 254
* A 002 HELLO
* B 003 CHAIN
* I 009 COLOR DEMO
* A 051 LISTA TELEFONICA
```

O * indica que o arquivo está protegido contra gravação.
A letra A na coluna à esquerda indica um arquivo contendo um programa em BASIC APPLESOFT.
A letra B indica um arquivo contendo um programa em BINARIO (linguagem de máquina).
A letra I indica um arquivo contendo um programa em Interger BASIC.
A letra T indica um ARQUIVO TEXTO (ou ARQUIVO DE DADOS - criado por um comando OPEN seguido de WRITE).
O número 002 que precede a primeira letra (A, neste caso) indica quantos "setores" do disquete ocupa este arquivo. Cada setor pode armazenar até 256 bytes de informação. No total, o disquete (de densidade simples) pode armazenar 496 setores de informação.
Finalmente, vem o nome do arquivo (este nome deve obrigatoriamente começar por uma letra e ter no máximo 30 caracteres).
Pode acontecer de haver um certo número de programas num disquete cuja relação de seus nomes não cabe de uma só vez na tela do vídeo. O comando CATALOG lista, inicialmente, os 18 primeiros nomes de arquivo. Para verificar os demais, pressione qualquer tecla exceto: RESET, CTRL ou SHIFT. Cada linha do catálogo representa um arquivo.

ORTOGRAFIA ALTERNATIVA
O APPLE CP/M utiliza o comando DIR para dar o catálogo dos arquivos programas. O TRSDOS (Sistema Operacional Disquete para computadores da linha TRS-80) utiliza LIB para obter o catálogo.

VEJA TAMBÉM
DIR, LIB

Função CDBL

SINTAXE GERAL

Número de linha PRINT CDBL(argumento)

CDBL é uma função que quando utilizada vem precedida por uma instrução, geralmente PRINT ou DEF.

EXEMPLO:

100 DEFDBL L-P

Em BASIC, indicará que todas as variáveis que comecem com as letras L,M,N,O e P serão de precisão dupla.

Esta função converte o argumento em um valor com precisão dupla. O valor resultante contará com 17 dígitos, mas somente aqueles contidos no argumento serão significativos.

CDBL pode ser útil, quando se quer forçar uma operação a ser efetuada em precisão dupla, mesmo que os operandos sejam de precisão simples ou ainda números inteiros.

EXEMPLO: PRINT CDBL(I%) irá retornar uma fração com 17 dígitos de precisão.

NOTA COMPLEMENTAR: CDBL é utilizado para mudar números ou variáveis numéricas do modo normal de "precisão simples" para "precisão dupla". As variáveis utilizadas na função CDBL retornam para a situação original de precisão simples se forem utilizadas novamente sem a função CDBL.

EXEMPLO: PRINT 1/CDBL(3)

Irá retornar à fração .3333333333333333

As variáveis de dupla precisão são capazes de armazenar números que contêm 17 dígitos (sendo impressos apenas 16 dígitos). As variáveis de precisão simples têm grau de precisão de seis dígitos. Deve-se ter muito cuidado para assegurar que os números utilizados para criar a resposta de dupla precisão sejam eles próprios de dupla precisão. Caso contrário, a resposta não passará de uma extensa mentira.

```
EXEMPLO 1
100 REM USANDO A FUNCAO CDBL
110 A = 5
120 B = 7
130 PRINT "CDBL MUDA A/B DE ";A/B;" PARA" CDBL(A)/CDBL(B)
140 PRINT "E DE VOLTA PARA O VALOR DE ";A/B;" AO SER REMOVIDA"
150 END

EXECUÇÃO DO PROGRAMA
RUN
CDBL MUDA A/B DE .714286 PARA .7142857142857143
E DE VOLTA PARA O VALOR DE .714286 AO SER REMOVIDA

EXEMPLO 2
200 REM USANDO A FUNCAO CDBL
210 A = 546.37
220 PRINT "A= "A
```

```
230 PRINT "CDBL(A)= "CDBL(A)
240 END
```

EXECUÇÃO DO PROGRAMA

```
RUN
A = 546.37
CDBL(A)= 546.3699951171875
```

VARIAÇÕES DE USO

Não se tem conhecimento de nenhuma.

VEJAM-SE TAMBÉM

DEFDBL, DEFSNG, DEFINT, CSNG, #, !, %, CINT

Função **CH**

SINTAXE GERAL

Número da linha PRINT CH "carácter"

CH é uma função utilizada no modo programa sendo precedida por uma instrução (geralmente PRINT).
CH é uma função reconhecida por alguns interpretadores e identifica o número ASCII do primeiro carácter de determinado string. PRINT CH "Z", por exemplo, fará exibir na tela 90 (valor ASCII de Z).
CH identifica unicamente o primeiro carácter de um string - PRINT CH "ABC" Fará exibir na tela 65 (valor ASCII de A)

OBS.: Para mais informações consulte a função ASC.

EXEMPLO 1

```
100 REM USANDO A FUNCAO CH
110 PRINT "O CODIGO ASCII PARA A LETRA B E' ";
120 PRINT CH "B"
130 IF CH "B"=66 THEN 160
140 PRINT "CH NAO FOI ACEITO"
150 GOTO 170
160 PRINT "CH FOI ACEITO"
170 END
```

EXECUÇÃO DO PROGRAMA

```
RUN
O CODIGO ASCII PARA A LETRA B E' 66
CH FOI ACEITO
```

VARIAÇÕES DE USO
Não se tem conhecimento de nenhuma.

VEJA TAMBÉM
CODE, ASC, CHR$, Código ASCII

Instrução **CHAIN**

SINTAXE GERAL

Número da linha CHAIN nomearq [,Dn] [,Sn] [,Vn]

CHAIN tem por finalidade chamar um programa, mantendo aberto os arquivos e passando-lhe variáveis do programa atual.

Em computadores que seguem a linha APPLE CHAIN nomearq [,Dn] [,Sn] [,Vn] carrega e roda um programa Integer BASIC a partir do disquete que esteja no drive especificado ou assumido sem limpar os valores de qualquer variável simples ou subscrita.

Dn,Sn, e Vn podem ser especificados em qualquer ordem. Se o número do drive (Dn) ou se o número do slot (Sn) forem omitidos, o drive referenciado por último é usado. Usa-se V0 para Vn ausente (número, volume).

EXEMPLO:

CHAIN PARTE DOIS, D1,S1,V0

(encadeamento)

Obs: CHAIN só pode ser usado com interpretador BASIC.

Mas se não precisar passar variáveis, é fácil encadear programas em APPLESOFT para carregarem e rodarem em seqüência.

Se o programa inclui uma última linha, tal qual:

3000 PRINT CHR$(4); "RUN PARTE B"

(Encadeamento em Applesoft)

Suponha que se deseje que o programa PARTE UM encadeie-se com o programa PARTE B.

Para fazer o encadeamento você deve ter o programa CHAIN no mesmo disquete (se nao tiver use o FID para copiá-lo no lugar certo) e em seguida insira estas duas linhas como sendo as últimas duas linhas a serem executadas no programa PARTE A

PRINT CHR$(4);"BLOAD CHAIN,520"

CALL 520 "PARTE B"

OBS: No comando CALL não poderá haver espaço entre o 0 do 520 e as aspas.

O CHAIN pode também ser utilizado como comando no modo direto.

EXEMPLO: CHAIN PARTE B

que irá carregar e rodar um programa em Integer BASIC chamado parte B sem limpar os valores das variáveis usadas pelo PROGRAMA PARTE A.

Para outros interpretadores BASIC que aceitam a instrução, CHAIN é utilizado para carregar na memória RAM do computador um programa novo proveniente de um dispositivo externo (tal como disco ou fita) e executar o referido programa sem comandos RUN adicionais. O programa pode ser ligado por CHAIN com qualquer outro programa, voltando inclusive ao programa inicial que pode servir de menu.

A principal vantagem da instrução CHAIN é que permite a execução consecutiva de programas inter-relacionados, de maneira automática, sem necessidade de se manter mais do que um deles no computador a determinado momento. Isto é especialmente útil onde exista um arquivo comum de DADOS armazenado externamente, passível de acesso e manipulação por programas no CHAIN. O CHAIN

encontra a sua melhor aplicação nos sistemas suficientemente grandes para contarem com armazenamento em disco, com tempos de acesso reduzidos.

Se os valores das variáveis tiverem que ser transmitidos de um programa para outro, deve ser criado para os mesmos um arquivo separado.

Antes que se permita ligar tal programa com CHAIN a outro, o mesmo deverá salvar os valores de suas variáveis no referido arquivo para que o programa NOVO as possa ler de volta antes de sua própria execução.

Algumas linguagens BASIC têm poder de passar diretamente para o segundo programa os valores das variáveis utilizadas pelo primeiro programa.

O BASIC Microsoft 5.0, por exemplo, aceita a instrução CHAIN "PROG2", 150, ALL, que se "encadeia" com um programa em disco denominado PROG2, o PROG2 inicia a execução na linha 150 depois de receber os valores de todas as variáveis que tenham sido definidas antes pelo programa de chamada.

O nome do programa novo deve ser incluído após a instrução CHAIN. Certos computadores especificam o número da linha inicial do programa novo por um número que segue o nome do programa. Sendo omitida a linha inicial, o computador começará automaticamente no início do programa novo.

10 CHAIN TEST,30, por exemplo determina ao computador para apagar o programa que está na memória e para carregar um programa chamado "TEST", proveniente de um dispositivo externo, começando a execução na linha 30 deste. O dispositivo externo de armazenamento pode ser especificado em determinados computadores (tais como na linguagem BASIC de DEC 10), por se colocar o nome do dispositivo depois de CHAIN, sendo seguido o mesmo por dois pontos.

Considere, por exemplo, 10 CHAIN PTR:TESTE,160, que determina que o computador carregue um programa denominado "TESTE" da Leitora de Fita de Papel, começando a execução na sua linha 160.

PROGRAMAS DE TESTE

Conserve este programa em disco ou fita magnética sob o nome "TESTE",

```
100 REM PROGRAMA *TESTE*
110 PRINT "ESTA' SENDO EXECUTADO AGORA O PROGRAMA TESTE"
120 FOR A = 1 TO 9
130 PRINT A;
140 NEXT A
150 PRINT "ESTE PROGRAMA DEVOLVE AGORA CHAIN DE VOLTA PARA O
PROGRAMA PRINCIPAL"
160 CHAIN PRINCIPAL, 130
170 END
```

Dê entrada agora no programa principal no computador e salve-o em disco ou fita sob o nome "PRINCIPAL"

```
200 REM PROGRAMA *PRINCIPAL*
210 PRINT "ESTE PROGRAMA DEVE CARREGAR E RODAR O PROGRAMA
TESTE
220 CHAIN TESTE
```

```
230 PRINT "CHAIN FOI APROVADO SE O PROGRAMA TESTE"
240 PRINT "IMPRIMIU UMA SERIE DE NUMEROS"
250 END
```

Prepare seu disco ou sua(s) fita(s) para serem lidos a
do e em seguida RUN (rode).

```
EXECUÇÃO DO PROGRAMA
RUN
ESTE PROGRAMA DEVE CARREGAR E RODAR O PROGRAMA TESTE
O PROGRAMA TESTE ESTA  SENDO RODADO AGORA
  2 3 4 5 6 7 8 9
STE  PROGRAMA  DEVOLVE  AGORA CHAIN DE VOLTA PARA O PROGRAMA
PAL
HAIN FOI APROVADO SE O PROGRAMA TESTE
MPRIMIU UMA SÉRIE DE NÚMEROS
```

e O SEU COMPUTADOR NÃO TIVER ESTA INSTRUÇÃO
operação CHAIN pode ser realizada com êxito, embora mais
nte, mediante a utilização de discos flexíveis ou até
s.
maioria dos pequenos sistemas BASIC começam cada programa
om CHAIN para o número da primeira linha, não tendo opção
meçar em qualquer outro ponto.
três programas seguintes demonstram como a operação CHAIN
er realizada num sistema de disco com TRS-80. A palavra
utilizada em vez de CHAIN. Observe que o nome do programa
encontra entre aspas.

```
1 ' REM PROGRAMA *PRINCIPAL*
110 PRINT
120 PRINT "ESTE E  O PRINCIPAL PROGRAMA  DE CONTROLE"
130  INPUT  "VAMOS  CHAIN PARA O PROGRAMA = 1 OU = 2 (TECLE 1
OU 2)";I
140 PRINT "FIQUE NA ESPERA PARA CARREGAR---"
150 ON I GOTO 160,170
160 RUN "TESTE1"
170 RUN "TESTE2"
180 END

100 REM PROGRAMA *TESTE1*
110  PRINT  "PROGRAMA  DE  TESTE  NUMERO 1 ESTA  SENDO RODADO
AGORA"
120 FOR A = 1 TO 9
130 PRINT "UM"
140 NEXT A
150 PRINT
160  PRINT  "VAMOS  AGORA  CHAIN  DE  VOLTA  PARA  O PROGRAMA
PRINCIPAL---"
170 RUN "PRINCIPAL"
180 END


100 REM PROGRAMA *TESTE2*
110  PRINT  "PROGRAMA  DE  TESTE  NUMERO 2 ESTA  SENDO RODADO
AGORA"
120 FOR A = 1 TO 9
130 PRINT "DOIS"
```

```
140 NEXT A
150 PRINT
160 PRINT "VAMOS AGORA . CHAIN DE VOLTA PARA O PROGRAMA
PRINCIPAL---"
170 RUN "PRINCIPAL"
180 END
```

Tecle cada programa e conserve em disco sob o nome dado em cada linha 100. Em seguida RUN (rode) o programa PRINCIPAL. Fique observando enquanto a tela e o(s) dispositivo(s) de disco executam o CHAIN dos programas entre si e os executam.

VARIAÇÕES DE USO

Não se tem conhecimento de outras.

VEJA TAMBÉM

CLOAD, CSAVE, COMMON, RUN

Instrução **CHANGE**

SINTAXE GERAL

Número de linha CHANGE A$ TO A

CHANGE é uma instrução reconhecida por alguns interpretadores BASIC e utilizada para converter strings de caracteres para seus números ASCII, e reconverter os números de volta para strings de caracteres.

CHANGE A$ TO A converte cada carácter do string A$ para seu número ASCII, armazenando-os todos num endereço A. O LENgth (comprimento) do string LEN(A$) está armazenado no elemento A(0) do arranjo.

CHANGE A para A$ converte os valores armazenados no arranjo A para um string de caracteres A$. O número de valores a ser convertido se encontra em A(0). Cada valor de A deve ficar entre 0 e 255.

EXEMPLO 1

```
100 REM USANDO CHANGE A$ TO A
110 DIM A(8)
120 A$ = "SULLIVAN"
130 CHANGE A$ TO A
140 PRINT "CHANGE CONVERTEU SULLIVAN PARA "
150 FOR N = 1 TO A(0)
160 PRINT A(N);" ";
170 NEXT N
180 END
```

EXECUÇÃO DO PROGRAMA

```
RUN
CHANGE CONVERTEU SULLIVAN PARA
83 85 76 76 73 86 65 78
```

EXEMPLO 2

```
200 REM USANDO CHANGE A TO A$
210 DIM A(3)
220 READ A(0)
230 FOR I = 1 TO A(0)
240 READ A(I)
250 NEXT I
260 CHANGE A TO A$
270 PRINT "CHANGE ";A$;" ";" APROVADO"
280 DATA 3,70,79,73
290 END
```

EXECUÇÃO DO PROGRAMA

```
RUN
CHANGE FOI APROVADO
```

Se O SEU COMPUTADOR NÃO TIVER ESTA INSTRUÇÃO

CHANGE pode ser simulado utilizando-se LEN, ASC, MID$ e CHR$.

No exemplo 1, faça a seguinte substituição na linha 130:

```
130 A(0) = LEN(A$)
131 FOR I = 1 TO A(0)
132 A(I) = ASC(MID$(A$,I,I))
133 NEXT I
```

No exemplo 2, faça a seguinte substituição na linha 260:

```
260 A$ = ""
261 FOR I = 1 TO A(0)
262 A$ = A$ + CHR$ (A(I))
263 NEXT I
```

NOTA

Para os microcomputadores que seguem a linha APPLE deve-se colocar aspas vazias como na linha 160 do exemplo 1 e na linha 260 do exemplo 2, para que ocorra o espaçamento entre os números no primeiro exemplo e espaçamento entre as palavras no exemplo 2.

VEJA TAMBÉM

LEN, ASC, MID$, CHR$, DIM.

Consulte ainda a tabela ASCII.

Função CHR$ • CHAR
CHAR$ • CHR

SINTAXE GERAL

Numero da linha PRINT CHR$ (exprnm)

CHR$(N) é uma função normalmente utilizada no modo programa, sendo precedida por uma instrução geralmente PRINT ou LET.

Esta função CHR$(N) funciona de maneira idêntica tanto nos microcomputadores que seguem a linha APPLE II como nos que seguem a linha TRS-80 fazendo retornar o carácter equivalente ao número N ou expressão N, utilizando o código ASCII. Contudo, nos computadores que seguem a linha SINCLAIR, o carácter retornado equivale ao padrão interno do equipamento, e não ao padrão ASCII.

EX: PRINT CHR$(7)

Em um programa para computadores que seguem a linha APPLE II o CHR$(7) provocará um "bip" no alto-falante do microcomputador não havendo equivalente direto para este exemplo na linha do TRS-80.

CHR$(N) efetua a conversão de um (argumento) valor decimal de (N) para o símbolo representativo deste código.

O argumento pode ser qualquer número inteiro de 0 a 255 ou qualquer expressão variável com um valor nesta faixa.

Nos sistemas de computação e, principalmente, de transmissão de dados a distância, todas as letras, sinais e caracteres especiais têm um código numérico equivalente. A tabela de códigos mais usada em microcomputadores é a denominada ASCII (American Standard Code for Information Interchange). A armazenagem dos caracteres no computador é feita através do código. Assim, quando o computador armazena qualquer DADO na memória, na realidade ele está armazenando os códigos numéricos correspondentes às letras, aos números e aos sinais (inclusive espaço).

OBS: A função CHR$(N) é usada para gerar caracteres que não se conseguem obter a partir do teclado para controle de periféricos etc.

O código de números decimais incluído entre parênteses após CHR$ pode ser representado por um número ou uma variável dentro da gama do código ASCII (tipicamente de 0 a 127). Entretanto a maioria dos computadores têm código ASCII ampliado (até 255) que inclui habilidades especiais e caracteres de gráficos. Muitos computadores utilizam certos números ASCII para finalidades especiais "não-padronizadas" (tipicamente para controles especiais como: impressora de linha, unidades de disquetes, apagar tela etc.)

CHR$(N) aplica-se a um número, e dá o carácter simples, cujo código é esse número.

O exemplo a seguir permite ver todos os caracteres ASCII que forem imprimíveis.

```
100 REM USANDO A FUNCAO CHR$
110 FOR X = 0 TO 255
120 PRINT X;
130 PRINT CHR$(X)
140 FOR T = 1 TO 400
150 NEXT X
160 NEXT T
170 END
```

Experimente executar este programa no seu computador para verificar toda a gama dos códigos ASCII dele.

NOTA: É possível imprimir aspas usando a instrução PRINT seguida da função CHR$(34)

EXEMPLO: PRINT CHR$(34) "SULLIVAN" CHR$(34)

Os códigos de 0 a 31 sao utilizados como códigos de controle, por isso, quando se usam estes códigos com CHR$, obtém-se a função efetuada por aquele código de controle.

CHR$ realiza o inverso da função ASC, isto é, fornece um STRING de um carácter. Carácter este que tenha um código gráfico de controle ou ASCII.

ORTOGRAFIAS ALTERNATIVAS

Algumas versões de BASIC apresentam ortografias diferentes em lugar de CHR$, assim como: CHR, CHAR e CHAR$.

VARIAÇÕES DE USO

A versão BASIC da MAXBASIC utiliza CHAR(L1,L2) e requer dois números, sendo o primeiro o Código ASCII, ao passo que o segundo indica quantos caracteres devem ser gerados.

EXEMPLO: CHAR(79,1) é o equivalente a CHR$(79)

CHAR(66,4) = BBBB

VEJA TAMBÉM

STRING, ASC, Código ASCII

Função **CINT**

SINTAXE GERAL

Número da linha INSTRUÇÃO CINT(argumento)

CINT(X) é uma função usada normalmente no modo programa devendo ser precedida de uma instrução geralmente PRINT ou LET.

A função CINT(X) é reconhecida por alguns interpretadores BASIC e utilizada para converter a seu valor de número inteiro o seu argumento (X) sendo que este pode estar representado por números individuais ou variáveis numéricas.

As variáveis utilizadas na função CINT retornam a sua precisao original se forem utilizadas novamente sem a função CINT.

A função CINT(X) fornece o maior inteiro menor que o argumento. Os números são sempre arredondados.

EX: PRINT CINT(2.5) RESULTA EM 2
PRINT CINT(-2.5) RESULTA EM -3

CINT(X) retorna o maior número inteiro, desde que não seja maior que o argumento.

A maioria dos computadores não permite que os números atribuídos à função CINT sejam menores do que -32768 ou maiores do que +32767.

A função INT é muito semelhante, não estando restrita, contudo, a uma gama de números tão limitados.

EXEMPLO 1

```
100 REM USANDO A FUNCAO CINT
110 DEFDBL R
120 R = 23145.7678
130 PRINT "CINT MODIFICA O VALOR DE R DE "; R; "PARA "; CINT(R)
140 PRINT "E VOLTA AO VALOR DE "; R; "QUANDO REMOVIDA"
150 END
```

EXECUÇÃO DO PROGRAMA

```
RUN
CINT MODIFICA O VALOR DE R DE 23145.7678 PARA 23145 E VOLTA AO VALOR DE 23145.7678 QUANDO REMOVIDA
```

VARIAÇÕES DE USO

A função CINT(X) poderá ser utilizada para acelerar uma operação envolvendo operandos de simples ou dupla precisão, sem perder a exatidao dos mesmos (presumindo que se esteja interessado em obter um resultado de números inteiros).

EXEMPLO: 200 P% = CINT(X#) + CINT(Y#)

VEJA TAMBÉM

DEFINT, INT, DEFSNG, DEFDBL, CDBL, CSNG, !, #, %

Comando/Instrução **CLEAR M**
CLEAR • CLR

SINTAXE GERAL

Número da linha CLEAR
ou
CLEAR variável numérica
ou
simplesmente CLEAR

CLEAR pode ser utilizado como comando no modo direto ou como intrução no modo programa.

A instrução CLEAR funciona de maneira aproximadamente idêntica nos computadores que seguem as linhas APPLE II, SINCLAIR ou TRS-80 reduzindo todas as variáveis numéricas a zero e todas as variáveis alfanuméricas a "nulo". Em computadores que seguem a linha TRS-80 pode se tornar necessário executar um CLEAR X em que X deve ser um número <>0 o qual indica em bytes o espaço a ser reservado em memória para STRING, porém para os da linha APPLE ou SINCLAIR não é necessário.

CLEAR ou também CLR, quando usado o Integer BASIC do APPLE, coloca todas as variáveis alfanuméricas com o conteúdo "NULO" (não é o mesmo que espaço) e zera todas as variáveis numéricas.

A Instrução CLEAR zera todas as variáveis numéricas e atribui às variáveis alfanuméricas ('STRING') um valor nulo. No entanto se a sintaxe utilizada for CLEAR(número) ou CLEAR(variável) além de inicializar todas as variáveis, o sistema operacional reserva para utilização em 'STRINGS' a quantidade de bytes indicada pelo número ou pela variável.

EXEMPLOS:

CLEAR 50 teremos 50 bytes para 'STRINGS'

I = 50.5 CLEAR I teremos 50 bytes para 'STRINGS'.

A reserva é feita para o conjunto de todas as variáveis alfanuméricas que estejam sendo utilizadas, em programa ou em modo direto, pois, mesmo depois da execução de um programa, os computadores que seguem a linha do computador TRS-80 continuam reservando para 'STRINGS' a quantidade de bytes indicada no último CLEAR encontrado no programa.

O BASIC MICROSOFT versão 5.0, ou posterior, não admite esta Sintaxe, pois a alocação de memória para 'STRING' é feita dinamicamente.

A Sintaxe CLEAR X = Y onde X e Y são valores ou variáveis é interpretada como CLEAR 0 nos computadores DGT-100, CP-500, FENIX e outros que seguem a linha TRS-80.

CLEAR no minicomputador LABO tem como finalidade estabelecer todas as variáveis numéricas como zero e todas as variáveis cadeia como nulas; e, opcionalmente, estabelecer o fim da memória e a quantidade de espaço de memória para a Pilha.

OBSERVAÇÕES: Quanto ao uso do CLEAR, é conveniente aplicá-lo com cuidado, pois em alguns raros computadores CLEAR tem outro significado; é idêntico a NEW, apagando variáveis e também programa.

Em computadores que seguem a linha inglesa do SINCLAIR, o efeito do CLEAR é de desmemorizar todo o espaço acumulado que foi reservado para variáveis - é como se as variáveis nunca tivessem sido definidas. Ao desligar e ligar o computador isto também acontecerá, mas então ele não se lembrará de nada quando voltar a ligá-lo.

As expressões podem conter o nome de uma variável onde quer que possam ter um número.

CONSIDERAÇÕES GERAIS

Quando o comando CLEAR for utilizado sem um argumento (digite CLEAR e pressione a tecla ENTER) ele anula todas as variáveis strings; quando usado acompanhado de um argumento (por exemplo, CLEAR 500), este comando realizará uma segunda função além desta descrita; isto torna um número determinado de bytes disponíveis para armazenagem do string. No exemplo, CLEAR 500 còloca 500 bytes disponíveis para strings.

Quando ligar o computador, um CLEAR 50 foi executado automaticamente.

```
EXEMPLO 1
100 REM USANDO A INSTRUCAO CLEAR
110 A = 500
120 A$ = "STRING DE TESTE"
130 PRINT "ANTES DA INSTRUCAO CLEAR A =";A
140 PRINT "E VARIAVEL DE STRING A$ = ";A$
150 CLEAR
160 PRINT "APOS EXECUTAR A INSTRUCAO CLEAR A= ";A
170 PRINT "E VARIAVEL DE STRING A$ = ";A$
180 END

EXECUÇÃO DO PROGRAMA
RUN
ANTES DA INSTRUCAO CLEAR A = 500
E VARIAVEL DE STRING A$ = STRING DE TESTE

APOS EXECUTAR A INSTRUCAO CLEAR A = 0
E VARIAVEL DE STRING A$=

EXEMPLO 2
200 REM USANDO A INSTRUCAO CLEAR X
210 X = 47
220 A$ = "SULLIVAN"
230 PRINT A$
240 CLEAR
250 PRINT X
260 END

EXECUÇÃO DO PROGRAMA
RUN
SULLIVAN
0
```

ORTOGRAFIAS ALTENATIVAS

Tanto os microcomputadores que seguem a linha do APPLE II como PET utilizam CRL como ortografia alternativa de CLEAR.

VARIAÇÕES DE USO

Alguns computadores utilizam CLEAR como instruçao especial para limpar as memórias intermediárias de entrada ou saída do terminal. Os computadores WANG utilizam CLEAR como comando apenas. Ao ser utilizado sozinho CLEAR apaga todas as linhas do programa e variáveis na memória. CLEAR1 funciona como NEW ou SCRATCH em outros computadores. CLEAR P remove as linhas do programa, mas deixa ficarem as variáveis. CLEAR Pn1, n2 irá remover as linhas do programa com números de linha entre n1 e n2. Sendo deixado fora n2, irá apagar todas as linhas do programa de n1 em diante. CLEAR V remove da memória unicamente as variáveis, ao passo que CLEAR N remove apenas as variáveis não comuns.

No BASIC-Microsoft 5.0 (BASIC-80), CLEAR nao reserva qualquer espaço para strings, mas proporciona a opção de reservar espaço no topo da memória. CLEAR 32000, por exemplo, poe em 0 todas as variáveis numéricas e reduz as variáveis de string para valor nulo, ao passo que reserva a memória além da linha 32000 para programas em linguagem de máquina.

CLEAR COMO TECLA ESPECIAL:

Em computadores cujos teclados possuem a tecla CLEAR, ao ser pressionada, limpa a tela retornando o cursor para a primeira posição da primeira linha. Seu código ASCII é 31. Utilizando a tecla CLEAR as memórias nao sao zeradas.

Obs.: O comando PRINT CHR$(31) não desempenha o papel do CLEAR.

VEJA TAMBÉM

FRE(A$), COMMON, NEW, SCRATCH

Função CLG • CLOG

SINTAXE GERAL

Número da linha LET variável = CLG(número)

A função CLG(n), reconhecida pelos interpretadores BASIC da HONEYWELL, é utilizada para calcular o valor do logaritmo comum (base 10) de qualquer número (n) cujo valor seja superior a 0.

```
EXEMPLO 1
100 REM TESTANDO A FUNCAO CLG
110 PRINT "DE ENTRADA NUM NUMERO POSITIVO"
120 INPUT R
130 L = CLG(R)
140 PRINT "O LOGARITMO BASE 10 DE ";R; "E"; L
3999 END
```

```
EXECUÇÃO DO PROGRAMA
RUN
DE ENTRADA NUM NUMERO POSITIVO
? 100
O LOGARITMO BASE 10 DE 100 E 4.60517019
```

ORTOGRAFIA ALTERNATIVA
Em alguns microcomputadores CLOG é utilizado em vez de CLG.

SE O SEU COMPUTADOR NÃO TIVER ESTA FUNÇÃO
Se o seu computador falhou no exemplo 1, experimente os exemplos contidos em LOG10 e LOG. Se esses também falharem, use a seguinte sub-rotina:

```
3150 REM *SUBROTINA DE LOGARITMOS COMUNS ENTRADA R, SAIDA L
3197 L = L * 0.4342945
3200 RETURN
3154 IF X > 0 THEN 3160
3156 PRINT "LOGARITIMO NAO DEFINIDO EM ";X
3158 STOP
3160 P = 1
3162 M = 2
3164 N = .5
3166 T = X
3168 S = 0
3170 IF X < P THEN 3178
3172 X = N*X
3174 S = S+P
3176 GOTO 3170
3178 IF X >= N THEN 3186
3180 X = M*X
3182 S = S-P
3184 GOTO 3178
3186 X = (X-.707107)/(X+.707107)
3188 L = X*X
3190 L = (((.598979*L+.961471)*L+2.88539)*X+S-.5)*.693147
```

```
3192 IF L > 1S-6 THEN 3196
3194 L = 0
3196 X = T
3198 RETURN
```

Para utilizar esta sub-rotina com o programa do Exemplo 1, altere a linha 130 para: 130 GOSUB 3150

FATORES DE CONVERSÃO

Para converter um logaritmo comum para logaritmo natural, multiplique o valor do logaritmo comum por 2.302585.

EXEMPLO:

```
X = CLG(R)*2.302585
```

Para converter logaritmo natural para logaritmo comum, multiplique por 0.3432945 o valor do logaritmo natural.

VARIAÇÕES DE USO

Não se tem conhecimento de nenhum.

VEJA TAMBÉM

LOG10, LOG

Função **CLK$ • CLK**

SINTAXE GERAL

Número da linha PRINT CLK$

CLK$ é utilizado apenas como função no modo programa, sendo precedido por um comando direto PRINT ou uma instrução.

CLK$ é utilizado com instruções PRINT em algumas versões BASIC, assim como o BASIC-PLUS 2 para indicar a hora em: horas (0 a 24), minutos, e segundos (hh:mm:ss). O computador insere automaticamente um "dois pontos" após os valores das horas e dos minutos e imprime a hora como string.

PRINT CLK$, por exemplo, imprimirá uma indicação semelhante a 20:12:15, indicando que a hora é 8:12 da noite mais 15 segundos.

```
EXEMPLO 1
100 REM USANDO A FUNCAO CLK$
110 PRINT "O SEU RELOGIO ESTA  MARCANDO EXATAMENTE ";
120 PRINT CLK$
130 PRINT "SENDO IMPRESSO UM NUMERO DE SEIS DIGITOS CLK$
FUNCIONOU"
140 END
```

```
EXECUÇÃO DO PROGRAMA
RUN
O SEU RELOGIO ESTA  MARCANDO EXATAMENTE 12:10:15
SENDO IMPRESSO UM NUMERO DE SEIS DIGITOS CLK$ FUNCIONOU
```

ORTOGRAFIAS ALTERNATIVAS

O CP-500 da PROLOGICA utiliza $CLKON no endereço 664 para chamar uma sub-rotina que ative o relógio no vídeo e utiliza $CLKOFF no endereço 673 para chamar outra sub-rotina que desativa o relógio.

CLK (n) é utilizado no BASIC da Sperry Univac system/9 para indicar a hora (hhmmss). Requer-se um expressão numérica (entre parênteses) após CLK, embora não exerça nenhum efeito sobre a função CLK. Mude a linha 120 do programa no Exemplo 1 para 120 PRINT CLK(0), e rode seu programa para ver se seu computador aceita CLK.

VARIAÇÕES DE USO

Não se tem conhecimento de nenhuma

VEJAM TAMBÉM

TIME, TIME$

Comando **CLOAD**

SINTAXE GERAL

CLOAD nome do arquivo programa

CLOAD é um comando usado no modo direto.

CLOAD é utilizado em alguns computadores, principalmente naqueles que seguem a linha do TRS-80, para transferência de (carga) de programa de fita cassete para a memória do computador.

Na aplicação deste comando é necessário ler atentamente o manual do aparelho utilizado, pois as diferenças de um para outro computador são grandes.

CLOAD nome do arquivo

O comando CLOAD possibilita a carga de um programa armazenado em cassete.

Coloque o gravador no modo PLAY (certifique-se de que as ligações corretas foram feitas e a fita rebobinada à posição correta).

O nome de arquivo pode ser qualquer caracter simples, exceto aspas.

Se introduzirmos o comando CLOAD, a máquina ligará o gravador e copiará o primeiro programa encontrado.

O BASIC também permite que especifique o nome do arquivo desejado, em seu comando CLOAD. Por exemplo, CLOAD "A" fará com que o computador ignore os programas em cassete, até que ele encontre o programa rotulado "A" localizado na fita. Quando o encontrar será separado entre todos os outros e copiado.

Enquanto o computador estiver procurando o arquivo "A", os nomes dos arquivos encontrados aparecerão no canto superior direito da tela, além de um "*" intermitente. Somente o primeiro carácter do nome do arquivo é utilizado pelo computador para CLOAD, CLOAD? e operações CSAVE, CLOAD também possibilitam que copie um programa da fita e limpe automaticamente o programa armazenado anteriormente no computador.

CLOAD? "NOME DO ARQUIVO"

Permite a comparação de um programa armazenado no cassete com um no computador. Isto lhe será útil quando gravar um programa na fita usando CSAVE, e desejar certificar-se de que a transferência foi bem sucedida. Você deve especificar CLOAD? "NOME DO ARQUIVO" e, se isso não for feito, o primeiro programa encontrado será testado. Durante o CLOAD?, o programa na fita e o programa na memória são comparados byte por byte. Se houver alguma discrepância (indicando uma gravação malfeita) a mensagem "MAL" aparecerá na tela. Neste caso, você deverá gravar (C SAVE) o programa novamente. (CLOAD ?, ao contrário de CLOAD, não apaga a memória do programa).

Certifique-se de digitar o ponto de interrogação, ou o computador interpretará seu comando como CLOAD.

EXEMPLO 1

Dê entrada neste programa ao computador pelo teclado, e em seguida em fita cassete. (Constam em CSAVE os respectivos pormenores)

```
100 REM PROGRAMA PARA VERIFICAR USO CLOAD
110 PRINT "ENTRE COM VALOR CREDITO"
120 INPUT C
130 PRINT "ENTRE COM VALOR DEBITO"
140 INPUT D
150 PRINT "CREDITO","DEBITO","SALDO"
160 S = C - D
170 PRINT C,D,S
180 END

EXECUÇÃO DO PROGRAMA
RUN
ENTRE COM VALOR CREDITO
?10
ENTRE COM VALOR DEBITO
?4
CREDITO     DEBITO     SALDO
10          4          6
```

Uma vez gravado o programa em fita cassete, apague a memória do computador com os comandos NEW, SCRATCH ou outro apropriado.

Rebobine a fita, coloque o gravador em seguida no modo "PLAY" e tecle o comando CLOAD.

O motor do gravador cassete será controlado pelo computador, que o liga e desliga antes e depois do ciclo "carregar" (LOAD). O gravador cassete deve 'tocar' o programa efetuando o LOAD (carregamento) ao computador.

Faça a listagem do programa para se certificar de que o programa armazenado agora na memória do computador é igual àquele a que se deu entrada inicialmente (ver LIST). Parecendo que tudo está bem, execute o programa (RUN).

O "nome programa" CLOAD é utilizado por alguns computadores equipados com CLOAD para carregar no cassete apenas aqueles programas que tiverem nome de programa coincidente. O nome de programa utilizado para identificar determinado programa em especial talvez contenha mais de uma letra ou de um número, sendo possível, contudo, que o computador reconheça unicamente o primeiro carácter.

Registre o PROGRAMA DE TESTE na cassete, utilizando CSAVE "A" (ver CSAVE), apague a memória do computador, e em seguida carregue novamente "A" ao computador, mediante a utilização de "CLOAD". Faça a listagem do programa, para verificar a possível presença de erros.

O "nome programa" CLOAD? é utilizado por alguns computadores equipados com CLOAD para comparar um programa armazenado na memória do computador com outro, gravado na cassete sob o nome de programa assinalado.

O computador efetua a comparação bit-por-bit dos dois e imprime mensagem de erro se encontrar alguma diferença. Isto permite comparar a fita com o conteúdo da memória para ter certeza de que executou CSAVE bem-sucedido, ou CLOAD perfeito, antes que apague um outro.

Experimente o programa exemplo 1 com fita cassete (armazenamento com nome de programa "A"). Não sendo impressa mensagem de erro, é que os dois programas coincidiram.

Adicione esta linha ao programa exemplo 1 armazenado no computador.

175 REM LINHA EXTRA

Verifique novamente o programa "A" em fita cassete, utilizando o comando CLOD ? "A". Deve-se imprimir uma mensagem de erro, sinal de que o computador encontrou uma diferença entre o programa armazenado no computador e o programa conforme armazenado na fita.

NOTA:

CLOAD* (nome de arranjo) é utilizado por alguns computadores que empregam CLOAD como comando para carregar um arranjo armazenado em fita cassete (sob o mesmo nome de arranjo). CLOAD*A, por exemplo, significa "carregue arranjo A".

CLOAD* (nome de arranjo) pode também ser utilizado como instrução de programa, para que os dados de arranjo possam ser carregados à medida que se execute um programa.

VARIAÇÕES DE USO

LOAD, LOAD "X"

Nos microcomputadores que seguem a linha APPLE, usa-se o comando LOAD, e nos que seguem a linha SINCLAIR usa-se o comando LOAD "X". Este comando carrega um arquivo na área de programa do equipamento.

CLOAD "X" NENHUM - Nos micros que seguem a linha TRS-80 o nome do ARQUIVO "X" é opcional.

Se omitido, tal como acontece nos micros que seguem a linha APPLE, gera carregado o primeiro ARQUIVO encontrado na fita. Tanto nos micros da linha TRS-80 como da linha SINCLAIR, será utilizada apenas a primeira letra do nome fornecido.

VEJA TAMBÉM

CSAVE, LIST, CHAIN, RECALL, APPEND,LOAD

Comando **CLOCK**

SINTAXE GERAL

CLOCK [chave]

Ativa visualização do relógio.

Chave fornece duas opções [ON-OFF] ao DOS dos microcomputadores que seguem a linha TRS-80 de modelo 1 e III, assim como o CP-500, DGT 1000 e outros.

Na ocasião da opção, será considerada a condição ON (ligado).

Este comando controla a visualização da hora real no canto superior direito da tela.

Se ativado, a hora-minuto-segundo será mostrada em ciclos de 24 horas e haverá um incremento a cada segundo, não importando o programa em execução.

A visualização do relógio está desativada na hora do acionamento do sistema operacional para disquete (DOS).

O relógio de tempo real está sempre funcionando, não importando se sua visualização está ativa ou não (exceção feita durante a entrada e saída de disco e cassete).

EXEMPLOS:

CLOCK
Ativa a visualização do relógio.

CLOCK (ON)
Ativa a visualização do relógio.

CLOCK (OFF)
Desativa a visualização do relógio.

VEJA TAMBÉM
TIME

Comando/DOS **CLOSE • $CLOSE**

SINTAXE GERAL

A Sintaxe do comando CLOSE, varia de computador para computador e depende do sistema operacional utilizado (DOS).

CLOSE desloca o buffer usado pelo arquivo em disquete especificado, e se a operação com disquete foi de escrita (gravação), guarda a informação restante do buffer no disquete.
CLOSE tem por finalidade concluir operações de entrada e saída em um arquivo de disco. Nos computadores que seguem a linha APPLE II(Americano) como MAXXI, da Polymax, MICROENGEHO, AP II da Unitron, Dactron e outros.

SINTAXE GERAL
CLOSE [nomearq]... se for utilizado no modo direto ou número da linha PRINT CHR$(4); "CLOSE nome do arquivo"

CLOSE é um comando do DOS nos microcomputadores que seguem a linha APPLE e requer PRINT e CTRL-D sendo usado no modo programa.
CLOSE f - fecha o arquivo especificado. CLOSE é usado depois que o programa acaba de usar o arquivo de dados e antes de encerrar o programa.
CLOSE fecha um ou mais arquivos de dados.
CLOSE completa qualquer trabalho dentro do sistema do computador no qual o BUFFER foi envolvido. Pode ser usado de modos diferentes.
EXEMPLOS:

```
PRINT CHR$(4);"CLOSE NOME DO ARQUIVO"
PRINT CHR$(4);"CLOSE";F$
PRINT CHR$(4);"CLOSE"
```

O comando CLOSE desloca a área de trabalho (o "BUFFER DE ARQUIVO" alocado para o arquivo especificado "f"). Se f não for especificado, todos os arquivos abertos serão fechados, com exceção daquele usado pelo comando EXEC.
Note agora que o comando CLOSE nao possui parâmetros para especificar o número do acionador e do conector.
Por exemplo:

```
PRINT CHR$(4);"CLOSE FICHARIO"
```

faz com que sejam fechados todos os arquivos chamados FICHARIO, não importando o acionador, tampouco a posição do controlador de disquete.
Da mesma forma o comando CLOSE fecha todos os arquivos de todos os disquetes (exceto um arquivo usado pelo comando EXEC).

EM COMPUTADORES QUE SEGUEM A LINHA DO TRS-80 como CP-500, DGT-100 e outros:
CLOSE fecha o acesso ao arquivo

SINTAXE GERAL
Número da linha CLOSE nmexp nmexp,...

nmexp é um valor de 1 a 15 e indica o número do buffer do arquivo (determinado ao se abrir o arquivo). Se nmexp for omitido, todos os arquivos abertos serão fechados.

Esse comando termina o acesso a um arquivo, através de um ou mais buffers especificados.

Se um exp não for determinado numa instrução OPEN anterior, então CLOSE nmexp não surtirá efeito.

Se nmexp for omitido, todos os arquivos abertos serão fechados.

Este comando termina o acesso a um arquivo, através de um ou mais buffers especificados.

Se um exp não for determinado numa instrução OPEN anterior, então CLOSE nmexp não surtirá efeito.

EXEMPLO:

CLOSE 1,2,8

Cancela as especificações de arquivo dos buffers 1,2,8. Esses buffers podem ser agora designados a outros arquivos, utilizando instruções OPEN.

CLOSE ARQ 01% + CONT%

Cancela a especificação de arquivo do buffer especificado pela soma (1% + CONT%).

OBS.: Nao retire um disquete que contenha um arquivo aberto para gravação (modo O, E ou R). Primeiro, feche o arquivo.

Isto é devido a impossibilidade de se escrever os últimos 256 bytes de dados no disco.

O fechamento de arquivo registrará os dados , se não o tiver feito.

Qualquer modificação no programa residente (NEW, edição, LOAD, MERGE, etc.) farão com que os arquivos abertos sejam fechados.

NO COMPUTADOR LABO

CLOSE tem a finalidade de concluir operações de entrada e saída em um arquivo de disco.

SINTAXE GERAL

Número da linha CLOSE # <número de arquivo> , # número de arquivo...>

Observações: <número de arquivo> é o número sob o qual o arquivo foi aberto.

Um CLOSE sem nenhum argumento fecha todos os arquivos abertos.

A associação entre um determinado arquivo e um número de arquivo termina quando da execuçao de um CLOSE. Esse número de arquivo pode agora ser reutilizado para abrir (OPEN) qualquer arquivo.

Um CLOSE para um arquivo seqüencial de saída escreve o BUFFER final de saída.

Os comandos END e NEW sempre fecham (CLOSE) todos os arquivos de discos automaticamente. (STOP não fecha os arquivos de disco)

ORTOGRAFIA ALTERNATIVA

$CLOSE é também utilizado em sistema operacional de disquete (DOS) de computadores como CP-500.

, $CLOSE fecha um arquivo depois que o processamento termina. É muito importante fazer um $CLOSE em todo arquivo aberto, antes de o programa terminar.

Se o arquivo não for fechado, a entrada do diretório para este arquivo ficará incorreta se algum registro for feito nele. Outros casos podem ocorrer, mas não serão discutidos aqui; porém é muito importante para o DOS-500 que todos os cuidados sejam tomados para gerenciamento dos arquivos.

VEJA TAMBÉM
OPEN, WRITE,READ (p/sistema operacional)

Instrução/Comando **CLRDOT**

SINTAXE GERAL

Número da linha CLRDOT L,C

CLRDOT é uma instrução utilizada no modo programa podendo também ser usada como comando no modo direto.

A instrução CLRDOT é utilizada por alguns computadores europeus, assim como o sueco ABC 80. Sua função é semelhante à da instrução RESET. CLRDOT L,C "desliga" um bloco de gráficos na tela. O bloco a ser "desligado" é especificado pelas coordenadas que seguem a instrução CLRDOT, sendo que L especifica a linha (de 0 até 71 no modo gráfico) e C especifica a coluna (de 2 a 79 no modo de gráficos).

CLRDOT 8,14 por exemplo, faz com que o computador desligue o bloco localizado na nona fileira e na décima quinta coluna a partir do canto esquerdo superior da tela do vídeo.

```
EXEMPLO 1 (p/computador ABC-80)
100 REM USANDO A INSTRUCAO CLRDOT
110 PRINT CHR o (12) "LIMPA TELA"
120  A$ = "CLRDOT FOI APROVADO SE APARECER UMA LINHA NO CANTO
ESQUERDO DA TELA E DESAPARECER EM SEGUIDA"
130 PRINT A$
140 FOR R = 1 TO 2000 : NEXT R
150 PRINT CLR ¤ (12)
160 FOR L = 0 TO 23
170 PRINT CUR(L,0) : CHR o (151) :
180 NEXT L
190 P = 5
200 FOR C = 2 TO 35
210 SETDOT L,C
220 NEXT C
230 FOR R = 1 TO 1000 : NEXT R
240 FOR C = 2 TO 35
250 CLRDOT L,C
260 NEXT C
270 FOR R = 1 TO 1000 : NEXT R
280 END
```

EXECUÇÃO DO PROGRAMA

```
RUN
CLRDOT  FOI  APROVADO SE APARECER UMA LINHA NO CANTO ESQUERDO
DA TELA
E DESAPARECER EM SEGUIDA
```

(Uma linha horizontal deve aparecer brevemente perto da borda superior da tela)

SE O SEU COMPUTADOR NÃO TIVER A INSTRUÇÃO CLRDOT.

Se o seu computador não aceita a instrução CLRDOT, porém aceita RESET, SET e PRINT @, execute o exemplo abaixo para verificar o resultado do Exemplo 1.

EXEMPLO 2

```
200 REM USANDO A INSTRUCAO RESET P/VERIFICAR EXEMPLO 1
210 PRINT CHR$(28) : PRINT CHR$(31)
220  A$  = "RESET FOI APROVADO SE APARECER UMA LINHA NO CANTO
ESQUERDO DA TELA DESAPARECENDO EM SEGUIDA "
230 PRINT A$
240 FOR R = 1 TO 2000 : NEXT R
250 PRINT CHR$(28) : PRINT CHR$(31)
260 PRINT @ P, A$
270 P = 8
280 FOR C = 6 TO 35
290 SET(P,C)
300 NEXT C
310 FOR R = 1 TO 2000 : NEXT R
320 FOR C = 6 TO 35
330 RESET(P,C)
340 NEXT C
350 FOR R = 1 TO 2000 : NEXT R
360 END
```

EXECUÇÃO DO PROGRAMA

```
RUN
RESET  FOI  APROVADO  SE APARECER UMA LINHA NO CANTO ESQUERDO
DA TELA DESAPARECENDO EM SEGUIDA
```

VARIAÇÕES DE USO

Não se tem conhecimento de nenhuma.

VEJA TAMBÉM

RESET, SET, $

Comando/Instrução **CLS**

SINTAXE GERAL

CLS
ou
número de linha CLS

CLS pode ser utilizado como comando no modo imediato ou como instrução no modo programa.

CLS limpa toda impressão na tela e move o cursor para a posição mais superior à esquerda da tela; é como um RESET no vídeo inteiro, usado quando se quer limpar o vídeo, antes de apresentar um resultado.

CLS limpa a tela sem apagar o programa da memória.

CLS é usado como comando ou instrução em computadores que seguem a linha TRS-80 (americano).

OBS.: CLS executa a mesma função da tecla (CLEAR) em muitos teclados, ou seja, apagando instantaneamente a tela inteira sem perturbar o programa.

EXEMPLO 1 (Usando CLS como comando direto)

Tecle, CLS em seguida aperte a tecla <ENTER>

OBS.: Toda impressão que estiver na tela será apagada e o cursor se posicionará no canto superior esquerdo da tela.

EXEMPLO 2

```
200 REM USANDO A INSTRUCAO CLS
210 PRINT "SE A TELA FOR LIMPA"
220 PRINT "E NAO APARECER MENSAGEM ERRO"
230 PRINT "O SEU COMPUTADOR ACEITA"
240 PRINT "A INSTRUCAO <CLS>"
250 FOR Y = 1 TO 1000 STEP 0.2
260 NEXT Y
270 CLS
```

EXECUÇÃO DO PROGRAMA

```
RUN
```

OBS.: Se você executar o programa e a tela não for limpa, seu computador não aceita a instrução CLS.

VARIAÇÕES DE USO

Alguns computadores, entre eles o COLOR COMPUTER TRS-80, utilizam CLS(n) para limpar a tela e de acordo com o valor de (n) determinam a cor de fundo na tela. Sendo utilizado CLS sem um número, a cor utlizada será a cor atual de fundo.

Os números de 0 a 8 abaixo indicam qual o número que liga determinada cor.

0 = PRETO 1 = VERDE 2 = AMARELO
3 = AZUL 4 = VERMELHO 5 = MARRON
6 = CYAN 7 = MAGENTA 8 = ALARANJADO

OBS.: Funcionando como instruções CLS e CLEAR são completamente diferentes.

Muitas telas de vídeo podem ser limpas mediante a utilização de um carácter do código ASCII; experimente PRINT CHR$(24) ou PRINT CHR$(12) ou verifique no seu manual e procure na tabela ASCII qual o código que é usado para APAGAR TODA TELA.

Em computadores que seguem a linha APPLE CLS equivale a HOME que tem por finalidade apagar a tela e posicionar o cursor no canto superior esquerdo da mesma. Nos computadores que seguem a linha SINCLAIR, CLS tem a mesma finalidade de limpar a tela, porém o cursor vai para o canto inferior esquerdo da tela.

VEJA TAMBÉM
HOME CHR$(X) tabela ASCII

Função **CODE**

SINTAXE GERAL

Número de linha PRINT CODE(string)

Ao todo existem 256 caracteres, e cada um tem um código entre 0 e 255. Para fazer a conversão entre códigos e caracteres, existem duas funções: CODE e CHR$.

CODE é utilizada pelos computadores que seguem a linha SINCLAIR (inglês) como TK-82, TK-83 TK-85, NE-8000, CP-200, para converter um carácter a seu "código numérico".

CODE aplica-se a uma STRING (alfanumérico), e dá o código do primeiro carácter do "STRING" (ou 0 se o "STRING" for vazio).

OBS.: O SINCLAIR (inglês) não utiliza o código ASCII empregado por virtualmente todos os demais computadores americanos.

EXEMPLO: PRINT CODE("A")

imprimirá 38, que é o número de código para letra A no computador SINCLAIR.

```
EXEMPLO 1
100 REM USANDO A FUNCAO CODE
110 PRINT "O CODIGO NUMERICO PARA A LETRA B E  ";
120 PRINT CODE("B")
130 PRINT
140 PRINT "ENTRE COM QUALQUER NUMERO, LETRA OU CARACTER";
150 INPUT A$
160 PRINT "O CODIGO NUMÉRICO DE " A$ "E  " CODE(A$)
170 END

EXECUÇÃO DO PROGRAMA
RUN
O CÓDIGO NUMÉRICO PARA LETRA B É  39

ENTRE COM QUALQUER NÚMERO, LETRA OU CARÁCTER ? $
O CÓDIGO NUMÉRICO DE $ E  13

VEJA TAMBÉM
CHR$, ASC
```

Comando/Instrução **COLOR**

SINTAXE GERAL

Número de linha COLOR = número da cor
COLOR = exprnm

COLOR pode ser utilizada como comando no modo direto ou como instrução no modo programa.

COLOR é geralmente usado em computadores que seguem a linha do APPLE II (americano) como APP II (Unitron), MICROENGENHO etc., para colocar a cor a ser plotada no modo gráfico de baixa resolução.

Em todos os computadores que seguem a linha APPLE II e aceitam o comando ou a instrução COLOR, uma cor deve ser especificada para gráficos antes de se tentar plotar uma linha ou ponto.

Já os computadores que seguem a linha TRS-80 mod I e III não dispõem de funções gráficas sem modificações; daí pode-se ignorar o comando ou instrução COLOR.

COLOR = N

Se N é um número real, será convertido para um inteiro. O limite dos valores de N é de 0 até 255; e são tratados em módulo 16.

Valores maiores que 15 repetem as cores mostradas abaixo, os nomes das cores estão associados a números:

COD.COR		COD.COR	
0	PRETO	8	MARRON
1	MAGENTA	9	LARANJA
2	AZUL ESCURO	10	CINZA
3	PURPURA	11	COR DE ROSA
4	VERDE ESCURO	12	VERDE
5	CINZA	13	AMARELO
6	AZUL MEDIO	14	AGUA MARINHA
7	AZUL CLARO	15	BRANCO

OBS:por ausência, fica adotado COLOR = 0.

O sinal de igual (=) deve ser colocado entre COLOR e o valor da COR.

COLOR = 12, por exemplo, seleciona a cor VERDE para a próxima instrução de gráfico. O comando GR coloca COLOR em zero (0). SCRN localiza a cor de um ponto sobre a tela. Quando usado no modo TEXT, COLOR determina qual caracter é colocado sobre a tela pela instrução PLOT. Se usado no modo HGR, COLOR é ignorado.

COLOR pode ser utilizado quer como comando, quer como instrução de programa.

```
EXEMPLO 1
100 REM USANDO A INSTRUCAO COLOR
110 GR
120 FOR N = 0 TO 15
130 COLOR = N
```

```
140 X = N*2
150 HLIN 0,39 AT X
160 NEXT N
170 END
```

EXECUÇÃO DO PROGRAMA

Se o seu computador aceita a instrução COLOR, cada uma das 16 cores deve ser exibida como linha horizontal de lado a lado da tela.

COLOR gera a cor para gráfico de baixa resolução não tendo efeito quando se está usando gráfico em alta resolução.

A instrução COLOR no modo programa é um fator na determinação de que carácter é colocado na tela pela instrução PLOT.

VEJA TAMBÉM

GR, HLIN-AT, VLIN-AT, PLOT

Instrução **COMMON • COM**

SINTAXE GERAL

Número da linha COMMON lista de variáveis.

COMMON, utilizada no BASIC 8220 (computador LABO) tem como finalidade passar os valores armazenados em variáveis para um programa encadeado (CHAIN).

Usa-se a instrução COMMON em conjunto com a instrução CHAIN. A instrução COMMON pode aparecer em qualquer parte de um programa, embora se recomende que apareça no início do programa.

A mesma variável não pode aparecer em mais de uma instrução COMMON. As variáveis matriciais são especificadas afixando-se "()" ao nome da variável.

EXEMPLO:1

```
100 REM USANDO INSTRUCAO COMMON
110 COMMON A,B,C,D (),G$
120 CHAIN "PROG3"
```

NOTA COMPLEMENTAR:

A instrução COMMON é utilizada em determinados computadores para transferência de valores armazenados em variáveis, de um programa para outro. Se cada um de dois (ou mais) programas contiver instruções semelhantes em COMMON (comum), o segundo programa será então ligado por CHAIN ao primeiro, sendo que os valores atuais armazenados nas variáveis denominadas em COMMON (comum) ficarão disponíveis ao segundo programa .

EXEMPLO: Se o primeiro programa contiver a instrução:

```
310 COMMON A,B,C,D,E
```

e o segundo programa contiver a instrução:

```
330 COMMON V,W,X,Y,Z
```

o valor final de A se vai tornar o valor inicial de V, o valor final de B se torna valor inicial de W e assim sucessivamente.

EXEMPLO 2

Armazene este programa em disquete ou fita cassete sob o nome "EXEMPLO"

```
200 REM USANDO A FUNCAO COMMON
210 COMMON X
220 PRINT "O PROGRAMA EXEMPLO RECEBEU EM X O VALOR "; X
230 X = X*2
240 CHAIN PRINCIPAL, 240
250 END
```

Dê agora entrada no computador com o seguinte programa e

conserve-o na mesma fita após o exemplo, ou num disquete sob o nome "PRINCIPAL".

```
200 REM PROGRAMA *PRINCIPAL*
210 COMMON X
220 X = 5
230 CHAIN EXEMPLO
240 PRINT "E DEVOLVEU EM X O VALOR "; X
250 END
```

Prepare seu disquete ou fita cassete para leitura sob comando de programa e em seguida RUN (rode).

```
EXECUÇÃO DO PROGRAMA
RUN
O PROGRAMA DE TESTE RECEBEU EM X O VALOR 5
E DEVOLVEU EM X O VALOR 10
```

ORTOGRAFIA ALTERNATIVA

Em alguns computadores, COM é utilizado como forma abreviada de COMMON.

VARIAÇÕES DE USO

A instrução COM em alguns computadores proporciona também a capacidade de se especificar a extensão das variáveis strings até no máximo 65 caracteres.

COM X$(80)1, Y$5, por exemplo, estabelece um arranjo de string de um só carácter X$(n) com 80 elementos, bem como a variável string Y$, com extensão de 5 caracteres.

VEJA TAMBÉM

CHAIN, DIM

Comando **CON • CONT**

SINTAXE GERAL

CON

CON é um comando que só deve ser usado em modo direto.

CON comando da versão Integer BASIC para computadores que seguem a linha APPLE para devolver a execução de um programa na instrução seguinte à parada. Se um programa escrito na versão Integer BASIC estiver sendo executado e se desejar pará-lo, basta teclar simultaneamente CTRL-C e então irá aparecer a mensagem STOPPEND AT e o número da linha onde parou a execução. Para continuar devemos usar o comando CON.

CON opera após a execução de uma parada com CTRL-C, e por vezes após o RESET. Se não houver interrupção do programa, o CON simplesmente trava o sistema. Um programa não pode continuar se foi interrompido por CTRL-C numa declaração de entrada tipo INPUT.

Se uma linha de programa foi inserida ou retirada, ou foi gerada uma mensagem de erro desde que a interrupção ocorreu, o CON poderá rodar normalmente, ou por vezes travar o sistema ou mesmo gerar mensagem de erro.

VARIAÇÕES DE USO

Na versáo BASIC APPLESOFT para os computadores que seguem a linha APPLE usa-se CONT.

VEJA TAMBÉM

CONT, END, STOP, RUN

Comando **CONT • CON • CO • C.**

SINTAXE GERAL

CONT

CONT é um comando do Applesoft; devolve a execução na próxima instrução após a parada.

CONT é utilizado como comando no modo direto e não tem aplicação como instrução no modo programa, uma vez que é usado somente depois que o programa tiver sido interrompido por uma intrução STOP, pela tecla BREAK ou CTRL-C (Control C).

Durante esta interrupção você pode verificar e mudar valores de variáveis e depois recomeçar a execução do programa (a partir do ponto em que foi interrompido) com o comando CONT.

Nos computadores que seguem a linha APPLE, o comando CONT opera após a parada do programa com STOP, END ou CTRL-C. Se uma declaração INPUT for interrompida com CTRL-C, o programa não poderá ser continuado.

Para o Integer BASIC usa-se o comando CON.

CONT (CONTinua) recomeça a execução do programa na linha em que se segue à interrupção e não como RUN, que faz com que a execução se inicie no começo do programa zerando antes as variáveis. OBS.: é normalmente usado após a parada do programa provocada por uma instrução STOP para depuração.

O comando CONT não deverá ser usado após a edição de uma linha, nem quando for gerada uma mensagem de erro depois do programa ter sido interrompido.

```
EXEMPLO 1
100 REM USANDO O COMANDO CONT
110 PRINT "USE O COMANDO CONT"
120 STOP
130 PRINT "*** FUNCIONOU ***"
140 END
```

EXECUÇÃO DO PROGRAMA

```
RUN
BREAK EM 120
```

Tecle CONT, em seguida aperte a tecla (ENTER - RETURN - CR - NEWLINE - FL ou EOL)

```
*** FUNCIONOU ***
```

ORTOGRAFIAS ALTERNATIVAS

Utilizam-se diversas outras abreviaturas de CONTinue, entre as quais estão CON, CO e C. Experimente cada uma delas com o programa EXEMPLO para verificar qual é a que seu computador aceita.

VARIAÇÕES DE USO

Não se tem conhecimento de nenhuma.

VEJA TAMBÉM

STOP, RUN, END, BREAK

Comando/DOS **COPY**
COPYA • COPI

SINTAXE GERAL

COPY (e pressionar a tecla RETURN, CR, ENTER, NEWLINE ou equivalente)

COPY aciona a impressora de forma a tirar uma cópia impressa de uma imagem estacionária na tela do vídeo.

COPY é um comando que permite transcrever para uma impressora exatamente o que está na tela do vídeo, naquele momento. É natural que se houver movimento, isto é , símbolos e caracteres mudando de lugar, seja necessário primeiro parar o processamento do programa, por exemplo, através do comando CTRL C ou BREAK.

Após a tela estar "congelada", o conteúdo da tela pode ser copiado em uma impressora, fazendo o que se costuma chamar "hard copy" do vídeo. É claro, também, que as características da impressora devem ser compatíveis com as da tela, caso contrário a impressão não reproduzirá fielmente o que está mostrado no vídeo.

Como exemplo de diferenças podemos citar o número de caracteres por linha (que deve ser compatível, caso contrário as linhas não ficarão com a mesma disposição que aparecem na tela), o tipo de caracteres (há caracteres, especialmente gráficos, no computador que não existem nas impressoras, o que significa que será impresso um símbolo diferente do que se encontra na tela).

COPYA em alguns microcomputadores é um programa utilitário para realizar cópias de disquete gravados.

Em computadores que seguem a linha SINCLAIR tais como TK-82, TK-83, TK-85, NE-8000, CP200

COPY imprime uma cópia do conteúdo do vídeo de TV.

```
EXEMPLO 1
100 LPRINT "ESTE PROGRAMA",,,
110 LLIST
120 LPRINT,, "ESCREVE O CONJUNTO DE CARACTERES",,,
130 FOR N = 0 TO 255
140 LPRINT CHR$ N;
150 NEXT N
```

Obtenha no vídeo uma listagem do programa acima e escreva COPY.

O acesso computador impressora pode ser interrompido pressionando a tecla BREAK(espaço), exceto quando a impressora estiver imprimindo. Se executar estas instruções sem ligar a impressora, perderá toda a instrução passando à instrução seguinte. Contudo, pode acontecer que ele encrave, pelo que a tecla BREAK o irá desencravar.

NO SISTEMA OPERACIONAL COM DISQUETE

COPY é utilizado como comando do sistema operacional (DOS).
Em computadores que seguem a linha TRS-80 (Versão Disquete) como CP-500, DGT-100 e outros:
COPY faz a cópia de um ou mais arquivos de três formas:

A) COPY arquivo-fonte arquivo-destino

Arquivo-fonte é uma especificação do arquivo a ser copiado.
Arquivo-destino é uma especificação de arquivo do nome e drive do arquivo copiado.

B) COPY arquivo-fonte:d

Arquivo-fonte (definido acima)
:d diz ao computador para copiar o arquivo no drive d, usando o mesmo nome de arquivo.

C) COPY/EXT:d

/EXT é uma especificação de arquivo arbitrária à qual o nome de arquivo é omitido e a extensão é dada. O sistema operacional (DOS) copiará todos os arquivos que tenham uma extensão compatibilizante não importando o nome do arquivo.
:d (definido acima).
Este comando copia o arquivo-fonte, no novo arquivo, definido pelo arquivo-destino. Ele possibilita a cópia de um disquete para outro utilizando um único drive se necessário. (No último caso, devem-se incluir especificações de drive em ambas as especificações de arquivo). Para sistemas de drive simples (DRIVE 0), ambos os disquetes devem ser do DOS. (Isto é, disquetes de dados não são permitidos no DRIVE 0).

EXEMPLOS

COPY ARQANTIG/BAS ARQNOVO/BAS

Esse comando copia o ARQANTG/BAS em outro chamado ARQNOVO/BAS. O DOS procurará em todos os drives pelo ARQANTG/BAS e o copiará no primeiro disco sem proteção de gravação.

COPY ARQNOME/TXT:1

Este comando especifica o arquivo chamado ARQUIVO DE NOME/TXT para um outro disco.

COPY ARQ/EXT:0:1

Este comando copia o ARQ/EXT do drive 0 para o drive 1.

COPY /BAS:0:1

Diz ao computador para copiar todos os arquivos do drive 0 que tenham a extensão/BAS.
Os arquivos serão copiados no drive 1, usando as extensões e nomes de arquivos presentes.

EXEMPLO PRÁTICO

Use COPY para conseguir a cópia do seu arquivo em outro disco, sempre que seu arquivo for atualizado.

Você também poderá utilizar este comando para reestruturar um arquivo para acesso mais rápido. Certifique-se de que o disco-destino esteja menos segmentado que o disco-fonte; do contrário o arquivo novo poderá ser mais segmentado que o antigo.

Para dar um outro nome ao mesmo arquivo, use RENAME, não COPY.

ORTOGRAFIA ALTERNATIVA

COPI é utilizado no computador LABO.

COPI copia um ou mais arquivos em um outro arquivo, na mesma unidade ou em unidades diferentes.

SINTAXE:

COPI # = #1 arq.fam. (CR)

ou

COPI # arq.fam.1 = #1 arq.fam.2, arq.fam.3

código da unidade do arquivo destino

#1 código da unidade do(s) arquivo(s) origem(ns) não havendo referência à unidade, assume-se a unidade corrente.

arq.fam. = nome do arquivo a ser copiado.

arq.fam.1 = nome do arquivo a ser criado com a cópia dos arquivos arq.fam.2 e arq.fam.3.

arq.fam.2, arq.fam.3 = nome dos arquivos que serão unidos para criar o novo arquivo (arq.fam.1)

As referências ambíguas somente devem ser usadas nos nomes de arquivo origem.

Para obter uma cópia de segurança de todos os arquivos do disco, é suficiente emitir o seguinte comando:

COPI # = #1 *.*

Nos computadores que seguem a linha APPLE como MICROENGENHO, AP-II e outros utiliza-se o programa COPYA, contido no disquete mestre, para copiar todo o conteúdo de um disquete em um outro novo.

Neste programa de cópia, o disquete a ser copiado será denominado disquete "original". O conteúdo total do disquete original será copiado em outro disquete denominado "duplicata". Este disquete duplicata não precisa ser inicializado antes de fazer a cópia.

Antes de copiarmos um disquete, seria uma boa idéia proteger o disquete original. Isto evitaria que seu conteúdo fosse, acidentalmente, apagado.

O programa, inicialmente, assume que o disquete original será colocado no acionador atualmente selecionado (isto é, de onde o programa COPYA foi carregado), conectado no cartão controlador no SLOT atualmente selecionado. O disquete duplicata será assumido como estando no outro acionador conectado no mesmo SLOT controlador (mesmo que ali não haja um outro acionador). Para usar qualquer um destes valores assumidos, para acionador ou para conector, basta pressionar a tecla RETURN (CR ou equivalente) à medida que os valores assumidos forem aparecendo. Se qualquer um destes valores assumidos estiver errado, ou para o disquete original ou para o duplicata, devemos digitar o número correto quando aparecer o valor assumido.

Eis um exemplo de uso do programa COPYA para um computador

equipado com apenas um acionador de disquete. Ele assume que o acionador de disquete esteja ligado no cartão controlador, no mesmo conector de onde foi carregado o programa COPYA.

PROCEDIMENTOS:

1) Coloque o disquete mestre no acionador atualmente selecionado e digite

RUN COPYA

Após alguns segundos veremos na tela

PROGRAMA DUPLICADOR DE DISQUETE
CONECTOR ORIGINAL : ASSUMIDO = 6

2) Se for para usar o valor assumido pelo programa, isto é, o conector número 6 (SLOT 6) neste exemplo, basta pressionar a tecla CR (RETURN ou equivalente).

3) Quando a mensagem

ACIONADOR ASSUMIDO = 1

aparecer, pressione CR (RETURN ou equivalente).

4) Pressione CR (RETURN ou equivalente), novamente, quando aparecer

CONECTOR DUPLICATA : ASSUMIDO = 6

5) Quando surgir o número assumido para o acionador do disquete duplicata

ACIONADOR ASSUMIDO = 2

digite 1, pois estamos usando apenas um acionador.

6) A mensagem

PRESSIONE CR (RETURN ou equivalente) PARA COMECAR A COPIA

aparecerá na tela do vídeo.

Este é o sinal para retirar o disquete mestre do acionador e inserir o disquete duplicata.

7) O programa prosseguirá, então, com a cópia, enviando, inicialmente, a mensagem de inserir o disquete original e, depois, informando que se encontra em fase de leitura das informações do disquete original. O programa informa quando o disquete duplicata deve ser inserido no acionador e quando está formatando (INITiando) o disquete. Depois que o disquete duplicata estiver inicializado, o programa dará a instrução de reinserir o disquete original de tal modo que o programa possa ler mais algumas informações e, então, reinserir o disquete duplicata para que estas informações possam ser copiadas. Estas duas etapas serão repetidas até que o disquete original esteja

totalmente copiado.

8) Quando a cópia estiver completa, veremos a mensagem

VOCE QUER FAZER MAIS UMA COPIA?

Se digitar S(sim), o procedimento de cópia será repetido, assumindo os mesmos valores anteriores. Se digitar um N(não), o controle do computador voltará ao DOS (sistema operacional com disquete).

Se utilizarmos mais de um acionador no programa de COPYA, não aparecerão as mensagens de retirar e inserir disquetes.

Caso tentarmos copiar um disquete protegido contra escrita, receberemos a seguinte mensagem de erro:

E/S : ERRO
PAROU EM (numero da linha)

Até que o rótulo de proteção contra escrita não seja retirado do disquete duplicata, nenhuma informação poderá ser nele gravada.

A mensagem

E/S ERROR

ou IMPOSSIVEL LER
ou IMPOSSIVEL ESCREVER

aparecerá na tela de vídeo se a porta do acionador estiver aberta ou se não houver disquete no acionador. Estas mensagens podem, também, indicar problemas com o disquete no acionador. Esteja certo de que os valores a serem assumidos não tenham sido alterados após estas mensagens.

Função COS • COSD • COSG

SINTAXE GERAL

Número da linha INSTRUCAO = variável COS(argumento)

COS(X) é uma função usada geralmente no modo programa devendo ser precedida de uma instrução LET ou PRINT.
COS(X) retorna o co-seno de um ângulo.
Esta função nos retorna o co-seno de um número ou expressão X, expresso em radianos.
Nao é disponível em Integer BASIC.
O cálculo do COS(X) é feito com precisão simples.

EXEMPLO 1

```
100 REM USANDO FUNCAO COS
110 PI = 3.1415926
120 PRINT "CO-SENO DE PI = ";
130 PRINT INT(COS(PI))
140 PRINT "CO-SENO DE PI/2 = ";
150 PRINT INT(COS(PI/2))
160 END
```

EXECUÇÃO DO PROGRAMA

```
RUN
CO-SENO DE PI = -1
CO-SENO DE PI/2 = 0
```

Para a obtenção do co-seno de X quando este estiver em graus, utilize:
COS(X * 0,0174533)

EXEMPLO 2

```
200 REM USANDO A FUNCAO COS EM GRAUS
205 P = 3.14159:R = 2 * P / 360
210 DEF FN R(X) = X * 0.0174533
220 PRINT
230 PRINT "ENTRE COM UM VALOR EM GRAUS"
240 INPUT D
250 PRINT "CO-SENO = "; COS (FNR(D))
260 GOTO 200
```

EXECUÇÃO DO PROGRAMA

```
RUN
ENTRE COM UM VALOR EM GRAUS
? 90
CO-SENO = -7.49014E-07

ENTRE COM UM VALOR EM GRAUS
? 0
CO-SENO = 1
```

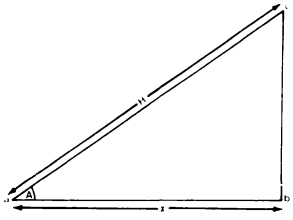
```
ENTRE COM UM VALOR EM GRAUS
? 45
CO-SENO = .707106543

ENTRE COM UM VALOR EM GRAUS
? 60
CO-SENO = .5
```

Para converter valores de radianos a graus, multiplique os radianos por 57,29578.

Alguns microcomputadores aceitam a medida do ângulo em graus ou grados (100 grados = 90 graus). Esses computadores utilizam a função COSD para graus e COSG para grados.

Co-seno (COS) é definido como sendo a relação entre o comprimento do lado adjacente ao ângulo em causa e o comprimento da hipotenusa, num triângulo de ângulo reto.



VARIAÇÕES DE USO

Alguns interpretadores (raros) convertem tudo automaticamente para graus.

SE O SEU COMPUTADOR NÃO TIVER A FUNÇÃO COS(X)

Se o seu computador aceita SIN(X) <seno> porém não tem a função COS(X), pode-se utilizar um GOSUB 3010 no programa principal para a seguinte sub-rotina:

```
3000 END
3010 REM *CO-SENO* ENTRADA X EM GRAUS, SAIDA Y
3020 Y = SIN((90-X)/57.29577951)
3030 RETURN
```

VEJA TAMBÉM

SIN, ASN, ATN, TAN, ACS, COSH, SINH, TANH

Função **COSH • CSH**

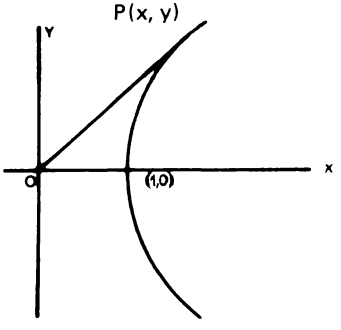
SINTAXE GERAL

Número de linha LET C = COSH(N)

COSH é uma função usada normalmente no modo programa e geralmente precedida de uma instrução LET ou PRINT.
COSH = (EXP(X) + EXP(-X))/2
Retorna o co-seno hiperbólico de X.

COSH(N) é uma função que calcula o co-seno hiperbólico de um número. As funções hiperbólicas exprimem relações baseadas numa hipérbole semelhante à maneira em que as funções trigonométricas se identificam num círculo. Se na hipérbole unitária (isto é, o gráfico de X*X - Y*Y =1) uma linha for estendida do ponto de origem até um ponto "P" na curva (veja-se diagrama), forma-se uma região com área N/2. COSH(N) dará então o valor da coordenada X do ponto de interseção. (SINH(N) dará o valor de Y.

Ao contrário do que ocorre com as funções trigonométricas, N não denomina a medida de um ângulo e, portanto, não se apresenta em graus ou radianos. N pode ser qualquer número real, positivo ou negativo, mas COSH(N) é sempre superior ou igual a 1.




```
EXEMPLO 1
100 REM USANDO A FUNCAO COSH
110 PRINT "DE ENTRADA NUM VALOR";
120 INPUT N
130 C = COSH(N)
140 PRINT "O CO-SENO HIPERBOLICO DE ";N;" E'";C
3999 END
```

```
EXECUÇÃO DO PROGRAMA (utilizando-se o valor 5)
DE ENTRADA NUM VALOR ? 5
O CO-SENO HIPERBOLICO DE 5 E' 74.2099485
```

Se O SEU COMPUTADOR NÃO TIVER ESTA FUNÇÃO

Se o seu computador não aceitar a função COSH, poderá calcular o valor substituindo a função COSH por EXP, conforme página seguinte

```
130 C = 0.5 * (EXP(N) + EXP(-N))
```

Se o seu computador tampouco tiver a função EXP, use a seguinte sub-rotina. O programa de sub-rotina achada sob EXP deve também ser incluído.

```
3000 GOTO 3999
3430 REM *SUBROTINA COSH* INPUT N, SAIDA C
3432 REM UTILIZA TAMBEM A, B, E, L, E X INTERNAMENTE
3434 X = N
3436 GOSUB 3200
3438 C = E
3440 X = -N
3442 GOSUB 3200
3444 C = 0.5*(C+E)
3446 RETURN
3999 END
```

Para utilizar esta sub-rotina, efetue a seguinte alteração no exemplo 1:

```
130 GOSUB 3430
```

Se o seu computador não aceitou a função EXP, utilize a seguinte sub-rotina.

```
3000 GOTO 3999
3200 REM SUBROTINA EXPONENCIAL - ENTRADA X, SAIDA E
3202 REM UTILIZA TAMBEM INTERNAMENTE A, B E L
3204 L = INT(1.4427*X)+1
3206 IF ABS(L)<127 THEN 3218
3208 IF X<=0 THEN 3214
3210 PRINT X; "ESTA' FORA DA FAIXA"
3212 STOP
3214 E = 0
3216 RETURN
3218 E = .693147*L-X
3220 B = X
3222 A = 1.32988E-3-1.41316E-4*E
3224 A = ((A*E-8.30136E-3)*E+4.16574E-2)*E
3226 E =(((A-0.166665)*E+0.5)*E-1)*E+1
3228 A = 2
3230 IF L>0 THEN 3238
3232 A = .5
3234 L = -L
3236 IF L = 0 THEN 3244
3238 FOR X = 1 TO L
3240 E = A*E
3242 NEXT X
3244 X = B
3246 RETURN
```

ORTOGRAFIA ALTERNATIVA

O BASIC-V da Harris utiliza CSH ao invés de COSH.

VARIAÇÕES DE USO

Não se tem conhecimento de nenhuma.

VEJA TAMBÉM

SINH, TANH, EXP

Função **COUNT**

SINTAXE GERAL

Número de linha LET variável = COUNT

COUNT é utilizado apenas como função no modo programa.
COUNT é uma função que "conta" o número de caracteres impressos desde o último retrocesso do carro.
COUNT é semelhante a POS.
COUNT é reconhecido por poucos interpretadores BASIC sendo uma função do computador ARCON ATOM.

EXEMPLO 1

```
100 REM USANDO A FUNCAO COUNT
110  PRINT  "OS  CARACTERES DESTA LINHA SÃO NA SUA TOTALIDADE
DE: ";
120 X = COUNT
130 PRINT X + 3
140 END
```

EXECUÇÃO DO PROGRAMA

```
RUN
OS CARACTERES DESTA LINHA SAO NA SUA TOTALIDADE DE: 37
```

VEJA TAMBÉM
LEN, POS

Comando **CREATE**

SINTAXE GERAL

CREATE nome de arquivo [LRL = aaa, REC = bbb]

Cria um arquivo pré-alocado.

CREATE é um comando utilizado nos microcomputadores que seguem a linha TRS-80 modelo 1 e III como CP-500 DGT 1000 e outros.

nome de arquivo é uma especificação de arquivo.

LRL = aaa é o comprimento lógico de uma gravação. aaa é um número decimal compreendido entre 0 e 255. No caso de omissão do mesmo, será considerado o número 256.

REC = bbb é o número de gravações a armazenar. bbb é o número de gravações desejadas.

No caso de omissão do mesmo, nenhuma gravação será alocada.

Este comando possibilita a criação de arquivos e a pré-alocação (separação) de espaço para futuras informações. Ele é diferente do procedimento de omissão (normal) do DOS 500, no qual o espaço é alocado a um arquivo dinamicamente, isto é, de acordo com a necessidade de se escrever dados do arquivo.

Se abrir o arquivo para gravações seqüenciais, o DOS 500 recuperará (deslocará) qualquer bloco em desuso, caso o arquivo esteja fechado. Se for aberto para acesso aleatório, o DOS 500 não recuperará espaço, enquanto o arquivo estiver fechado. Pode-se usar CREATE para preparar um arquivo que contenha uma quantidade conhecida de dados. Geralmente, isto acelera as operações de gravação de arquivos. A leitura de arquivos será também mais rápida, pois arquivos pré-alocados são menos segmentados ou dispersos no disco, necessitando de menos movimento do mecanismo de leitura/gravação para localizar as gravações.

EXEMPLOS

CREATE ARQDADOS/BAS (REC = 300,LRL = 0)

Cria um arquivo chamado ARQDADOS/BAS e aloca espaço para 300 gravações de 256 bytes.

CREATE NOMES/TXT.IRIS (LRL = 64,REC =50)

Cria um arquivo chamado NOMES/TXT protegido pela senha IRIS.

O arquivo será amplo o bastante para conter 50 gravações, cada uma com 64 bytes de comprimento.

CREATE FOLHAPAG/BAS não aloca espaço algum nele.

EXEMPLO PRÁTICO

Suponha que deseje armazenar informações sobre até 250 empregados, cada gravação dos dados será desta forma:

NOME (até 25 letras)

CARTEIRA DE IDENTIDADE (até 11 caracteres)

DESCRICAO DO SERVICO (até 92 caracteres)

Sendo assim, suas gravações necessitariam ter 128 bytes (25+11+92) de comprimento.

Poderíamos criar um arquivo apropriado com este comando:

```
CREATE PESSOAL/TXT (REC=250,LRL=128)
```

Uma vez criado, este arquivo pré-alocado permitiria gravações mais rápidas do que um arquivo alocado dinamicamente, pois o DOS 500 não teria que parar a gravação periodicamente para alocar mais espaço (exceto se você exceder a quantidade pré-alocada).

Comando **CSAVE**

SINTAXE GERAL

CSAVE("nome do programa")

CSAVE é utilizado para gravar o programa contido na memória de computador para fita cassete.

CSAVE "X" sintaxe para os computadores das linhas TRS-80 e SINCLAIR, salva o conteúdo da área de programa numa fita cassete; "X" é opcional para os computadores que seguem a linha TRS-80, obrigatório para os computadores da linha SINCLAIR e inexistente nos micros que seguem a linha APPLE cuja sintaxe é somente SAVE.

O gravador deve estar devidamente conectado e no modo de gravação, a fita cassete posicionada, antes que você dê entrada ao comando CSAVE. Deve especificar o nome do arquivo com este comando.

O nome do arquivo programa pode ser qualquer carácter alfanumérico, a não ser ASPAS (").

O programa armazenado na fita cassete irá então sustentar o nome do arquivo especificado e poderá ser localizado pelo comando CLOAD, o qual pergunta por este nome de arquivo particular.

OBS.: Deve-se sempre escrever os nomes dos arquivos apropriados na caixa do cassete para posteriores referências.

EXEMPLO: CSAVE "PROG1" descarrega o presente e anexa o rótulo "PROG1"

CSAVE "A" descarrega o presente programa e anexa o rótulo "A"

O comando CSAVE é utilizado por alguns computadores tais como os com interpretador microsoft da linha TRS-80; CP-500, FENIX, DGT-100.

```
EXEMPLO 1
100 REM USANDO O COMANDO 'CSAVE'
110 PRINT "ESTE PROGRAMA TESTA A CARACTERISTICA CSAVE"
120 END
```

Ligue o gravador de cassete no modo de gravar e tecle o comando CSAVE. O computador deve controlar a operação do gravador de cassete ligando e desligando o motor no começo e no fim do ciclo de gravação.

Depois que o programa tiver sido gravado em fita cassete, tecle NEW (ou qualquer que seja a intrução exigida) para limpar a memória, removendo o programa. Carregue o programa da fita de volta para o computador (veja CLOAD). Liste o programa para se certificar de que o programa conservado na memória do computador é idêntico ao que se deu inicialmente entrada.

```
EXECUÇÃO DO PROGRAMA
ESTE PROGRAMA TESTA A CARACTERISTICA CSAVE
```

NOTA COMPLEMENTAR: CSAVE (nome do programa) é utilizado em alguns computadores que empregam CSAVE para designar um nome

específico ao programa que está sendo gravado em fita cassete. O nome do arquivo pode conter um ou mais números, letras ou outros símbolos selecionados de ASCII, mas é possível que seja reconhecido pelo computador apenas o primeiro caráçter. O nome do programa identifica o programa para recuperação posterior através do comando CLOAD (nome do programa).

Grave o programa do exemplo 1 em fita cassete, mediante o comando CSAVE "PROG1", apague a memória, e em seguida carregue o programa de volta ao computador, mediante a aplicação do comando CLOAD "PROG1".

Liste o programa para verificar possíveis erros.

VARIAÇÕES DE USO

CSAVE* pode ser também utilizado com BASIC 5,0 da Microsoft para salvar em fita magnética os valores de um arranjo numérico.

VEJA TAMBÉM

CLOAD, LIST, STORE

Função **CSNG**

SINTAXE GERAL

Número de linha PRINT CSNG(argumento)

CSNG é uma função que quando utilizada vem precedida por uma instrução, geralmente PRINT.
CSNG conversão numérica para precisão simples.
CSNG retorna para a precisão simples a representação do argumento.
Quando o argumento é um valor de dupla precisão, ele é retornado com seis dígitos significativos, sendo que o último dígito significativo é arredondado pelo 5/4.
EXEMPLOS: PRINT CSNG(.7777777777777778) e arredondado para .777778
PRINT CSNG(.3333333333333333) é retornado como .333333
PRINT CSNG(X) converte o argumento representado no valor da variável X para um número de precisão simples.

OBS.: A função CSNG é utilizada para alterar números ou variáveis numéricas que foram definidos previamente como sendo de "dupla precisão" de volta para a "precisão simples" normal. As variáveis listas na função CSNG retornam aos estados originais de dupla precisão, se forem utilizadas novamente sem a função CSNG.

As variáveis de "precisão simples" são capazes de armazenarem números com não mais de sete dígitos (imprimindo-se apenas seis dígitos). A dupla precisão significa uma exatidão que se estende a 17 dígitos. Sendo utilizado CSNG com um número de dupla precisão tendo mais de seis dígitos, o respectivo número será arredondado para seis casas significativas.

```
EXEMPLO 1
100 REM USANDO A FUNCAO 'CSNG'
110 DEFDBL Y
120 Y = 2345678901234567
130  PRINT  "CSNG ALTERA O VALOR DO ARGUMENTO Y DE ";Y;" PARA
";CSNG(Y)
140 PRINT "E DE VOLTA PARA O VALOR DE ";Y;" AO SER REMOVIDA"
150 END
```

EXECUÇÃO DO PROGRAMA

```
CSNG  ALTERA  O VALOR DO ARGUMENTO Y DE 2345678901234567 PARA
2.34568e+15
E DE VOLTA PARA O VALOR DE 2345678901234567 AO SER REMOVIDA
```

VARIAÇÕES DE USO

Nao se tem conhecimento de nenhuma

VEJA TAMBÉM

DEFSNG, DEFDBL, DEFINIT, CDBL, !, #, %, CINT

Função **CVI • CVS • CVD**

SINTAXE GERAL

CVI CVS CVD

No BASIC 8220 dos computadores da linha LABO, as funções CVI, CVS e CVD convertem valores de cadeia em valores numéricos. Os valores numéricos que são lidos a partir de um arquivo de disco randômico devem ser convertidos de cadeia para números.

CVI converte uma cadeia de 2 bytes em inteiro.

CVS converte uma cadeia de 4 bytes em um número de precisão simples.

CVD converte uma cadeia de 8 bytes em um número de dupla precisão.

ARQUIVOS RANDÔMICOS

A criação de arquivos randômicos e o acesso a eles requer mais passos de programa do que os arquivos seqüencias, mas há vantagens em usar arquivos randômicos. Uma das vantagens é que estes arquivos exigem menos espaço no disco, porque o BASIC os armazena em formato binário compactado.

(Um arquivo seqüencial é armazenado como uma série de caracteres ASCII).

VARIAÇÕES DE USO

Nao se tem conhecimento de nenhuma.

Função **CUR**

SINTAXE GERAL

Número da linha CUR(Índice linha, Índice coluna)

CUR é utilizado como função no modo programa, juntamente com a instrução PRINT.

CUR é utilizada em alguns computadores europeus. Posiciona o próximo carácter de impressão num ponto desejado L,C.

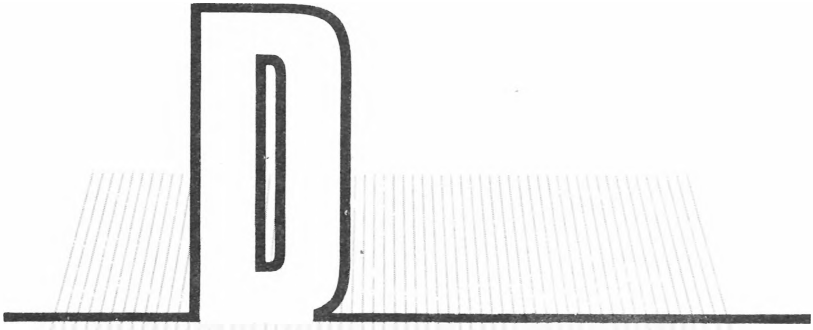
PRINT CUR(L,C) produz resultados semelhantes a PRINT AT (64*L+C) ou PRINT @ (64*L,C)

```
EXEMPLO 1
100 REM USANDO A FUNCAO CUR
110 PRINT CHR$(12):REM LIMPA TELA
120 PRINT CUR(12,16); "LISTA"
130 PRINT "A FUNCAO CUR PASSOU SE LISTA ESTA NO CENTRO"
140 END
```

```
EXECUÇÃO DO PROGRAMA
RUN
                        LISTA
A FUNCAO CUR PASSOU SE LISTA ESTA NO CENTRO
```

VEJA TAMBÉM

PRINT AT, @, LOCATE



DATA • DAT • D. • D
DEF • DEFDBL
DEFINT • DEFSNG • DEFSTR
DEG • DEGREE • DELETE
DEL • DIGITS • DIM • DIR
DRAW • DSP

Operador D

D operador que indica a modalidade de precisão dupla.

D utilizado como operador nos computadores que seguem a linha TRS-80 e reconhecido por algumas versões BASIC para definir uma variável como número de dupla precisão.

O meio pelo qual o BASIC armazena as informações determina o espaço de memória que essas ocuparão e, ao mesmo tempo, afeta a velocidade de processamento, sendo que quanto maior a quantidade de dados, maior o tempo para processá-los.

Os dados numéricos podem ser classificados como números inteiros, de precisão simples ou dupla.

Os inteiros são os mais eficientes e menos precisos, já os de precisão dupla são mais precisos e menos eficientes.

O operador D é utilizado para assinalar "dupla precisão, nos números expressos em exponencial ou notação científica padrão".

EX: 1.23456689D+20

Os números expressos em precisão simples são escritos em notacão exponencial, utilizando-se a letra "E".

EX: 1.2345E+20.

PRECISÃO DUPLA - Números de precisão dupla envolvem até 17 dígitos significativos e podem representar valores na mesma faixa usada para números de precisão simples. Um valor de precisão dupla requer oito bytes de memória para armazenagem por isso as operações aritméticas envolvendo no mínimo um número de precisão dupla são mais vagarosas do que se forem efetuadas utilizando-se operandos inteiros ou de precisão simples.

Por exemplo: 8.00100708D12

Quando o símbolo D for usado em um número decimal. representará "precisão dupla vezes 10 elevado a..." Portanto, 8.00100708D12 representa o valor 8.00100708 X 10 elevado a 12.

```
EXEMPLO 1
100 REM 'D' USANDO O OPERADOR D
110 X# = 23456789O123456789
120 PRINT "NOTACAO EXPONENCIAL D FOI APROVADA SE"
130 PRINT X#; "CONTEM A LETRA D"
140 END

EXECUÇÃO DO PROGRAMA
RUN
NOTACAO EXPONENCIAL D FOI APROVADA SE
2.345678901234568D+18 CONTEM A LETRA D
```

VARIAÇÕES DE USO

A letra "D", à semelhança de todas as demais letras do alfabeto, é utilizada por todos os computadores para indicar uma variável numérica.

VEJA TAMBÉM

E, #, !, DEFDBL, DEFSNG

Instrução **DATA • DAT • D.**

SINTAXE GERAL

Número da linha DATA lista de constantes
DATA const [,const...]

DATA é uma instrução utilizada no modo programa. Não será dada mensagem de erro ao se tentar entrar DATA em modo imediato, mas os dados não serão lidos dentro do programa com READ.

A instrução DATA tem por finalidade armazenar as constantes numéricas e alfanuméricas que são acessadas pela(s) instrução(ões) READ do programa.

A instrução DATA contém dados a serem lidos por uma instrução READ. Os itens constantes da instrução DATA devem estar separados entre si por vírgulas; entretanto não pode haver vírgula depois da intrução DATA e nem depois do último valor que for entrado. Os valores da intrução DATA podem ser constituídos de números tanto positivos como negativos ou de STRINGS (valores alfanuméricos).

DATA X,Y,Z

Quando a instrução READ é encontrada em um programa DATA X,Y,Z será lido seqüencialmente.

A instrução DATA pode ser colocada em qualquer linha do programa, ela não precisa ser executada para ser acessada por uma declaração READ, pois somente pode ser utilizada quando na execução do programa for executada a instrução READ.

OBS: As instruções READ e DATA equivalem à instrução de entrada INPUT sendo que os valores de entrada contidos numa instrução DATA serão armazenados automaticamente nos endereços da memória representados pelas variáveis contidas na instrução READ.

A instrução DATA especifica tanto valores numéricos quanto alfanuméricos. Os valores (constantes) alfanuméricos não precisam ser colocados entre aspas; as aspas não são necessárias a menos que o valor alfanumérico tenha espaços, vírgulas, traço ou dois pontos.

Um sinal aspas (") não pode ser usado como constante; ele deve ser especificado usando a função CHR$(34).

Um ou mais parâmetros constantes podem ser nulos, ou seja, apenas espaços em branco.

Para uma constante numérica nula é colocado 0 (zero).

Para uma constante alfanumérica nula é colocado " " espaço em branco entre aspas na variável alfanumérica.

A instrução DATA não é disponível em Integer BASIC.

EXERCÍCIO 1

```
100 REM USANDO A INSTRUCAO DATA
110 PRINT "NOME","IDADE"
120 READ N$
130 IF N$ = "FIM" THEN GOTO 160
135 READ I
140 IF I > 18 THEN PRINT N$,I
```

```
150 DATA
    ROSSANA,40,CLOVIS,39,SUSY,16,
WELLIGTON,18,ANDRE,19,PATRICIA,20,PAULO,51,JANETE,38,FIM
155 GOTO 120
160 PRINT "FIM DA LISTA"
170 END

EXECUÇÃO DO PROGRAMA
RUN
NOME              IDADE
ROSSANA           40
CLOVIS            39
ANDRE'            19
PATRICIA          20
PAULO             51
JANETE            38
FIM DA LISTA
```

OBSERVAÇÕES: A maioria dos computadores permite valores alfanuméricos (strings) nùma instrução DATA. Alguns exigem que os strings estejam sempre entre aspas, ao passo que outros requerem aspas tão somente se o string precedido por vírgula contiver dentro de si uma vírgula ou estiver seguido de um espaço em branco, uma vírgula ou "dois pontos".

```
EXERCÍCIO 2
200 REM USANDO A INSTRUCAO DATA UTILIZANDO-SE STRINGS
210 DATA "LINHA ",210, "FOI"
220 READ X$,X,Y$
230 PRINT "A INSTRUCAO DATA NA ";X$;X;Y$; " APROVADA"
240 END

EXECUÇÃO DO PROGRAMA
RUN
A INSTRUCAO DATA NA LINHA 210 FOI APROVADA
```

Remova as aspas das variáveis de string na linha 210 e faça rodar o programa de novo para ver se são necessários no seu interpretador.

As instruções DATA podem ser colocadas em qualquer linha do programa, porém a primeira instrução READ ao ser executada irá procurar a instrução DATA de menor número de linha.

ORTOGRAFIAS ALTERNATIVAS

Alguns computadores (raros) utilizam DAT como abreviatura de DATA.

VARIAÇÕES DE USO

Não se tem conhecimento de nenhuma.

VEJA TAMBÉM

READ, RESTORE

Instrução **DEF**

SINTAXE GERAL

```
DEF FNnvar(arg) = exprnm
```

Número da linha instrução DEF FN nome (lista de parâmetros) = Definição da Função.

A declaração DEF FNX permite funções de aplicação específica a ser definida e usada dentro de programas em Applesoft.

Nos computadores que seguem a linha APPLE esta instrução permite que o programador defina uma função outra vez através da função FNX. Como inexiste equivalente na linha TRS-80, deve-se especificar sempre que necessário.

DEF FN é uma instrução encontrada no BASIC MICROSOFT e utilizada para definir uma função que será encontrada repetidamente em um programa.

EXEMPLO:

```
DEFFN MOD(X,Y) = (X ↑ 2 + Y ↑2) ↑ (1/2)
A = MOD(6,8)
```

Se for definida a função MOD tal como acima, ao ser aplicada aos valores 6 e 8 teremos em A o resultador, 10. Isto seria equivalente a ter colocado:

```
A = (6 ↑ 2 + 8 ↑ 2) ↑ (1/2)
```

Como se pode ver, economiza-se memória do computador e esforço do programador, pois agora toda vez que se necessitar calcular a fórmula, com valores diferentes, basta utilizar MOD.

A instrução DEF permite ao usuário DEFinir (criar) novas funções (sendo que a maioria dos computadores tem algumas funções já integradas na estrutura) que podem então ser utilizadas da mesma forma de quaisquer funções intrínsecas (partes integrantes).

Quando FNnvar é chamada, o valor da variável fictícia é especificado por uma expressão numérica, variável ou constante.

DEF FN deve ser executado antes que se possa chamar a função definida por ele.

A declaração DEF FN inteira deve aparecer numa única linha de programa. Entretanto uma função definida anteriormente pode ser incluída em exprnm, de forma que podem ser desenvolvidas funções da complexidade que se desejar.

EXEMPLO

```
100 X = 2
110 DEF FNA(N) = 3*N-1
120 PRINT FNA(X)
```

A função FN neste exemplo é denominada "A" (FNA), sendo-lhe designada a equação 3*N-1 na linha 120. A variável numérica (X) que segue FNA e substituída para a variável simulada (N) na instrução DEF cada vez que se executa FNA.

EXEMPLO 1

```
100 USANDO A INSTRUCAO DEF
```

```
110 PRINT "DE ENTRADA NO RAIO DE UM CIRCULO (EM POLEGADAS)";
120 INPUT R
130 DEF FNC(X)=2*3.14159*X
140 PRINT "A CIRCUNFERENCIA DE UM CIRCULO"
150 PRINT "COM RAIO DE "; R; "POLEGADAS E' "; FNC(R);"
POLEGADAS"
160 END
```

```
EXECUÇÃO DO PROGRAMA
RUN
(UTILIZANDO-SE 5)
DE ENTRADA NO RAIO DE UM CIRCULO (EM POLEGADAS)? 5
A CIRCUNFERENCIA DE UM CIRCULO
COM RAIO DE 5 POLEGADAS E' DE 31.4159 POLEGADAS
```

Alguns computadores permitem mais de uma variável na expressão DEFinida, devendo cada uma delas ser listada após a função FN (variável).

```
EXEMPLO 2
200 REM USANDO A VARIAVEL MULTIPLA DEF
210 DEF FNA(X,Y) = (X+Y)/2
220 PRINT "DE ENTRADA EM QUAISQUER DOIS NUMEROS";
230 INPUT X,Y
240 A = FNA(X,Y)
250 PRINT "A MEDIA DE ";X; " E ";Y; " E' ";A
500 END
```

```
EXECUÇÃO DO PROGRAMA
RUN
(UTILIZANDO-SE 10 E 30)
DE ENTRADA EM QUAISQUER DOIS NUMEROS? 10, 30
A MEDIA DE 10 E 30 E' 20
```

Alguns microcomputadores permitem que a mesma função seja DEFinida em mais de uma linha. No exemplo seguinte a função FNA é definida como sendo X*2 se o valor da variável X for menor do que 10, ou como X/2 se o valor de X for maior do que 10 ou igual a 10.

```
EXEMPLO 3
300 REM USANDO A INSTRUCAO DEF QUE REQUER MAIS DE UMA LINHA
310 PRINT "DE ENTRADA NUM VALOR X QUE SEJA SUPERIOR A 10 OU
INFERIOR A 10 ";
320 INPUT X
330 DEF FNA(X)
340 FNA = X*2
350 IF X 10 THEN 370
360 FNA = X/2
370 FNEND
380 PRINT "O VALOR DA FUNCAO E "; FNA(X)
390 END
```

```
EXECUÇÃO DO PROGRAMA
RUN
(UTILIZANDO-SE 12)
DE ENTRADA NUM VALOR X QUE SEJA SUPERIOR A 10 OU INFERIOR A
10?12
```

O VALOR DA FUNCAO E 6

A instrução FNEND no exemplo 3 instrui ao computador que desista de definir a função FNA. As instruções DEF de linhas múltiplas devem sempre terminar com a instrução FNEND, sendo que o computador não permite desvios para dentro ou para fora de instruções DEF de linhas múltiplas. Para mais informações veja a seção referente a FENEND.

OBS: DEF FN não é disponível em Integer BASIC.
A declaração de definição DEF FN é ilegal em modo imediato. Entretanto uma função que seja definida após o último NEW, CLEAR ou LOAD pode ser referenciada em modo imediato.

SE O SEU COMPUTADOR NÃO TIVER ESSA FUNÇÃO
Se o seu computador não tiver a capacidade DEF, substitua FN por uma sub-rotina que contenha a mesma equação.
A instrução DEF no exemplo 2 pode, por exemplo, ser substituída pela seguinte sub-rotina.

```
1000 A = (X+Y)/2
1010 RETURN
```

juntamente com as seguintes alterações do exemplo 2

Delete a linha 210, altere a linha 240 e adicione a linha 260

```
240 GOSUB 1000
260 GOTO 500
```

Alguns BASICs permitem definir pela instrução DEF funções de string.

10 DEF FLN$(A$) = LEFT$(A$,1), por exemplo, devolve o primeiro carácter de um string (isto é conveniente para fins de verificação de uma entrada no teclado).

VARIAÇÕES DE USO
Não se tem conhecimento de nenhuma.

VEJA TAMBÉM
FN, FNEND, GOSUB, RETURN

Instrução **DEFDBL**

SINTAXE GERAL

Número da linha instrução DEFDBL grupo de letras.

DEFDBL declara que uma variável ou conjunto de variáveis são números de precisão dupla.

Suponha que num determinado programa você precisa que as variáveis L até P sejam de precisão dupla.

Portanto, o programa deve iniciar da seguinte forma:

```
100 DEFDBL L-P
```

Todas as variáveis que começarem com as letras L,M,N,O e P serão de precisão dupla.

DEFDBL faz com que variáveis, começando com qualquer letra do grupo especificado, sejam armazenadas e tratadas como de dupla precisão, a menos que um carácter de declaração de tipo seja adicionado. A dupla precisão permite 17 dígitos de precisão; 16 dígitos são mostrados quando uma variável de dupla precisão é impressa.

EXEMPLO

```
100 DEFDBL A-E,S-Z
```

faz com que variáveis, começando com uma das letras de A a E ou de S a Z, sejam de dupla precisão.

DEFDBL é uma função normalmente usada no começo de programas, porque ela pode redefinir as variáveis sem caracteres de declaração de tipo.

OBSERVAÇÃO:

A instrução DEFDBL deve ser utilizada tão-somente onde não seja suficiente a precisão simples, uma vez que as variáveis de dupla precisão exigem maior espaço na memória e a sua manipulação requer mais tempo. Na maioria dos computadores a linha DEFDBL deve ser executada antes que a variável listada na instrução DEFDBL se designe um valor numérico.

```
EXEMPLO 1
100 REM USANDO A INSTRUCAO DEFDBL
110 X = 2.345678901234567
120 PRINT "DEFDBL NA LINHA 140 MUDOU O VALOR DA VARIAVEL X"
130 PRINT "DE ";X; "PARA ";
140 DEFDBL X
150 X = 2.345678901234567
160 PRINT X
170 END
```

EXECUÇÃO DO PROGRAMA

```
RUN
DEFDBL  NA  LINHA  140 MUDOU O VALOR DA VARIAVEL X DE 2.23468
PARA 2.345678901234567
```

A maioria dos computadores com capacidade DEFDBL permitem também designar mais de uma variável como sendo de "dupla

precisão" por meio de uma única instrução DEFDBL. DEFDBL B,E,F, por exemplo, define as variáveis B,E e F como sendo de dupla precisão, ao passo que DEFDBL B-F define como sendo de dupla precisão todas as variáveis que começam com as letras B até F.

```
EXEMPLO 2
200 REM USANDO A INSTRUCAO DEFDBL COM VARIAVEIS MULTIPLAS
210 DEFDBL B,F,J-L
220 B = 1/3
230 F = 2/3
240 J = 1/9
250 K = 7.89012345678905678D + 17
260 L = 7.89012345678905678D + 17
270 PRINT "DEFDBL APROVOU CONTANTO QUE OS SEGUINTES"
280 PRINT "NUMEROS CONTENHAM MAIS DE 7 DIGITOS;"
290 PRINT B;F;J;K;L
300 END
```

EXECUÇÃO DO PROGRAMA

```
RUN
DEFDBL APROVOU CONTANTO QUE OS SEGUINTES NUMEROS CONTENHAM
MAIS DE 7 DIGITOS;
```

O "D" antes do "+17" é a mesma coisa que o "E" na notação exponencial, significando, contudo, que o número tem "precisão dupla".

Alguns computadores talvez não imprimam os primeiros três valores conforme assinalados na execução do programa, pelo fato de ser efetuado o cálculo em precisão simples. Este problema pode ser eliminado nos computadores que tiverem sinal indicador de dupla precisão (como, por exemplo, o sinal #). Para produzir os resultados corretos coloque o sinal após cada fração nas linhas 220,230 e 240 conforme segue, para produzir os resultados certos.

```
220 B = 1/3 #
230 F = 2/3 #
240 J = 1/9 #
```

VARIAÇÕES DE USO

Não se tem conhecimento de nenhuma.

VEJA TAMBÉM

DEFSNG, DEFINT, #, %, !, CDBL, CSNG, CINT, D e E.

Instrução **DEFINT**

SINTAXE GERAL

Número da linha, instrução DEFINT, número de variáveis.

A instrução DEFINT é utilizada para DEF(inir) declarar que as variáveis listadas pela instrução DEFINT são números INT(eiros).

DEFINT diz que uma variável ou conjunto de variáveis são números inteiros. Há duas vantagens em usar números inteiros: gasta-se menos memória e as operações aritméticas são executadas mais rapidamente com números inteiros.

As variáveis definidas como sendo números inteiros armazenam o valor de número inteiro dos números designados. Isto tem especial utilidade nos programas extensos, uma vez que se requer menor capacidade de memória para armazenar números inteiros do que números com decimais.

Suponha que num determinado programa você precisa que as variáveis de A até F sejam inteiras. Portanto, o programa deve iniciar da seguinte forma:

```
100 DEFINT A-F
```

Uma desvantagem em potencial da utilização da instrução DEFINT é a incapacidade por parte de muitos interpretadores de processar valores numéricos superiores ao que permite a função INT do interpretador (tipicamente de -32767 a +32767).

A linha DEFINT deve ser executada pelo computador antes que se designe um valor numérico a uma variável listada na instrução DEFINT.

EXEMPLO 1

```
100 REM USANDO A INSTRUCAO DEFINT
110 DEFINT X
120 X = 15.86
130 Y = 15.86
140 IF X = 15 THEN 160
150 GOTO 170
160 IF Y = 15.86 THEN 190
170 PRINT "DEFINT NAO APROVOU NA LINHA 110"
180 GOTO 210
190 PRINT "A INSTRUCAO DEFINT FOI APROVADA NA LINHA 110"
200 PRINT "MUDANDO O VALOR DA VARIAVEL X DE ";Y; "PARA ";X
210 END
```

EXECUÇÃO DO PROGRAMA

```
RUN
A  INSTRUCAO DEFINT FOI APROVADA NA LINHA 110 MUDANDO O VALOR
DA VARIAVEL X DE 15.86 PARA 15
```

A maioria dos computadores com capacidade DEFINT permite também a designação de variáveis múltiplas (com separação por vírgulas) numa única instrução DEFINT. DEFINT B,G,N, por exemplo, define como números inteiros as variáveis B,G e N. DEFINT B-N define como números inteiros todas as variáveis que começam com as letras de B até N.

```
EXEMPLO 2
200 REM USANDO A INSTRUCAO DEFINT COM VARIAVEIS MULTIPLAS
210 DEFINT X,J,K-P
220 X = 8.52
230 Y = 12.25
240 J = -9.91
250 K = 7.562
260 Z = 2352.777
270 P = 15.5
280 PRINT "SE FOREM INTEIROS OS NUMEROS ";X;J;K;Z;P;"
290 PRINT "E O NUMERO ";Y;"FOR DECIMAL,  DEFINT"
300  PRINT  "FOI  APROVADO NO TESTE DE VARIAVEIS MULTIPLAS DA
LINHA 210."
310 END

EXECUÇÃO DO PROGRAMA
RUN
SE FOREM INTEIROS OS NUMEROS 8 -10 7 2352 15
E O NUMERO 12.25 FOR DECIMAL, DEFINT
FOI APROVADO NO TESTE DE VARIAVEIS MULTIPLAS DA LINHA 210.
```

Se o interpretador tiver carácter instrutor de dupla precisão (tal como o sinal # na Basic Microsoft) e um desses caracteres for atribuído a uma variável listada na instrução DEFINT, a variável será tratada como sendo de dupla precisão (ou de precisão simples) porque os caracteres instrutivos sobrepujam a instrução DEFINT.

```
EXEMPLO 3
300 REM USANDO A INSTRUCAO DEFINT
310  REM  UTILIZA  CARACTER  "#"  COMO  INSTRUCAO  TIPO DUPLA
PRECISAO
320 DEFINT X,Y
330 X = 7.12345678901234
340 Y# = 7.12345678901234
350 IF X = Y# THEN 400
360 PRINT "X = ";X
370 PRINT "Y# = ";Y#
380 PRINT "FOI APROVADO, MOSTRANDO QUE # SOBREPUJA DEFINT"
390 GOTO 410
400  PRINT  "CARACTER  #  SOBREPUJA,  DEFINT  NAO  APROVOU NO
TESTE"
410 END

EXECUÇÃO DO PROGRAMA
RUN
X = 7
Y#= 7.12345678901234
FOI APROVADO. MOSTRANDO QUE # SOBREPUJA DEFINT

VARIAÇÕES DE USO
Não se tem conhecimento de nenhuma.

VEJA TAMBÉM

INT. #, DEFSNG, DEFDBL, CINT, CSNG, CDBL, % e !
```

Instrução **DEFSNG**

SINTAXE GERAL

Número da linha, instrução DEFSNG variável.

DEFSNG declara que uma variável ou conjunto de variáveis são números de precisão simples. Faz com que qualquer variável, começando com uma letra no grupo especificado, seja armazenada e tratada como de simples precisão, a menos que um carácter de declaraçãc de tipo seja adicionado. Variáveis de simples precisão e constantes são armazenadas com sete dígitos de precisão e impressos com seis dígitos de precisão. Desde que todas as variáveis numéricas sejam assumidas como sendo de simples precisão, a menos que DEFinidas de outra forma, a instrução DEFSNG é primariamente usada para redefinir variáveis, as quais tenham sido anteriormente definidas como de dupla precisão ou números inteiros.

EX: 100 DEFSNG I, W-Z

Faz com que as variáveis, começando com a letra I ou qualquer letra de W a Z, sejam tratadas como de simples precisão. Entretanto, I% seria ainda uma variável de número inteiro e I#, uma variável de dupla precisão, devido ao uso de caracteres de declaração de tipo. Uma vez que a maioria dos computadores tratam automaticamente as variáveis como sendo de precisão simples, a instrução DEFSNG é utilizada em programas para redefinir variáveis. Na maioria dos computadores a linha DEFSNG deve ser executada antes que se atribua a instrução DEFSNG um valor numérico. A linha 110 abaixo indica que tanto X como Y devem ser mantidos com dupla precisão.

```
EXEMPLO 1
100 REM USANDO A INSTRUCAO DEFSNG
110 DEFDBL X,Y
120 X = 7.890654321678904
130 Y = X
140 PRINT "VALOR DE DUPLA PRECISAO DE Y = ";Y
150 DEFSNG Y
160 Y = X
170 PRINT "VALOR DE PRECISAO SIMPLES DE Y = ";Y
180 END

EXECUÇÃO DO PROGRAMA
RUN
VALOR DE DUPLA PRECISAO DE Y = 7.890654321678904
VALOR DE PRECISAO SIMPLES DE Y = 7.89066
```

A maioria dos computadores com a capacidade DEFSNG permite também a atribuição de variáveis duplas (com separação por vírgulas) numa única instrução DEFSNG. DEFSNG B,G,N, por exemplo, define as variáveis B,G e N como sendo de precisão simples, ao passo que DEFSNG B-N define todas as variáveis que começam com B até N como sendo de precisão simples.

```
EXEMPLO 2
200 REM USANDO A INSTRUCAO DEFSNG COM VARIAVEIS MULTIPLAS
```

```
210 DEFDBL B,F,J-L
220 GOSUB 300
230 PRINT "OS VALORES DE DUPLA PRECISAO DE B,F,J,K e L SAO"
240 PRINT B;F;J;K;L
250 DEFSNG B,F,J-L
260 GOSUB 300
270  PRINT  "OS  VALORES  DE  PRECISAO SIMPLES DE B,F,J,K e L
SAO"
280 PRINT B;F;J;K;L
290 GOTO 370
300 REM SUBROTINA
310 B = 7890.654321
320 F = B/10
330 J = F/10
340 K = J/10
350 L = K/10
360 RETURN
370 END
```

```
EXECUÇÃO DO PROGRAMA
RUN
OS VALORES DE DUPLA PRECISAO DE B,F,J,K e L SAO
7890.654321000001    789.0654321000001
78.90654321000001    7.890654321000001
OS VALORES DE PRECISAO SIMPLES DE B,F,J,K e L SAO
7890.66    789.066   78.9066   7.89066
```

Se o interpretador tiver características instrutivas de dupla precisão (tais como o sinal # na Basic da Microsoft) e/ou de números inteiros (tais como o sinal % na Basic Microsoft), sendo atribuído um desses caracteres a uma variável listada na instrução DEFSNG, a variável é tratada como sendo de dupla precisão (ou como número inteiro) uma vez que os Caracteres Instrutivos sobrepujam a instrução DEFSNG.

```
EXEMPLO 3
300 REM USANDO A INSTRUCAO DEFSNG COMO DUPLA PRECISAO
310 DEFSNG M,N
320 M = 7.890123456789056
330 N# = 7.890123456789056
340 IF M = N# THEM 390
350 PRINT "M =";M
360 PRINT "N# =";N#
370  PRINT  "  APROVOU  O  TESTE  COM  O SINAL # SOBREPUJANDO
DEFSNG"
380 GOTO 400
390 PRINT "O CARACTER # DE SOBREPUJAR NAO FOI APROVADO"
400 END
```

```
EXECUÇÃO DO PROGRAMA
RUN
M = 7.89012
N# = 7.890123456789056
APROVOU O TESTE COM O SINAL # SOBREPUJANDO DEFSNG
```

```
VARIAÇÕES DE USO
Não se tem conhecimento de nenhuma.

VEJA TAMBÉM
DEFINT, #, DEFDBL, !, CSNG, CDBL, CINT, %
```

Instrução **DEFSTR**

SINTAXE GERAL

Número da linha instrução DEFSTR variáveis.

DEFSTR declara que uma variável ou conjunto de variáveis devem ser tratadas como strings.

A variável listada na instrução DEFSTR é tratada da mesma forma como se fosse definida numa variável string pelo sinal $(string).

100 DEFSTR R

Todas as variáveis que começarem com a letra R serão variáveis de string.

OBSERVAÇÃO:

É importante, nos programas extensos, especificar tão-somente aquelas variáveis que necessitam de armazenamento string, uma vez que as variáveis de string necessitam de mais espaço na memória do que as variáveis numéricas.

A linha DEFSTR deve ser executada antes que a variável definida receba uma notação de string.

Suponha que num determinado programa você precisa que as variáveis P,R e X sejam strings, portanto, o programa deve iniciar da seguinte maneira:

100 DEFSTR P,R,X

EXEMPLO 1

```
100 REM USANDO A INSTRUCAO DEFSTR
110 R = 17
120 PRINT "VARIAVEL NUMERICA R =  ";R
130 DEFSTR R
140 R = "STRING DE PROVA"
150 PRINT "VARIAVEL DE STRING R = ";R
160 END
```

EXECUÇÃO DO PROGRAMA

```
RUN
VARIAVEL NUMERICA R = 17
VARIAVEL DE STRING R = STRING DE PROVA
```

A maioria dos computadores com capacidade DEFSTR permite também a atribuição de variáveis múltiplas (com separação por vírgulas), mediante uma única instrução DEFSTR. DEFSTR B,G,N, por exemplo, define como variáveis de string as variáveis B,G e N. DEFSTR B-N define todas as variáveis que começam com as letras de B até N como sendo variáveis de string.

EXEMPLO 2

```
200 REM USANDO A INSTRUCAO DEFSTR COM VARIAVEIS MULTIPLAS
210 DEFSTR B,F,M-O
220 B = " DEFSTR"
230 F = " PASSOU NA"
240 M = " PROVA DE"
250 N = " VARIAVEL MULTIPLA"
```

```
260 O = " NA LINHA 210,"
270 PRINT B;F;M;N;O
280 END

EXECUÇÃO DO PROGRAMA
RUN
DEFSTR PASSOU NA PROVA DE VARIAVEL MULTIPLA NA LINHA 210
```

Alguns interpretadores exigem que se reserve espaço na memória para os strings atribuídos mediante uma instrução DIM ou CLEAR.

Os interpretadores com caracteres instrutivos (tais como %, # ou !) tomam precedência sobre a função DEFSTR ao serem acrescidos a variáveis relacionadas na instrução DEFSTR. Essa característica pode ser testada ao se efetuarem essas alterações no Exemplo 2.

```
260 O = " NA LINHA"
270 PRINT B;F;M;N;O
275 B! = 210
276 PRINT B!
```

O carácter instrutivo de precisão simples (!) acrescentado às linhas 275 e 276 deve sobrepujar a instrução DEFSTR na linha 210, imprimindo o mesmo resultado.

VARIAÇÕES DE USO

Não se tem conhecimento de nenhuma.

VEJA TAMBÉM

DEFDBL, DEFINT, DEFSNG, DIM, CLEAR, $, D (notação exponencial), E (notação exponencial), % (operador de números inteiros), # (precisão dupla) e ! (precisão simples).

Comando/Função/Instrução
DEG • DEGREE

SINTAXE GERAL

DEG e em seguida teclar RUN (executando o programa)

DEG é utilizada por alguns microcomputadores e reconhecida por apenas algumas versões de BASIC, como comando que faz com que o computador execute funções trigonométricas em graus (ao invés de radianos). Um grau é aproximadamente 0.0174533 radianos.

```
EXEMPLO 1
100 REM USANDO DEG COMO COMANDO
110 X = SIN(2.5)
120 PRINT "O SENO DE 2.5 RADIANOS E";X
130 END
```

EXECUÇÃO DO PROGRAMA
RUN
Conforme se viu acima, o computador executará o programa e calculará o seno de um ângulo que mede 2.5 radianos.

O SENO DE 2.5 RADIANOS E 0.598472144

Teclar o comando DEG. Em seguida RUN (rode). O computador dará como saída o seno do ângulo que mede 2.5 DEGrees (graus)

O SENO DE 2.5 RADIANOS E' 0.0436194061

Para transferir o computador de volta para o modo de radianos, teclar RAD ou SCR. (SCR fará também SCRatch eliminar o programa inteiro).

SE O SEU COMPUTADOR NÃO TIVER ESSE COMANDO (FUNÇÃO, INSTRUÇÃO)
Se o seu computador não tiver o comando DEG, isto pode ser simulado no programa multiplicando-se os valores em graus por 0.0174533.
Para utilizar essa conversão no Exemplo 1, efetue a seguinte alteração no programa:

```
110 X = SIN(2.5 * 0.0174533)
```

VARIAÇÕES DE USO
DEG converte as medidas de ângulos na forma de graus, minutos e segundos para a de graus e frações decimais, no computador de bolso TRS-80.
Pode-se utilizar o computador de bolso para converter de volta graus decimais para a forma de graus-minutos-segundos do arco.

EXEMPLO: Entrando com DMS 33.6736
DMS converte uma magnitude de 33.6736 graus para 33 40'25".
Alguns computadores (tais como os que utilizam MAX BASIC)

têm DEG(n) como função intrínseca para converter a graus um valor (n) expresso em radianos.

```
EXEMPLO 2
200 REM USANDO DEG COMO FUNCAO
210 PRINT "DE ENTRADA NUM ANGULO (EXPRESSO EM RADIANOS)"
220 INPUT X
230 Y = DEG(X)
240  PRINT  "  O ANGULO EM RADIANOS DE ";X;" E  IGUAL A ";Y;"
GRAUS"
250 END

EXECUÇÁO DO PROGRAMA (UTILIZANDO-SE 2.5)
RUN
DE ENTRADA NUM ANGULO (EXPRESSO EM RADIANOS)
? 2.5
O ANGULO EM RADIANOS DE 2.5 E' IGUAL A 143.23945 GRAUS
```

ORTOGRAFIA ALTERNATIVA

Alguns computadores (tais como o Sharp/TRS-80 de bolso) utilizam DEGREE como instruçáo que coloca o computador no modo de grau para efeitos de cálculos trigonométricos.

Se seu computador náo tiver a funçáo DEG, pode ser simulada multiplicando-se por 57.29578 os valores em radianos. Para utilizar essa conversão no Exemplo 2, efetue a seguinte alteração no programa:

```
230 Y = X * 57.29578
```

VEJA TAMBÉM

SIN, COS, TAN, ATN, RAD, ASN, ACS

Comando **DELETE • DEL**

SINTAXE GERAL

DELETE nomearq [,Dn] [,Sn] [,Vn]
ou
DELETE número da linha - número da linha.

Retira um arquivo do disco ou apaga linhas de programa no modo direto.

DELETE deve ser usado somente como comando no modo direto, pois não há sentido em usar DELETE como instrução dentro de um programa mandando apagar uma ou mais linhas do mesmo programa.

DELETE é reconhecido por muitos interpretadores BASIC e tem por finalidade suprir linhas de programa. Este comando permite alterar programas mandando apagar uma linha ou um conjunto de linhas de um programa armazenado na memória do computador.

Se desejar apagar apenas uma parte do programa, use o comando DELETE em vez de NEW que apaga todo o programa.

EXEMPLOS:

DELETE 200
(em modo direto, será cancelada a linha 200 do programa).

DELETE 30-90
(em modo direto, serão apagadas todas as linhas de 30 a 90)

DELETE -100
(modo direto, serão apagadas todas as linhas até a de número 100)

DELETE 50-
(modo direto, apagará as linhas do número 50 até o fim do programa)

EXEMPLO 1

```
100 REM USANDO O COMANDO DELETE
110 PRINT "LINHA 110"
120 PRINT "LINHA 120"
130 PRINT "LINHA 130"
140 PRINT "LINHA 140"
150 PRINT "LINHA 150"
160 PRINT "LINHA 160 E FIM DA PROVA DELETE "
```

RUN (rode) o programa para ter certeza de que foram lançadas corretamente todas as linhas.

EXECUÇÃO DO PROGRAMA

```
RUN
LINHA 110
LINHA 120
LINHA 130
LINHA 140
LINHA 150
LINHA 160 E FIM DA PROVA DELETE
```

Uma só linha do programa pode ser eliminada da memória do computador, utilizando-se o comando DELETE (número da linha).

Para testar essa característica, experimente o comando DELETE 140 e rode o programa. Esse comando deve ter eliminado a impressão da linha 140. Verifique dando um LIST e em seguida rodando o programa (RUN).

Em alguns computadores pode-se eliminar da memória mais de uma linha de programa mediante a utilização do comando DELETE (número da linha - número da linha). Eliminam-se assim todos os números de linha dentro da faixa especificada pelo referido comando. Para testar essa característica, experimente o comando DELETE 120-130, e de RUN (rodar) o programa. Devem ter desaparecido as linhas 120 e 130. Alguns computadores exigem que existam realmente os primeiros e/ou últimos números de linha. Outros apagam todos os números na faixa, ainda que não estejam sendo utilizados os números especificados nos dois extremos.

A sintaxe DELETE - (número da linha) é utilizada por alguns computadores para eliminar todos os números de linha desde o primeiro número da linha do programa até o número da linha especificada no comando DELETE. Para testar essa característica, experimente o comando DELETE 150 e rode o programa. Devem ser eliminadas todas as linhas menos a linha 160.

Alguns computadores com a característica DELETE permitem eliminar grupos de números de linha e mais números de linha individuais, mediante o uso de vírgulas.

DELETE 110,140,150 elimina as linhas 110, 140 e 150 do programa. LIST o programa para certificar-se de que as linhas foram realmente eliminadas.

Alguns computadores utilizam DELETE (número da linha) para eliminar todos os números de linha a começar do número da linha especificado no comando DELETE, até chegar no final. Para testar essa característica, experimente o comando DELETE 120-. LIST o programa para se certificar de que foram eliminadas todas as linhas a partir da LINHA 120.

NOTA: Utilizando o DOS (sistema operacional disquete).
DOS 3.3 nos computadores que seguem a linha APPLE:
DELETE retira um arquivo do disquete.
Tendo a seguinte Sintaxe Geral
DELETE nomearq [,Dn] [,Vn] [,Sn]
O nome do arquivo especificado será retirado do disquete.
Dn, Sn, e Vn: Drive número, Slot número, e volume número podem ser especificados em qualquer ordem ou serem omitidos.
Este é um comando do DOS e requer PRINT e CTRL-D em modo programa.

ORTOGRAFIA ALTERNATIVA

Alguns computadores (tais como os computadores que seguem a linha APPLE) utilizam DEL como comando.

DEL lin1, lin2

A versão APPLE de DEL utiliza vírgula nos lugares em que DELETE utiliza tracinhos. Para DELetar (apagar) as linhas 110 até 140, tecle DEL 110,140.

DELetar uma única linha requer que se tecle DEL 120,120 (ou simplesmente teclar o número da linha e pressionar a tecla RETURN).

DEL deve ser seguido por dois números de linha separados por vírgula. Nenhum deles pode ser negativo, e o segundo deve ser maior ou igual ao primeiro.

OBSERVAÇÃO:

Se os números forem idênticos, uma linha (no mínimo) será eliminada.

SE O SEU COMPUTADOR NÃO TIVER ESTE COMANDO

Se o seu computador não tiver o comando DELETE, o mesmo resultado pode ser conseguido teclando individualmente cada número de linha, e, em seguida, apertando a tecla ENTER ou RETURN. Para eliminar numa única operação todos os números de linha, utilize o comando NEW ou SCRATCH.

VEJA TAMBÉM

NEW, LIST, SCRATCH

Instrução **DIGITS**

SINTAXE GERAL

Número da linha DIGITS número,número

A instrução DIGITS é utilizada em BASIC Estendida e reconhecida por poucos interpretadores BASIC para especificar o número máximo de dígitos a ser impresso por uma instrução PRINT.

20 DIGITS 8,2 poderia, por exemplo, ser utilizado num programa em que todos os valores impressos representam cruzeiros e centavos. O primeiro número especifica o número total de dígitos a ser impresso e o segundo o número de casas para a direita da vírgula decimal. O segundo número não deve ser maior do que o número total de dígitos a serem impressos.

Se o valor real for demasiado grande para ser impresso no número de casas permitido, o valor será impresso sob forma exponencial. A parte fracionária do número é arredondada para o número desejado de dígitos onde necessário, não sendo exibidos os dígitos mais para a direita.

EXEMPLO 1

```
100 REM PROGRAMA USANDO DIGITS
110 DIGITS 5,3
120 A = 0.4567839
130 PRINT A
140 PRINT "DIGITS APROVOVOU SE FOR IMPRESSO 0.456"
150 END
```

EXECUÇÃO DO PROGRAMA

```
RUN
0.456
DIGITS APROVADOU SE FOR IMPRESSO 0.456
```

A Super BASIC PERCOM utiliza a instrução DIGITS para especificar tão-somente o número de dígitos a ser impresso após o ponto decimal.

110 DIGITS = 3, por exemplo, limita a três casas decimais todos os valores impressos.

SE O SEU COMPUTADOR NÃO TIVER ESTA INSTRUÇÃO

Não estando disponível DIGITS no seu computador, experimente na linha 110 a instrução PRECISION.

O número máximo de dígitos após o ponto decimal pode também ser apagado eliminando a linha 110, e substituindo a linha 130 por :

```
130 PRINT USING "**.****";A
```

Se tampouco tiver disponível PRINT USING, não se desespere! Substitua:

```
130 PRINT INT(A*10000 + 0.5)/10000
```

VARIAÇÕES DE USO

Não se tem conhecimento de nenhuma.

VEJA TAMBÉM

PRECISION, PRINT USING, IMAGE, INT

Instrução **DIM**

SINTAXE GERAL

DIM var(sub[,sub...]) [,var(sub[,sub...])...]
ou
Número da linha DIM, lista de variáveis subscritas.

DIM A(20), T$(5,4), Y(4+A,8)

DIM é utilizada como instrução no modo programa e quando executada, o computador reserva espaço e dimensões para armazenar os elementos de uma variável subscrita, que pode ser numérica ou alfanumérica.

Tanto nos computadores que seguem a linha APPLE quanto nos da linha TRS-80, se uma variável não houver sido definida por uma declaração DIM, mas estiver sendo utilizada de forma subscrita, é assumido que cada um dos índices poderá variar de 0 a 10.

Para os computadores que seguem a linha Sinclair é obrigatória a utilização do DIM para variáveis subscritas numéricas e o índice varia de 1 (e não 0) até o valor máximo indicado na declaração DIM.

DIM é uma instrução utilizada para reservar espaço de memória interna do computador quando se trabalha com variáveis subscritas, ou seja, com variáveis simplesmente ou multiplamente indexadas.

DIM pode ser considerada uma forma abreviada da palavra DIMensão e que tem o sentido de número de valores (ou fileiras) a ser armazenado.

Nas variáveis com apenas um índice, ou vetores, declara-se apenas uma dimensão.

A instrução DIM é utilizada para estabelecer o número de elementos permitido num conjunto numérico ou alfanumérico. A DIMensão do conjunto é estabelecida colocando todas as variáveis do conjunto depois da instrução DIM, sendo seguidas as mesmas pelo tamanho do conjunto, entre parênteses. O tamanho do conjunto é representado pelo índice ou subscrito que está entre parênteses após o nome do conjunto.

EXEMPLO:

DIM B(12) permite à variável subscrita B utilizar os 13 elementos do conjunto desde B(0), B(1), B(2), até....B(12).

Alguns computadores começam o conjunto com o elemento B(1) ao passo que alguns outros como aqueles que seguem a ANSI BASIC podem definir o elemento mais baixo do conjunto como sendo o de índice 0 ou 1, mediante a utilização da instrução BASE.

EXEMPLO 1

```
100 REM USANDO A INSTRUCAO DIM
110 DIM B(15)
120 PRINT "OS NUMEROS SAO ARMAZENADOS E IMPRESSOS ";
130 PRINT "A PARTIR DE UM CONJUNTO NUMERICO DE UMA SO' DIMENSAO: "
140 FOR Y = 1 TO 15
150 B(Y) = Y
160 PRINT B(Y)" ";
```

```
170 NEXT Y
180 END

EXECUÇÃO DO PROGRAMA
RUN
OS NUMEROS SAO ARMAZENADOS E IMPRESSOS A PARTIR DE UM CONJUNTO NUMERICO DE UMA SO' DIMENSAO:
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
```

Para verificar a capacidade de seu interpretador para armazenar os elementos do conjunto a começar do índice 0 (ou subscrito 0), faça a seguinte alteração no Exemplo 1:

```
140 FOR Y = 0 TO 15
```

Se o seu interpretador aceitou no conjunto o elemento B(0) quando rodar o programa irá imprimir os números de 0 até 15.

A maioria dos computadores permite a cada conjunto utilizar elementos desde o subscrito 0 (ou 1) até o subscrito 10 sem a necessidade de DIMensionar.

Para variáveis com dois índices (ou dois subscritos), em matrizes, constituiu-se uma tabela de L linhas e C colunas, e a instrução deve conter estas dimensões máximas.

EXEMPLO:

Para reservar espaço na memória, quando se pretende operar com uma variável Subscrita A(6,6) usa-se:

Se a declaração de DIM contiver mais linhas ou colunas do que o necessário, o programa poderá funcionar em alguns casos, mas haverá maior utilização de memória, o que poderá ser prejudicial. Por outro lado, se a declaração DIM considerar menos linhas ou colunas do que as que serão utilizadas, será emitida mensagem de erro.

Para as variáveis alfanuméricas, isto é, fileiras, a instrução fixa o número máximo de fileiras.

Ex: 30 DIM F$(25)

significa que o vetor terá, no máximo, 26 fileiras nos computadores que armazenam a partir do elemento de subscrito 0 e 25 nos computadores que armazenam a partir do elemento de subscrito 1.

OBS: em alguns poucos casos a declaração DIM pode ser omitida, entretanto, é conveniente usá-la sempre, antes de qualquer operação com matrizes.

DIM possibilita o ajuste da profundidade, isto é, do número de elementos permitidos por dimensão da matriz, ou lista de matrizes.

Se nenhuma instrução DIM foi utilizada, a profundidade de 11 (subscritos de 0 a 10) será fornecida a cada dimensão da matriz.

OBS: Se for usado um nome de variável subscrita (com uma ou mais dimensões), sem uma instrução DIM, supõe-se que o valor máximo de seu(s) índice(s) (ou subscritos) é 10.

Se for usado índice que seja maior que o máximo especificado, ocorre um erro.

O valor mínimo de um índice é sempre 0, exceto especificação em contrário.

O comando DIM estabelece todos os elementos (não declarados) das matrizes especificadas com um valor inicial zero.

DIM é também utilizado em alguns microcomputadores para estabelecer o tamanho do elemento de maior extensão para arranjos numéricos e de strings que contiverem duas (ou mais) dimensões.

DIM B(25,20), por exemplo, estabelece que o número máximo a ser atribuído ao primeiro índice (que representa a linha) será 25 e que o segundo índice (que representa a coluna da matriz) terá como número máximo 20.

A maioria dos computadores com capacidade de conjuntos de duas e de três dimensões reserva automaticamente espaço de memória para até o elemento de subscrito 10 (ou seja de 0 a 10 onze elementos em cada dimensão).

Muitos computadores pequenos (tais como as variações dos interpretadores da MICROSOFT) reservam espaço de memória dos elementos para apenas a primeira e a segunda dimensão.

```
EXEMPLO 2
200 REM USANDO O DIM NUM COJUNDO DE DUAS DIMENSOES
210 DIM B(4,4)
220 PRINT "ESTES NUMEROS SAO ARMAZENADOS E IMPRESSOS NO
VIDEO ";
230 PRINT "EM UMA MATRIZ NUMERICA DE DUAS DIMENSOES "
235 PRINT
240 FOR L = 1 TO 4
250 FOR C = 1 TO 4
260 B(L,C) = L
270 NEXT C
280 NEXT L
290 FOR L = 1 TO 4
300 FOR C = 1 TO 4
310 PRINT B(L,C),
320 NEXT C
330 PRINT
340 NEXT L
350 END
```

```
EXECUÇÃO DO PROGRAMA
RUN
ESTES NUMEROS SAO ARMAZENADOS E IMPRESSOS NO VIDEO EM UMA
MATRIZ NUMERICA DE DUAS DIMENSOES

1     1           1           1
2     2           2           2
3     3           3           3
4     4           4           4
```

O exemplo abaixo testa a capacidade do computador de DIMensionar variáveis de um conjunto numérico de três dimensões:

```
EXEMPLO 3
300 REM USANDO A INSTRUCAO DIM EM UMA VARIAVEL SUBSCRITA COM
TRES DIMENSOES
310 DIM B(3,4,2)
320 PRINT "ESTES NUMEROS SE ARMAZENAM E SE IMPRIMEM"
330 PRINT "COM UM CONJUNTO NUMERICO DE TRES DIMENSOES"
335 PRINT
340 FOR X = 1 TO 2
350 FOR L = 1 TO 3
360 FOR C = 1 TO 4
370 B (L,C,X) = L
380 NEXT C
390 NEXT L
```

```
400 NEXT X
410 FOR X = 1 TO 2
420 FOR L = 1 TO 3
430 FOR C = 1 TO 4
440 PRINT B(L,C,X),
450 NEXT C
460 NEXT L
470 PRINT
480 NEXT X
490 END
```

EXECUÇÃO DO PROGRAMA

```
RUN
```

ESTES NUMEROS SE ARMAZENAM E SE IMPRIMEM COM UM CONJUNTO NUMERICO DE TRES DIMENSOES

```
1     1     1     1
2     2     2     2

3     3     3     3

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

Para computadores que seguem a linha APPLE existem grandes diferenças entre as versões do Integer BASIC e o APPLESOFT

A) SINTAXE GERAL EM INTERGER BASIC

DIM VAR(sub) [,var(sub)...]

Em Interger BASIC só podemos definir variáveis subscritas (numéricas ou alfanuméricas) de uma dimensão.

Quando se dimensiona uma variável subscrita numérica, é reservado espaço na memória para o número de elementos mais 1.

Os elementos são numerados de 0 até o subscrito indicado.

O elemento 0 de um vetor (arranjo) é igual a variáveis simples de mesmo nome

EX: A(0) = A

Todo sub (subscrito) deve ser um valor inteiro entre 1 e 255 na instrução DIM e o maior dimensionamento permitido está vinculado ao espaço de memória existente.

B) SINTAXE GERAL EM APPLESOFT

DIM var(sub[,sub...]) [,var(sub[,sub])...]

A instrução DIM dimensiona variáveis subscritas de uma ou mais dimensões da seguinte forma:

var(sub) vetores de dimensão simples

var(sub1,sub2) matrizes de duas dimensões

var(sub1,sub2,sub3...) variáveis de dimensões múltiplas

A versão BASIC APPLESOFT permite variáveis subscrita numérica (inteiro ou real) e alfanumérica. Cada elemento de uma variável subscrita é de um tipo especificado pelo nome da variável usada (numérica ou alfanumérica). O número de dimensão para a variável subscrita é dado pelo número de subscritas usadas na instrução DIM. Quando uma variável subscrita é dimensionada cada subscrito deve estar dentro da faixa de 0 até sub, onde sub é o subscrito correspondente à mesma variável na instrução DIM.

O número de dimensões para uma variável subscrita é limitada pela quantidade de memória disponível sendo que o número máximo

pode ser até 88, e isto só é possível quando muitos subscritos forem 0. Uma variável subscrita só pode ser DIMensionada uma vez.

OBSERVAÇÃO:
Se um elemento de uma variável subscrita for referenciado antes que a instrução DIM tenha sido executada, o APPLESOFT atribui o valor de 10 para cada subscrito e a partir daí a variável subscrita será tratada como se tivesse sido DIMensionada em 10 para cada dimensão.

SE O SEU COMPUTADOR NÃO TIVER ESTA INSTRUÇÃO
Se o seu computador não permitir conjuntos multidimensionais, não é difícil simulá-los mediante uma única dimensão. Para utilizar um conjunto de duas dimensões tal como A(2,5), DIMemsionar o conjunto como A(10) e substituir cada referência a A(1,J) por A((I-1)*4+J)). Se forem utilizados os subscritos de zero, o conjunto será DIMensionado como A(17), isto é (2+1)*(5+1) - 1 = 17.
Utilize em seguida A(1*5+J) no lugar de A(I,J).
De maneira semelhante, para três dimensões, para dar instrução para a variável subscrita A(2,5,3), utilize DIM A(30) (ou DIM A(64), se estiver utilizando zero como subscrito), e substitua A(I+J+K) por A(((I-1)*5+(J-1)*3+K) (ou por A((I*5+J)*3+K, se estiver utilizando subscritos zeros.

GERALMENTE
Para uma MATRIZ MxN:
Sem subscrito zero: DIM A(M*N) e utilize A((I-1)*N+J)

Com subscrito zero: DIM A((M+1)*(N+1)-0) e utilize A(L*N+J)

Para um CONJUNTO LxMxN
sem subscrito zero: DIM A(L*M*N) e utiliza A(((I-1)* M+(J-1)) * N+K)
Com subscrito zero: DIM A((L+1)*(N+1)-1) e utiliza A((I*M*J)*N+K)

VARIAÇÕES DE USO
Não se tem conhecimento de nenhuma.

VEJA TAMBÉM
CLEAR, MAT INPUT, MAT PRINT, MAT READ

Comando/DOS **DIR**

SINTAXE GERAL

DIR:d[INV,SYS,PRT]
Lista o diretório do disquete.

DIR é um comando do sistema operacional com disquete, utilizado nos microcomputadores da linha TRS-80 e também usado no CP/M.
Este comando apresenta no vídeo o diretório de arquivos gravados em um disquete.
d é o diretório do drive desejado.
No caso de omissão, será considerado o drive 0.
INV lista os arquivos invisíveis do usuário. No caso de omissão, serão listados os arquivos não invisíveis do usuário.
SYS lista arquivos de sistema e de usuários. No caso de omissão, apenas os arquivos não invisíveis do usuário serão listados.
PRT lista o diretório na impressora. No caso de omissão, o diretório será listado no vídeo somente.
Se nenhuma opção for dada, o DOS-500 listará os arquivos não invisíveis do usuário na tela.
Este comando fornece informações a respeito do disco e dos arquivos nele contidos.
Para paralisar a listagem, pressione @. Para continuar, pressione ENTER e para finalizar, a tecla BREAK

EXEMPLOS

DIR
Mostra o diretório de arquivos não-invisíveis do usuário do Drive 0.

DIR:1 (PRT)
Lista o diretório dos arquivos do usuário no Drive 1 através da impressora.

VARIAÇÕES DE USO
Não se tem conhecimento de nenhuma.

VEJA TAMBÉM
CATALOG

Instrução **DRAW**

SINTAXE GERAL

Número da linha DRAW exprnm [AT colh,linh]

DRAW é uma instrução geralmente usada no modo programa.

DRAW é uma instrução do BASIC APPLESOFT utilizada nos computadores que seguem a linha APPLE, para desenhar uma forma gráfica de alta resolução na tela.

Não há equivalente para os computadores que seguem a linha TRS-80, contudo a instrução POKE pode ser usada como substituto.

Esta instrução não é disponível na versão Integer BASIC do APPLE.

Em computadores que seguem a linha APPLE II como AP II (UNITRON), MICROENGENHO, MAXXI (POLYMAX) DRAW é utilizada para desenhar uma configuração predeterminada (numerada L), a começar do ponto X,Y.

EXEMPLO: DRAW N AT X,Y

Draw com a primeira opção desenha uma forma no modo gráfico de alta resolução, começando do ponto em que a coordenada X é o valor de Y e cuja coordenada Y é o valor Z. A forma desenhada é o X-ésimo definição de forma na tabela de formas previamente carregadas usando o comando SHLOAD (ou uma tabela de forma pode ser escrita dentro da memória do computador em código hexadecimal, usando o programa monitor). X deve estar nos limites 0 até N, onde N é o número (de 0 a 255) da definição de forma dado no byte 0 da tabela de forma. Y deve estar nos limites de 0 até 278. Z nos limites de 0 até 191.

Se qualquer um desses limites é excedido, a mensagem de erro aparecerá na tela (? VALOR ILEGAL).

A cor, a rotação e a escala da forma do desenho devem ter sido especificadas antes de DRAW ser executado.

A segunda opção é similar à primeira, mas desenha a forma especificada iniciando no último ponto plotado pelo mais recente comando HPLOT, DRAW ou XDRAW executado.

DRAW exprnm [AT colh,linh]

A forma identificada pelo valor inteiro de exprnm será desenhada na cor determinada pela declaração HCOLOR executada por último. A escala e rotação da forma devem ser indicadas por instruções SCALE e ROT antes de se dar DRAW.

O desenho inicia no local dado pelo valor inteiro da expressão numérica colh e linh. Se não tiver sido especificada a posição na declaração DRAW, a forma inicia no último ponto indicado pela última instrução DRAW, XDRAW ou HPLOT realizado.

Não devemos usar a instrução DRAW se não houver tabela de formas na memória, pois o sistema poderá travar ou apresentar uma forma sem sentido na tela.

Se o programa se estender até uma área de memória de alta resolução, ele também poderá ser destruído.

OBSERVAÇÃO

O número da forma especificado na exprnm deve estar entre 0

e o número de formas da tabela não podendo exceder a 255.

VARIAÇÕES DE USO

Outra versão, DRAW X,Y é utilizada nos computadores que seguem a linha SINCLAIR ZX-80 para desenhar linha DRAWTO X,Y que executa uma linha da posição atual até a posição (X,Y).

Alguns computadores utilizam PLOT da mesma forma que estes computadores utilizam DRAW.

VEJA TAMBÉM

PLOT, XDRAW

Instrução **DSP**

SINTAXE GERAL

Número da linha DSP var

DSP é reconhecido por alguns interpretadores em BASIC como ferramenta analítica para exibir determinada variável e seu valor, cada vez que se atribua à variável um valor.

DSP var é reconhecida na versão Integer BASIC dos computadores que seguem a linha APPLE e não é disponível na versão BASIC APPLESOFT.

DSP apresenta a variação de valores de variáveis na execução de um programa em Integer BASIC.

Para desligar o DSP use NO DSP var que cancela o modo display para a variável especificada em Integer BASIC.

O valor da variável var e o número de linha corrente são apresentados sempre que o valor desta variável é alterado.

Nos computadores que seguem a linha TRS-80, a instrução equivalente é o TRON e para o NO DSP será o TROF.

A instrução RUN cancela todas as instruções DSP dadas.

Devemos usar os comandos GOTO ou CON ao depurar programas com DSP em modo direto.

O número de linha associado da linha é também exibido, sendo precedido do sinal *. Permite-se a utilização de mais de uma instrução DSP num programa.

EXEMPLO:

```
200 DSP X
210 DSP Y
```

Quando executado o programa (exemplo acima) instruir o computador a exibir (imprimir) as variáveis X e Y, e seus valores, juntamente com os números de linha, cada vez que sejam designados ou redesignados valores às mesmas.

EXEMPLO 1

```
100 REM USANDO A INSTRUCAO DSP
110 DSP X
120 DSP Y
130 X = 10
140 Y = 15
150 Z = X * Y
160 X = X + Z
170 PRINT "INSTRUCAO DSP FOI APROVADA"
180 END
```

EXECUÇÃO DO PROGRAMA

```
*130 X = 10
*140 Y = 15
*160 X = 25
A INTRUÇÃO DSP FOI APROVADA
```

SE SEU COMPUTADOR NÃO TIVER ESTA INSTRUÇÃO

Esta característica muito cômoda para percepção de defeitos pode ser duplicada acrescentando uma linha de teste temporária em cada ponto em que está sendo alterada a variável cuja pista se vem seguindo.

```
EXEMPLO 2
200 REM SIMULACAO DE DSP
210 X = 7
220 PRINT "*30 X = "; X
230 Y = 15
240 PRINT "*40 Y = "; Y
250 Z = X*Y
260 X = X + Z
270 PRINT "*50 X = "; X
280 PRINT "FIM DA SIMULACAO DSP"
290 END
```

EXECUÇÃO DO PROGRAMA

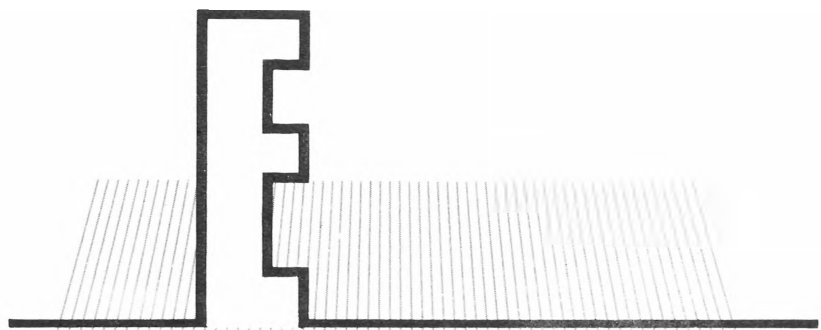
```
*30 X = 7
*40 Y = 15
*50 X = 112
```

VARIAÇÕES DE USO

Não se tem conhecimento de nenhuma.

VEJA TAMBÉM

TRON, TRACE



E • EDIT • ELSE
END • EQ • ERASE • ERL
ERRL • ERR • ERRN • ERROR
EXEC • EXCHANGE • EXIT • EXP

Operador E

O operador E é aceito tanto nos equipamentos que seguem a linha APPLE como TRS-80 ou SINCLAIR, se utiliza para indicar "notação exponencial", ou "notação científica padronizada", de precisão simples.

3.21E + 12, por exemplo, significa 321 seguido de 10 zeros.

Os números expressos em dupla precisão são escritos na notação exponencial mediante a aplicação da letra "D".

Um exemplo disto seria 2.34567890D+20

EXEMPLO 1

```
100 REM USANDO EXPOENTE DE PRECISAO SIMPLES "E"
105 X = 234567890
110 PRINT "O OPERADOR E FOI APROVADO SE"
120 PRINT X; "CONTEM A LETRA E"
130 END
```

EXECUÇÃO DO PROGRAMA

```
RUN
O OPERADOR E FOI APROVADO SE 2.34568E+08 CONTEM A LETRA E
```

VARIAÇÕES DE USO

A letra E, à semelhança de todas as demais letras do alfabeto, é utilizada em todos os computadores para indicar uma variável numérica.

VEJA TAMBÉM

D, !, #, DEFSNG, DEFDBL, CSNG, CDBL

Comando **EDIT**

SINTAXE GERAL

EDIT Número de linha

Este comando passa para o modo de edição, não é disponível nos computadores que seguem a linha APPLE II, sendo reconhecido em computadores que seguem a linha TRS-80.

Deve-se especificar a linha que se deseja corrigir, e isto se faz de duas maneiras, descritas abaixo:

EDIT número de linha - permite a correção de uma linha específica. Se o número de linha não for usado no programa, um erro FC ocorrerá

ou

EDIT permite a correção de linhas do programa (última linha introduzida ou alterada ou na qual o erro ocorreu)

EXEMPLO

20 FOR R = 0 TO 50 : PRINT "EXEMPLO" : NEXT 50

aperte ENTER

aparecerá: 20

Você está então no modo de edição e pode editar a linha 20.

Estando no modo de edição o vídeo apresenta: 20

Apertando a tecla de espaço vez por vez permitirá que você alcance o lugar desejado para fazer a edição.

Se deseja andar oito posições para fazer uma edição, pode apertar a tecla de espaço sete vezes ou então apertar a tecla 7(sete) e a tecla de espaço.

ESTANDO NO MODO DE EDIÇÃO VOCÊ PODERÁ USAR AS TECLAS ABAIXO COMO SE SEGUEM:

TECLA DE ESPAÇO

Se apertar a tecla de espaço, o cursor se moverá para a direita.

EXEMPLO:

Estando no modo de edição o vídeo apresenta:

20-

Aperte a tecla de espaço uma vez e o vídeo apresenta:

20 F -

Apertando mais uma vez:

20 FO -

Portanto este comando permite que alcance o lugar desejado para fazer a edição.

Se desejar andar sete posições para fazer uma edição pode apertar a tecla de espaço sete vezes ou então apertar a tecla 7(sete) e a tecla de espaço.

TECLA <-----

É semelhante à tecla de espaço, só que o movimento é da direita para a esquerda, isto é, para trás.

TECLA L

Quando aperta a tecla L, o resto da linha, que está sendo

editada, é listada no vídeo. O cursor executa um line-feed e o número da linha aparece novamente. Você continua no modo de edição.

EXEMPLO:

```
EDIT 20   RETURN
20 -
```

Aperte L (não é necessário apertar RETURN)

```
20 FOR R=0 TO 50:PRINT "EXEMPLO":NEXT
20 -
```

TECLA I

Este comando permite inserir caracteres. Você sempre coloca caracteres onde o cursor se localiza.

Se apertar a tecla <--- deleta caracteres ao invés de inseri-los.

EXEMPLO:

Suponhamos que deseja inserir o carácter 1 após a palavra "EXEMPLO" da linha 20.

```
EDIT 20   RETURN
20
```

Agora aperte a tecla de espaço quantas vezes forem necessárias para alcançar a letra "O" da palavra EXEMPLO. O vídeo mostrará:

```
20 FOR R=0 TO 50:PRINT "EXEMPLO -
```

Aperte a tecla I para entrar no modo de inserção. Você pode então colocar os caracteres desejados, no caso o 1.

```
20 FOR R=0 TO 50:PRINT "EXEMPLO 1 -
```

Para sair do modo de inserção veja o comando abaixo.

TECLA SHIFT - ↑

Aperte SHIFT ↑ para sair de um dos seguintes modos: I, X ou H.

Após apertar SHIFT ↑ ainda está no modo de edição e pode, por exemplo, inserir caracteres em outras posições da linha.

No exemplo acima, após apertar SHIFT ↑ aperte a tecla L para listar a linha inteira. Se tudo está conforme deseja, aperte a tecla RETURN para sair do modo de edição e voltar ao modo de comando.

TECLA D

Esta tecla deleta o carácter que está à direita do cursor. Os caracteres que forem deletados serão incluídos entre pontos de exclamação.

EXEMPLOS:

Na linha 20 que estamos usando, suponhamos que deseja trocar a palavra "EXEMPLO" pela palavra "TESTE".

```
EDIT 20  RETURN
20 -
```

Aperte a tecla de espaço até alcançar a primeira aspa:

```
20 FOR R=0 TO 50:PRINT"-
```

Aperte a tecla D sete vezes (ou então 7D).

Se apertou 7D o vídeo mostrará:

```
20 FOR R=0 TO 50:PRINT "EXEMPLO! -
```

Agora pode usar o comando I para inserir a palavra "TESTE"

TECLA A

Esta tecla cancela todas as modificações feitas na linha, enquanto você está no modo de edição.

Por exemplo, se deletou algum carácter e depois desistiu, aperte a tecla A para reiniciar a edição da linha.

NOTA - A tecla A só funciona enquanto você está editando a linha. Isto é, se deletou algum carácter e apertou RETURN, a modificação é irrecuperável.

Por isso é sempre aconselhável usar o comando L para verificar se a linha está conforme desejado, antes de terminar a edição com a tecla RETURN.

-, Lembre-se de que deve sair de qualquer subcomando (através de SHIFT ↑) para poder usar A.

TECLA X

Este comando move o cursor para o fim da linha e entra automaticamente no modo de inserção.

EXEMPLO:

Suponha que na linha 20 você pretenda acrescentar uma nova instrução.

```
EDIT 20
20 -
```

Aperte a tecla X

```
20 FOR R=0 TO 50:PRINT "EXEMPLO 1":NEXT -
```

Você está no modo de inserção e pode colocar a instrução desejada. Por exemplo, escreva: PRINT "FIM" no fim da linha. Agora saia do comando X (através de SHIFT ↑) e aperte L, para verificar a linha.

TECLA H

Quando aperta esta tecla você deleta tudo que está à direita do cursor e entra automaticamente no modo de inserção.

TECLA C

Este comando permite que se troquem caracteres começando na posição do cursor.

Se apertar C sem um número precedente, o computador permite que troque o carácter que está na posição do cursor.

Se apertar C precedido de um número, o computador permite que troque tantos caracteres quantos os especificados pelo número que precede C.

Depois que trocou os caracteres, retorna automaticamente para o modo de edição, isto é, não é necessário apertar SHIFT ↑ para sair deste comando.

EXEMPLOS:

Suponha que no nosso exemplo você deseja trocar o 50 por 63.

```
EDIT 20
20 -
```

Aperte a tecla de espaço até o carácter que precede o 5

```
20 FOR R=0 TO -
```

Agora aperte 2C, para dizer ao computador que deseja trocar dois caracteres. Coloque, então, o número 63.

```
20 FOR R=0 TO 63 -
```

Quando apertou 2C o computador deleta dois caracteres e fica esperando dois novos caracteres. Note que o número de caracteres deletados é sempre igual ao número de caracteres inseridos.

A tecla '<---' não funciona com o comando C. Portanto, ela não deve ser usada. Se cometeu um erro durante o comando C, edite a linha novamente para corrigi-la.

TECLA E

Termina a edição e mantém todas as alterações feitas. Antes de usar a tecla E, deve sair de qualquer outro comando através de SHIFT ↑.

TECLA Q

Termina a edição e cancela todas as alterações feitas. Antes de usar a tecla E, deve sair de qualquer outro comando através de SHIFT ↑.

TECLA S

Formato para este comando: nSc

Este comando move o cursor até a n-ésima ocorrência do carácter c.

POR EXEMPLO:

3SP - o computador move o cursor até a terceira ocorrência do carácter P

2S: - o computador move o cursor até a segunda ocorrência do carácter:

1SA ou SA - o computador move o cursor até a primeira ocorrência do carácter A. Note que, se desejar a primeira ocorrência, o 1 é opcional.

Após mover o cursor até o ponto desejado, você pode executar qualquer outro subcomando.

TECLA K

Formato para este comando: nKc

Este comando diz ao computador para deletar a partir do cursor até a n-ésima ocorrência do carácter c, exclusive.

EXEMPLO:

2KP - o computador deleta tudo que está entre o cursor e a segunda ocorrência do carácter P

1K: ou K: - o computador deleta tudo que está entre o cursor e a primeira ocorrência do carácter:. Note que 1 é opcional.

Supondo que a linha 20 esteja desta maneira:

```
20 FOR R=0 TO 63: PRINT "EXEMPLO 1" : NEXT : PRINT "FIM"
```

Se desejar tirar o loop desta linha, faça:

```
EDIT 20
20 -
```

Aperte 2KP

```
20 ! FOR R=0 TO 63 : PRINT "EXEMPLO" : NEXT : ! PRINT "FIM"
```

Aperte L

```
20 PRINT "FIM"
```

NOTA - A letra P maiúscula é diferente da letra p minúscula. Por isso usamos 2KP. Se a palavra exemplo estiver escrita em maiúsculas devemos usar 3KP.

EDIT tem por finalidade entrar numa modalidade de edição em uma linha especificada. O objetivo deste comando é permitir alterações em linhas já escritas do programa trazendo-as para serem editadas novamente. É um comando especial utilizado por alguns computadores (tais como os que utilizam BASIC Microsoft), que permite editar a linha do programa especificada pelo comando EDIT. Assemelha-se aos comandos RUN e LIST, pelo fato de que, se não for seguido por nenhum número, a primeira linha do programa é implicada automaticamente em determinados computadores.

EXEMPLO 1

```
100 REM USANDO O COMANDO EDIT
```

```
110 PRINT "PODE ESTE PROGRAMA SER MODIFICADO"
120 PRINT "PELO COMANDO EDIT ?"
130 END
```

Depois de carregar o programa, tecle EDIT 110 para determinar se o computador possui a característica EDIT. O computador deve imprimir o número 110 seguido possivelmente por um cursor. Isto indica que o computador está no modo EDIT, estando pronto para modificar a linha 110.

O comando EDIT talvez chame seu editor, mas terá que verificar o sistema manual da máquina para ver como realizar a edição e depois voltar para a linguagem BASIC. Em certos casos é tão fácil como teclar RETURN ou ENTER. Em outras máquinas (principalmente as de tempo compartilhado que aceitam muitas linguagens diferentes) é necessário toda uma série de comandos para entrar no "editor" e sair dele.

VARIAÇÕES DE USO

Existem muitas versões de editores de textos, caracteres e linhas. Cada um deles fala a sua própria "linguagem", que não é BASIC. Esta Enciclopédia BASIC não abrange, portanto, as linguagens editoras.

Instrução **ELSE**

SINTAXE GERAL

Número de linha IF...THEN...ELSE instrução ou número de linha.

Esta instrução complementa o comando IF...THEN, não sendo disponível nos computadores que seguem a linha APPLE II.
ELSE é utilizado para executar uma instrução alternativa onde não seja satisfeita a condição relacional de uma instrução IF-THEN.
IF A=7 THEN 500 ELSE STOP
por exemplo instrui o computador a se desviar para a linha 500 se A for igual a 7, mas para STOP (parar) se A não for igual a 7.

EXEMPLO 1

```
100 REM PROGRAMA DE TESTE 'ELSE'
110 A=1
120 IF A < 7 THEN 150 ELSE GOTO 180
130 PRINT "ELSE NAO APROVOU"
140 GOTO 199
150 PRINT A;
160 A=A+1
170 GOTO 120
180 PRINT "ELSE FOI APROVADO"
199 END
```

EXECUÇÁO DO PROGRAMA

```
RUN
1 2 3 4 5 6 ELSE FOI APROVADO
```

SE SEU COMPUTADOR NÁO TIVER ESTA INSTRUÇÁO
Se seu computador não tiver a instrução ELSE, isto pode ser simulado no Exemplo 1 mudando a linha 120 para:

```
120 IF A < 7 THEN 150
```

e acrescentando a seguinte linha:

```
125 GOTO 180
```

VARIAÇÓES DE USO
Não se tem conhecimento de nenhuma.

VEJA TAMBÉM
IF-THEN, GOTO

Instrução **END**

SINTAXE GERAL

Número de linha END

END é utilizado como instrução no modo programa. END interrompe a execução do programa e retorna ao nível de comando. O comando END é aceito na maioria dos microcomputadores para estabelecer o fim físico do programa. Nos microcomputadores que seguem a linha SINCLAIR END é substituído por STOP.

Nos computadores que seguem a linha APPLE, o comando END é opcional se estiver usando o BASIC APPLESOFT.

Em Integer BASIC, o comando END deve ser a última instrução executada ou a mensagem NO END ERR aparece no vídeo.

END pode ser colocado em qualquer lugar do programa para terminar sua execução. Ao contrário da instrução STOP, END termina a execução normalmente (sem que seja impressa a mensagem BREAK). END é geralmente usado para forçar o término do programa em um ponto que não seja o fim do programa.

OBS: Em muitos computadores a instrução END é facultativa.

EXEMPLO 1

```
100 REM USANDO INSTRUCAO END
110 LET A = 1
120 LET A = A + 1
130 PRINT A;
140 END
```

EXECUÇÃO DO PROGRAMA

```
RUN
2
```

EXEMPLO 2

```
200 REM USANDO END CONTROLANDO O PROGRAMA
210 INPUT X1
220 INPUT X2
230 GOSUB 2000
240 PRINT "O PRODUTO DE "X1 "X" X2 " E' IGUAL A "; P
250 END
2000 P = X1 * X2
2010 RETURN
```

EXECUÇÃO DO PROGRAMA

```
RUN
? 5 (se entrar com valor 5)
? 9 (se entrar com valor 9)
O PRODUTO DE 5 X 9 E' IGUAL A 45
```

OBS.: A instrução END na linha 250 encerra o programa principal (provocando uma mensagem de erro RG return sem Gosub). Portanto, a linha 2010 só pode ser admitida por uma instrução de desvio, tal como 230 GOSUB 2000.

VARIAÇÕES DE USO

Não se tem conhecimento de nenhuma.

VEJA TAMBÉM

STOP

Operador EQ

EQ é utilizado em alguns computadores (tais como o T.I. 990) como palavra facultativa para o sinal de "igual a" onde é utilizada como operador relacional.

Não pode ser utilizado para atribuir um valor a uma variável.

Por isso que se utiliza na linha 110 o sinal =

Para mais informações veja-se = (igual).

EXEMPLO 1

```
100 REM USANDO O OPERADOR EQ (EQUAL)
110 X = 15
120 IF X EQ 10 THEN 150
130 PRINT "O OPERADOR EQ NAO FOI APROVADO"
140 GOTO 160
150 PRINT "O OPERADOR EQ FOI APROVADO"
160 END
```

EXECUÇÃO DO PROGRAMA

```
O OPERADOR EQ FOI APROVADO
```

VARIAÇÕES DE USO

Não se tem conhecimento de nenhuma.

VEJA TAMBÉM

=, <>, IF-THEN, GE, GT, LE, LT, NE, <, >, <= e >=

Comando/Instrução **ERASE**

SINTAXE GERAL

Número de linha ERASE lista de variáveis.

ERASE é aceito em alguns microcomputadores podendo ser utilizado como comando ou como instrução no modo programa.

Como comando ERASE é utilizado para apagar um programa da memória. Equivalente a NEW ou SCRATCH utilizado em outros computadores. Para testar ERASE no seu computador, dê entrada num programa curto tal como:

```
100 REM ESTE E  UM PROGRAMA PARA TESTAR ERASE
110 END
```

Tecle LIST para ter certeza de que o programa está na memória do seu computador.

Tecle ERASE, e em seguida tecle LIST de novo. Se ERASE surtiu efeito, o programa deve ter sido apagado.

Alguns interpretadores utilizam ERASE com a finalidade de eliminar matrizes de um programa e utilizar o espaço de armazenamento que ocupava.

As matrizes podem ser redimensionadas após serem eliminadas, ou pode-se usar o espaço de memória anteriormente alocado pela matriz para outras finalidades.

Se se tentar redimensionar uma matriz sem antes eliminá-la, ocorre um erro.

EXEMPLO 1

```
100 REM USANDO ERASE COMO INSTRUCAO
110 DIM B(20)
120 FOR X = 1 TO 15
130 B(X) = X
140 NEXT X
150 ERASE B
160 PRINT "ERASE FOI APROVADO SE 0 = ";B
170 DIM B(5,5)
180 B(5,5) = 2
190 PRINT "ERASE APROVOU ";B(5,5);" TESTES"
200 END
```

EXECUÇÃO DO PROGRAMA

```
ERASE FOI APROVADO SE 0 = 0
ERASE APROVOU 2 TESTES
```

VEJA TAMBÉM

NEW, SCRATCH, DIM, LIST

Função **ERL • ERRL**

SINTAXE GERAL

Número de linha instrução ERL

ERL é uma função devendo ser utilizada no modo programa, devendo ser precedida por uma instrução PRINT ou IF...THEN

ERL é uma função aceita nos microcomputadores que seguem a linha TRS-80 mod.I e III, não sendo disponível nos da linha APPLE II.

ERL devolve o número da linha com erro.

```
EXEMPLO 1
100 REM USANDO A FUNCAO ERL
110 ON ERROR GOTO 160
120 INPUT "COLOQUE UM NUMERO QUALQUER";N
130 PRINT "O INVERSO DESTE NUMERO E' ";1/N
140 PRINT "A RAIZ QUADRADA DESTE NUMERO E' ";SQR(N)
150 GOTO 120
160 IF ERL = 130 THEN 180 ELSE IF ERL = 140 THEN 200
170 PRINT "O ERRO ACONTECEU FORA DA LINHA 13 E 140 : END
180 PRINT "INFINITO"
190 RESUME 140
200 PRINT "IMAGINARIA"
210 N = -N
220 PRINT SQR(N)
230 RESUME 150

EXECUÇÃO DO PROGRAMA
COLOQUE UM NUMERO QUALQUER? 0
O INVERSO DESTE NUMERO E INFINITO
A RAIZ QUADRADA DESTE NUMERO E' 0
COLOQUE UM NUMERO QUALQUER? 3
O INVERSO DESTE NUMERO E .333333
A RAIZ QUADRADA DESTE NUMERO E 1.73205
COLOQUE UM NUMERO QUALQUER? -7
O INVERSO DESTE NUMERO E -.142857
A RAIZ QUADRADA DESTE NUMERO E IMAGINARIA SQR(N)
COLOQUE UM NUMERRO QUALQUER?
```

A função ERL é utilizada em conjunto com a instrução ON-ERROR para identificar o último número de linha em que ocorreu um erro.

A função ERL inicializa no valor numérico da linha numérica de 65535 (valor máximo de dois bytes). Ocorrendo um erro, ERL muda para o número de linha em que ocorreu o erro.

O número de linha contido na função ERL muda cada vez que ocorre um erro numa linha diferente.

Por se utilizar ERL em rotinas de erro, é possível identificar a linha de erro e tomar a ação apropriada.

```
EXEMPLO 2
200 REM USANDO A FUNCAO ERL
```

```
210 ON ERROR GOTO 290
220 PRINT "DE ENTRADA NO NUMERO 10, 20, OU 30";
230 INPUT N
240 A = 10/(N-10)
250 A = 10/(N-20)
260 A = 10/(N-30)
270 PRINT "O NUMERO ";N;"NAO CAUSOU ERRO"
280 GOTO 220
290 PRINT "UM ERRO ACABA DE OCORRER NA LINHA "; ERL
300 RESUME 220
310 END
```

EXECUÇÃO DO PROGRAMA

```
DE ENTRADA NO NUMERO 10, 20 OU 30? 10
UM ERRO ACABA DE OCORRER NA LINHA 240
DE ENTRADA NO NUMERO 10, 20 OU 30? 20
UM ERRO ACABA DE OCORRER NA LINHA 250
DE ENTRADA NO NUMERO 1Q, 20 OU 30? 30
UM ERRO ACABA DE OCORRER NA LINHA 260
DE ENTRADA NO NUMERO 10, 20 OU 30?
```

ORTOGRAFIA ALTERNATIVA

Os computadores 35, 45 e 85 da Hewlett Packard utilizam ERRL.

Nos microcomputadores que seguem a linha APPLE deve-se usar:

```
X = PEEK(218) + PEEK(219) * 256
```

VARIAÇÕES DE USO

Não se tem conhecimento de nenhuma

VEJA TAMBÉM

ERROR, ON-ERROR-GOTO, RESUME

Função ERR • ERRN

SINTAXE GERAL

Número de linha função ERR

A função ERR é similar ao ERL, exceto que ERR fornece um valor que está relacionado com o código de erro através da seguinte fórmula:

ERR/2+1 = código de erro

O código de erro contido na função ERR muda cada vez que ocorre um erro diferente. Por se utilizar ERR em rotinas para captar erros, é possível identificar o tipo de erro que ocorreu e tomar as medidas apropriadas. Consulte o manual do seu computador para uma listagem dos diversos códigos de erro.

```
EXEMPLO 1
100 REM USANDO A FUNCAO ERR
110 DIM X(8)
120 CLEAR
130 ON ERROR GOTO 190
140 PRINT "DE ENTRADA NUM NUMERO DE AMOSTRA";
150 INPUT N
160 X (N) = 10/N
170 PRINT "O NUMERO ";N;" NAO CAUSOU ERRO"
180 GOTO 140
190 IF ERR = 9 THEN 220
200 IF ERR = 11 THEN 250
210 GOTO 270
220 PRINT "O NUMERO ";N;" E' DEMASIADO GRANDE"
230 PRINT "UTILIZE UM NUMERO ENTRE 1 E 8"
240 RESUME 120
250 PRINT "O NEMOR NUMERO PERMITIDO E' 1"
260 RESUME 0
270 PRINT "O NUMERO ";N;" CAUSOU UM CODIGO DE ERRO DE"; ERR
280 END
```

```
EXECUÇÃO DO PROGRAMA
DE ENTRADA NUM NUMERO DE AMOSTRA? 1
O NUMERO 1 NAO CAUSOU ERRO
DE ENTRADA NUM NUMERO DE AMOSTRA? 9
O NUMERO 9 NAO CAUSOU ERRO
DE ENTRADA NUM NUMERO DE AMOSTRA? 11
O NUMERO 11 CAUSOU UM CODIGO DE ERRO DE 16
```

ORTOGRAFIA ALTERNATIVA

Alguns microcomputadores HEWLETT PACKARD utilizam ERRN.

Os microcomputadores que seguem a linha APPLE II utilizam PEEK(222).

VARIAÇÕES DE USO

O BASIC TRS-80 Nível II armazena um valor na função ERR que não é igual ao código real de erro. Para converter os valores armazenados na função ERR para o código de erro em si, divida o

valor ERR por dois e acrescente um.

ERR fornece um valor que está relacionado com o código do erro, através da seguinte fórmula:

ERR/2+1= CODIGO DE ERRO

OBSERVAÇÃO:

Se desejar mais um exemplo tecle novamente o Exemplo 1 da função ERL trocando apenas as linhas 160 e 170 por:

```
160 IF ERR/2+1 = 11 THEN 180 ELSE IF ERR/2+1 = 5 THEN 200
170 PRINT "ERRO INESPERADO": END
```

```
EXECUÇÃO DO PROGRAMA
DE ENTRADA NUM NUMERO DE AMOSTRA? 8
ERRO INESPERADO
```

OBSERVAÇÃO:

CODIGO 11 - DIVISAO POR ZERO (0)

CODIGO 5 - PARAMETRO INCOMPATIVEL (PI)

VEJA TAMBÉM

ERL, ON-ERROR, RESUME, DIM, CLEAR

Comando/Instrução **ERROR**

SINTAXE GERAL

Número de linha ERROR

ERROR é reconhecido por alguns microcomputadores podendo ser utilizado como comando no modo direto ou como instrução no modo programa.

ERROR (cod) simula erro especificado pelo código. Este comando permite a simulação de um erro durante a execução de um programa. A maior utilização desta instrução é para testes de rotinas que são atingidas pela instrução ON ERROR GOTO.

EX:

```
100 ERROR 2
```

Consulte o manual do seu computador para verificar a natureza do erro.

EXEMPLO 1

```
100 INPUT X
110 IF X > 32000 THEN ERROR 7
120 END
```

Ao se atribuir à variável X um valor superior a 32000, satisfaz-se a condição da instrução IF-THEN na linha 110 e o computador gera a mensagem de ERROR.

ON ERROR IN 110

As variáveis não podem ser utilizadas como códigos de ERROR.

Cada código deve ser especificado por um número inteiro real de código de erro. Se um comando ERROR especificar um código para o qual não tenha sido definida uma mensagem de erro, o BASIC respondde com a mensagem ERRO não codificado. A execução de um comando ERROR para o qual não haja uma rotina de detecção de erro, faz com que se imprima uma mensagem de erro e se interrompa a execução.

ERROR pode também ser lançado como comando para testar códigos de erro específicos. Veja o manual do seu computador para uma listagem de suas mensagens de erro.

EXEMPLO 2

```
200 REM USANDO ERROR
210 PRINT "ERROR APROVOU SE FOR IMPRESSA MENSAGEM"
220 PRINT "DE ERRO OS OU COM ESPACO PARA STRINGS ESGOTADO"
230 ERROR 14
240 END
```

EXECUÇÃO DO PROGRAMA (de forma típica)

```
ERROR APROVOU SE FOR IMPRESSA MENSAGEM
DE ERRO OS OU COM ESPACO PARA STRINGS ESGOTADO
? OS ERROR NA 230
```

VARIAÇÓES DE USO

Náo se tem conhecimento de nenhuma.

VEJA TAMBÉM

ON-ERROR-GOTO, RESUME, ERR, ERL

Comando EXEC

SINTAXE GERAL

EXEC nomearq[,Rn] [,Dn] [,Sn] [,Vn]

O comando EXEC executa um arquivo de texto em disco como se cada carácter fosse digitado. É um comando do DOS 3.3 sistema operacional de disquete utilizado nos microcomputadores que seguem a linha APPLE II.

O comando EXEC é semelhante ao comando RUN exceto que o nome do arquivo é um arquivo-texto (arquivo de dados) contendo comandos do DOS e instruções em BASIC como se eles tivessem sido teclados. Isto permite montar arquivos que podem controlar os micros que seguem a linha APPLE. Um arquivo de texto a ser usado com EXEC deve conter combinação de comandos BASIC, comandos DOS e linhas de programa. Quando EXEC é executado, a primeira linha do arquivo especificado é lida do disco. Se for comando, ele será executado imediatamente; se for linha de programa irá para a memória, exatamente como se tivesse sido entrado pelo teclado.

Um arquivo EXEC pode ser usado para entrar num programa inteiro, listá-lo, executá-lo, guardá-lo no disco, trazê-lo de volta, e qualquer outra coisa que se pode fazer a partir do teclado. EXEC é um comando do DOS, requer PRINT e CTRL-D em modo programado. Pode-se até usar um arquivo EXEC para criar e executar outro arquivo EXEC.

O parâmetro R, se presente, especifica qual campo do arquivo será executado primeiro. Quando usado com o EXEC, o parâmetro R sempre conta a partir do início do arquivo: o primeiro campo do arquivo é o campo 0, o segundo é o 1 etc. O número que segue o R deve ser uma constante inteira entre 0 e 32767. Se Rn especificar o último campo após o fim de arquivo, nada acontece. Se ele especificar dois ou mais campos depois do fim de arquivo, aparece a mensagem END OF DATA.

Se uma declaração INPUT é executada enquanto o arquivo EXEC está aberto, a resposta é tirada do arquivo EXEC.

Quando a última linha de um arquivo foi executada, o arquivo EXEC fecha a si mesmo. Quando um comando EXEC é encontrado num arquivo de controle EXEC, o arquivo original é fechado e qualquer outro comando dele é ignorado; o novo arquivo EXEC é aberto e passa a operar normalmente.

Se o arquivo não existe no drive Dn do slot Sn, aparece a mensagem (ARQUIVO NAO ENCONTRADO) de erro. Se o disco no drive Sn não tem volume Vn, aparece um erro VOLUME MISMATCH.

DN,SN, e VN podem ser especificados em qualquer ordem. Se Dn, e Sn são omitidos, o drive referenciado por último é adotado. V0 é usado por seqüência de Vn. Também n pode estar ausente, quando se adota D0, S0 e V0.

Se o comando EXEC abrir um arquivo, o comando CLOSE não poderá fechá-lo. Quando um arquivo EXEC completa todos os seus comandos, ele se fecha e pára a execução.

VARIAÇÕES DE USO

Somente um comando EXEC pode ser ativado por vez.

VEJA TAMBÉM

CLOSE, RUN

Instrução **EXCHANGE**

SINTAXE GERAL

Número da linha instrução EXCHANGE variáveis.

EXCHANGE deve ser utilizado no modo programa.

EXCHANGE é uma instrução disponível em apenas algumas versões do BASIC (de microcomputadores americanos e europeus), que permuta os valores de duas variáveis ou de dois elementos de um arranjo.

EXCHANGE X,Y resulta em fazer o valor original de X ficar armazenado em Y e o valor original de Y ficar armazenado em X. EXCHANGE é muito útil para dispor valores de um arranjo em ordem ascendente ou descendente.

EXEMPLO 1

```
100 REM USANDO A INSTRUCAO EXCHANGE
110  PRINT "DE ENTRADA EM DOIS VALORES (SEPARADOS POR VIRGULAS)"
120 INPUT X,Y
130 IF X<=Y THEN 150
140 EXCHANGE X,Y
150 PRINT X;" E' MENOR DO QUE ";Y
160 GOTO 110
170 END
```

EXECUÇÃO DO PROGRAMA

```
RUN
DE ENTRADA EM DOIS VALORES (SEPARADOS POR VIRGULAS)
? 13,10
10 E MENOR DO QUE 13
DE ENTRADA EM DOIS VALORES (SEPARADOS POR VIRGULAS)
? 8,1
1 E MENOR DO QUE 8
DE ENTRADA EM DOIS VALORES (SEPARADOS POR VIRGULAS)
?
```

A versão MAXBASIC utiliza o sinal de igual em dobro (==) para intercambiar o conteúdo de duas variáveis do mesmo tipo.

140 X == Y, por exemplo, é igual à linha 140 do Exemplo 1.

SE O SEU COMPUTADOR NÃO TIVER ESTA INSTRUÇÃO

Se EXCHANGE não funcionar no seu computador, trate de utilizar SWAP na linha 140. Não estando disponíveis nem uma nem outra instrução, os valores podem ser intercambiados substituindo a linha 140 e incluindo as linhas 138 e 142.

```
138 T=X
140 X=Y
142 Y=T
```

Instrução EXIT

SINTAXE GERAL

Número da linha, instrução EXIT

EXIT é uma instrução utilizada por algumas versões BASIC (de microcomputadores americanos e europeus) para EXIT de um laço FOR-NEXT antes que esse laço tenha completado o número especificado de ciclos. EXIT transfere o controle do programa para o número de linha designado e cancela o laço FOR-NEXT. O valor do contador de laço nessa ocasião continua a estar disponível para uso no resto do programa.

```
EXEMPLO 1
100 REM USANDO A INSTRUCAO EXIT
110 PRINT "DE ENTRADA NUMA PALAVRA - TECLE PRONTO PARA SAIR"
120 FOR X=1 TO 500
130 INPUT A$
140 IF A$ = "PRONTO" THEN EXIT 190
150 PRINT "OUTRA";
160 NEXT X
170 PRINT "EXIT NAO APROVOU;"
180 GOTO 200
190 PRINT "EXIT FOI APROVADO, PRONTO FOI PALAVRA NUMERO";X
200 END
```

```
EXECUÇÃO DO PROGRAMA
RUN
DE ENTRADA NUMA PALAVRA - TECLE PRONTO PARA SAIR
?INICIO
OUTRA ?VERIFICAR
OUTRA ?SAIR
OUTRA ?PRONTO
EXIT FOI APROVADO, PRONTO FOI PALAVRA NUMERO 4
```

SE O COMPUTADOR NÃO TIVER ESSA INSTRUÇÃO

GOTO pode substituir EXIT na maioria dos casos. Alguns computadores não saberão qual o laço que está em atividade se outro FOR X= seguir o laço que já efetuou o EXIT dessa maneira. Nesses computadores, substitua a linha 140 por:

```
140 IF A$ <> PRONTO THEN 150
141 IF A$ = PRONTO THEN 190
142 Y = X:REM ARMAZENA O VALOR ATUAL DO CONTADOR DE LACO
144 X = 200: REM FACA X TER VALOR ACIMA DO LIMITE DO LACO
146 GOTO 160
```

e acrescente as linhas 165 e 185.

```
165 IF X=200 THEN 185
185 X = Y
```

Estas linhas permitem ao laço terminar "normalmente" antes de prosseguir com o resto do programa.

Veja também
FOR, NEXT, GOTO

Função EXP

SINTAXE GERAL

Número da linha LET variável = EXP (exprnm)

EXP(exprnm) é uma função utilizada no modo programa devendo ser precedida por uma instrução LET ou PRINT; esta função calcula a base do logaritmo natural (2,718281828) e eleva à potência declarada na (exprnm).

Esta função é disponível nos computadores que seguem a linha APPLE II na versão APPLESOFT e nos que seguem a linha TRS-80 em linguagem de nível II.

A função EXP (n) calcula o valor de base do logaritmo natural e (2,718282...) elevado à potência de (n).

É exatamente o contrário do que ocorre ao se utilizar a função LOG.

X = 2,718282 * 2,718282 * 2,718282

O valor (n) pode ser escrito como número ou como variável numérica.

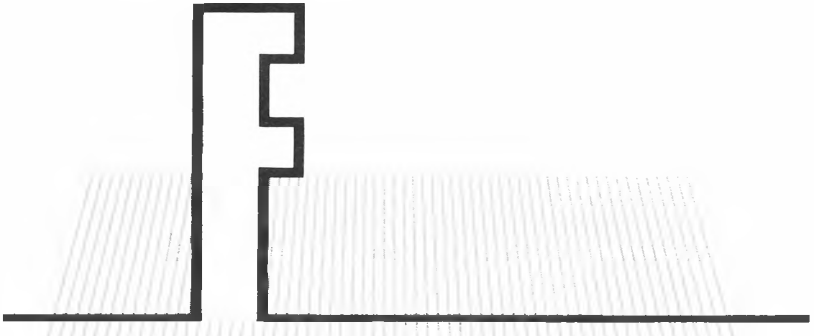
```
EXEMPLO 1
100 REM USANDO A FUNCAO EXP
110 N = 3.71506
120 E = EXP (n)
130 PRINT "SE O EXPONENCIAL NATURAL DE ";N;" FOR ";E
140 PRINT "A FUNCAO EXP FOI APROVADA"
150 END

EXECUÇÃO DO PROGRAMA
RUN
SE O EXPONENCIAL NATURAL DE 3.71506 FOR 41.0610507
A FUNCAO EXP FOI APROVADA

EXP (X)
Devolve o número (Neperiano) e elevado à potência X
X deve ser <=87,3365.
Ex: 10 X = 7
    20 PRINT EXP(X-1)
    RUN
    403.428793
EXP fornece o exponencial natural de X, que é EX.
Esta é a função inversa da logarítmica, assim:
X = EXP (LOG(X))
EX: 100 PRINT EXP (-X)
```

A função trigonométrica EXP, é geralmente calculada até oito dígitos de exatidão.

VEJA TAMBÉM
LOG, LOG10, CLG



FAST • FILL • FIX • FLASH
FLOW • FN • FNEND
FOR . . . TO . . . STEP • FRAC • FRA
FRE • $FREE • FRE (0)
FRE (STRING) • FP

Comando **FAST**

SINTAXE GERAL

FAST

FAST é uma comando usado no modo direto, sendo utilizada pelos microcomputadores que seguem a linha SINCLAIR. Os microcomputadores que seguem a linha SINCLAIR podem trabalhar em duas velocidades FAST(rápido) ou SLOW(lento). Quando se liga pela primeira vez o computador, este trabalha no modo SLOW.

Contudo, o computador pode trabalhar numa velocidade quatro vezes maior, em que o computador se esquece da imagem exceto quando não tem mais nada a fazer. Para ver como isto funciona escreva

FAST

Agora sempre que pressionar uma tecla, o vídeo piscará, porque o computador deixa de dar a imagem enquanto resolve a tecla que se pressionou.

EX:

```
100 FOR N = 0 TO 255
110 PRINT CHR$ N;
120 NEXT N
```

Quando passar todo o programa, o vídeo ficará de uma cor acinzentada até que o resultado da instrução PRINT apareça no vídeo.

O vídeo aparece também durante uma instrução INPUT, enquanto o computador espera que se introduza o valor para INPUT.

Tente este programa:

```
100 INPUT A
110 INPUT A
120 GOTO 10
```

Para se voltar ao normal (computação e imagem), escreva:

SLOW

É apenas uma questão de gosto, o fato de se querer mudar o modo de computação e imagem. Normalmente usa-se o modo rápido quando:

a) o seu programa contém uma série de cálculos numéricos, em especial se não tem muitas imagens, e no caso de as haver estas não precisam se demorar muito tempo ou

b) estiver fazendo um programa longo. Já deve ter reparado que a listagem volta a ser feita sempre que se introduz uma nova linha, o que se torna cansativo.

FAST começa o modo rápido, em que a página de projeção é projetada apenas no fim do programa, ou enquanto os dados INPUT estao para ser introduzidos, ou durante uma pausa.

OBS: Página de projeção é a área dentro do computador onde ele armazena a figura do vídeo (tv) concebida engenhosamente de modo a guardar espaço só para o que foi digitado até aí.

VARIAÇÕES DE USO

Não se tem conhecimento de nenhuma.

VEJA TAMBÉM

SLOW

Instrução FILL

SINTAXE GERAL

Número de linha, variáveis

FILL é uma instrução utilizada geralmente no modo programa e reconhecida por alguns interpretadores e microcomputadores americanos e europeus.

FILL assinala a determinado byte na memória do microcomputador um valor de número inteiro entre 0 e 255 (valor máximo de 8 bits).

FILL 3000,15, por exemplo, "enche" o endereço de memória 3000 com o número decimal .15.

Os microcomputadores variam de acordo com a quantidade de memória e de endereços de memória disponíveis para serem FILL-ados, sem apagar a memória dedicada a outras finalidades.

A função EXAM pode ser utilizada para inspecionar aquilo que FILL colocou na memória. Aguns computadores, no lugar desta, utilizam PEEK ou FETCH.

Consulte o manual de instruções do seu computador antes de rodar o Exemplo 1, para determinar que os endereços de memória de 18368 até 18380 são locais de memória não críticos.

EXEMPLO 1

```
100 REM USANDO A INSTRUCAO FILL
110 FOR A = 18368 TO 18380
120 READ B
130 FILL A,B
140 NEXT A
150 FOR A = 18369 TO 18380
160 B = EXAM(A)
170 PRINT CHR$(B);
180 NEXT A
190 DATA 70,73,76,76,70,79,73,65,80,82,79,86,65,68,79
200 END
```

EXECUÇÃO DO PROGRAMA

```
RUN
FILL FOI APROVADO
```

VARIAÇÕES DE USO

Não se tem conhecimento de nenhuma.

VEJA TAMBÉM

POKE, PEEK, USR, SYSTEM

Função **FIX**

SINTAXE GERAL

Número de linha, LET variável = função FIX(X)

FIX(X) é uma função usada geralmente no modo programa, devendo ser precedida por uma instrução LET ou PRINT.

É reconhecida pelos interpretadores BASIC dos commputadores que seguem a linha TRS-80, para fornecer a parte inteira do argumento (X) da função FIX(X); nos microcomputadores que seguem a linha APPLE II a função equivalente é INT(X)

A função FIX é utilizada para remover todos os números para a direita da vírgula decimal. Assemelha-se pelas suas operações à função INT, exceção feita ao fato de que FIX não arredonda para baixo os números negativos.

```
100 PRINT FIX(5.6)
110 PRINT FIX(-5.6)
```

por exemplo, imprime os números 5 e -6

FIX tem capacidade para manipular qualquer número, pequeno ou grande, dentro dos limites do interpretador do computador.

EXEMPLO 1

```
100 REM USANDO A FUNCAO FIX
110 X = -15.4365
120 B = FIX(X)
130 PRINT "FIX FOI APROVADO SE ";X; " E MUDADO PARA ";B
140 END
```

EXECUÇÃO DO PROGRAMA

```
FIX FOI APROVADO SE -15.4365 E  MUDADO PARA -15
```

SE O SEU COMPUTADOR NÃO TIVER ESTA FUNÇÃO

Se o seu computador não tiver a capacidade de aplicar a função FIX, mas possui as funções ABS, INT e SGN, a linha 120 do Exemplo 1 pode ser substituída por:

```
120 B = SGN(X)*INT(ABS(X))
```

VARIAÇÕES DE USO

Não se tem conhecimento de nenhuma

VEJA TAMBÉM

INT, ABS, SGN

Instrução/Comando **FLASH**

SINTAXE GERAL

Número de linha, FLASH

FLASH é utilizado nos microcomputadores com BASIC APPLESOFT. que seguem a linha Apple II como comando no modo direto ou como instrução no modo programa; para colocar a tela no modo FLASH, esta declaração do APPLESOFT faz os caracteres do vídeo piscarem (torna pulsante qualquer conteúdo impresso).

Nesse modo, toda a saída por PRINT é exibida alternativamente sob forma de caracteres brancos sobre fundo preto e em seguida como caracteres pretos sobre fundo branco.

Para restaurar a exibição na tela para seu modo normal, sem centelhamento, teclar NORMAL.

FLASH opera alterando um pouco o código ASCII padrão; por isso é que todos os caracteres enviados para o disquete em modo FLASH serão TRANSFERIDOS com códigos incorretos. Ao serem lidos de volta, aparecerão caracteres errados.

EXEMPLO 1

```
100 REM USANDO FLASH
110 FLASH
120 PRINT "ESTA E  UMA MENSAGEM PISCANTE"
130 END
```

Para rodar este programa, limpe a tela e tecle RUN

EXECUÇÃO DO PROGRAMA

```
RUN
ESTA E' UMA MENSAGEM PISCANTE (a tela deve estar piscando)
```

VARIAÇÕES DE USO

FLASH provoca um piscar em qualquer mensagem escrita na tela. Não há comando direto semelhante nos computadores que seguem a linha TRS-80.

FLASH não é disponível no INTEGER BASIC

VEJA TAMBÉM

NORMAL, INVERSE

Comando/Instrução **FLOW**

SINTAXE GERAL

Número de linha FLOW

FLOW é utilizado como comando no modo direto ou como instrução no modo programa, porém é reconhecido em apenas algumas versões BASIC.

FLOW ativa uma característica que imprime os números das linhas de um programa na tela, à medida que é executada. É um comando usado pelo BASIC da Micropolis.

FLOW é desligado com NOFLOW, e serve como auxílio na localização e retificação de erros em programas.

FLOW pode ser utilizado dentro de um programa em conjunto com NOFLOW, para rastrear apenas determinado trecho do programa.

```
EXEMPLO 1
100 REM USANDO FLOW
110 PRINT "FLOW VARRE TODA LINHA"
120 FLOW
130 GOTO 180
140 PRINT "ATE SER DESLIGADO POR"
150 NOFLOW
160 PRINT "NOFLOW"
170 PRINT "QUE SEGUE A INSTRUCAO FLOW"
180 GOTO 140
190 END

EXECUÇÃO DO PROGRAMA
RUN
FLOW VARRE TODA LINHA
<130> <170> QUE SEGUE A INSTRUCAO FLOW
<180> <140> ATE SER DESLIGADO POR
<150> NOFLOW

VARIAÇÕES DE USO
Não se tem conhecimento de nenhuma.

VEJA TAMBÉM
NOFLOW, TRACE, TRACE ON, TRON
```

Função **FN**

SINTAXE GERAL

Número de linha DEF FN varnm(exprnm)

FN é uma função usada geralmente no modo programa devendo ser precedida de uma instrução DEF.

FN chama uma função definida pelo usuário definida anteriormente. Não é disponível em Integer BASIC.

FN é uma função que permite utilizar um processo "definido pelo usuário" como se fosse função já embutida no computador. A função definida pelo usuário é denominada por uma letra que segue FN, sendo acompanhada por um ou mais valores contidos entre parênteses, tais como FNA(Y,N)

A instrução DEF define o processo que será executado ao se utilizar FN mais tarde.

Por exemplo:

```
100 DEF FNA(Y)=1/Y
110 PRINT FNA(N)
```

A função FN, neste exemplo, se denomina "A" (FNA), sendo definida na linha 100 como sendo a função N/Y. FNA é utilizada aqui para calcular o recíproco de qualquer expressão numérica (exceto aquela que tiver valor de zero, evidentemente).

A variável (N) que segue FNA é substituída pela "variável numérica" (Y, neste exemplo), na instrução DEF, cada vez que seja executada FNA. Pode ser utilizada no lugar de N qualquer variável ou expressão numérica válida.

varnm é o nome da função. O valor da expressão é adotado sempre que a variável fictícia aparece na definição da função, e o resultado da expressão é calculado.

Se na execução de um programa FN nome for executado antes de DEF FN, aparecerá uma mensagem de erro.

```
DEF FN A(N) = f(N)
FN A(X)
```

onde f(N) é uma expressão em N

Primeiro é definida a função FNA, usando-se DEF.

Definida a função esta pode ser usada na forma FNA (X) onde o argumento X pode ser qualquer expressão aritmética.

EXEMPLO 1

```
100 REM USANDO A FUNCAO FN
110 DEF FNY(A) = (A-32)*,5555555
120 PRINT "DE ENTRADA NUMA TEMPERATURA EM GRAUS FAHRENHEIT";
130 INPUT F
140 C = FNY(F)
150 PRINT F;"GRAUS FAHRENHEIT =" ;C;" GRAUS CELCIUS"
160 END
```

EXECUÇÃO DO PROGRAMA

```
RUN
DE ENTRADA NUMA TEMPERATURA EM GRAUS FAHRENHEIT? 70
70 GRAUS FAHRENHEIT = 21,1111 GRAUS CELCIUS
```

VARIAÇÕES DE USO

Alguns BASIC (tais como BASIC-PLUS da DEC) permitem que FN seja definida como função que atua sobre strings:

Exemplo: DEF FNP(A$,N) = RIGHT$(STRING$(N," ")+A$,N)

pode ser utilizado para "encher" um string de caracteres com espaços em vazio pela frente para fazer com que tenha extensão N.

SE O SEU COMPUTADOR NÃO TIVER ESTA FUNÇÃO

Se o seu computador não lhe permite definir funções dessa maneira, terá de escrever a função desejada em cada linha de programa em que haja necessidade dela.

VEJA TAMBÉM

DEF, FNEND

Instrução **FNEND**

SINTAXE GERAL

Número de linha FNEND

A instrução FNEND é utilizada em apenas alguns microcomputadores que têm capacidade de DEFinir e reDEFinir uma função em pontos diferentes através de um programa, terminado (END) o processo de DEFinição da função.

Cada instrução DEF que se estende além de uma linha deve terminar com instrução FNEND, sendo que o computador não pode se desviar dessas instruções DEF ou entrar nas mesmas antes de se executar a instrução FNEND.

EXEMPLO 1

```
100 REM USANDO A INSTRUCAO FNEND
110  PRINT  "DE  ENTRADA  NUM VALOR DE A QUE SEJA SUPERIOR OU
INFERIOR A 10";
120 INPUT A
130 DEF FNA(A)
140 FNA = A*2
150 IF A<10 THEN 170
160 FNA = A/2
170 FNEND
180 PRINT "O NOVO VALOR DE A E  ";FNA(A)
190 END
```

EXECUÇÃO DO PROGRAMA

```
RUN
DE  ENTRADA  NUM  VALOR  DE A QUE SEJA SUPERIOR OU INFERIOR A
10? 6
O NOVO VALOR DE A E' 12
```

VARIAÇÕES DE USO

Não se tem conhecimento de nenhuma

VEJA TAMBÉM

DEF, FN

Instrução/Comando **FOR... TO... STEP**

SINTAXE GERAL

FOR varnm = exprnm1 TO exprnm2 [Step exprnm3]
ou
Número de linha FOR variável numérica = valor inicial TO valor final STEP incremento.

FOR pode ser utilizado como comando no modo direto ou como instrução no modo programa
FOR...TO...STEP...:NEXT
FOR gera um Loop automático juntamente com NEXT. A instrução FOR...NEXT é aceita pela maioria dos microcomputadores. Nos micros que seguem a linha APPLE podemos usar FOR na versão APPLESOFT ou Integer BASIC utilizando a sintaxe geral
FOR varnm = exprnm1 TO exprnm2 [STEP exprnm3]
Quando a instrução FOR é executada pela primeira vez, é dado o valor exprnm1 à variável varnm. As instruções seguintes ao FOR são executadas até que uma instrução NEXT apareça. O valor da variável será incrementado de exprnm3 (ou 1 se não foi dado incremento para STEP).
Em seguida, o novo valor da variável já incrementado será comparado com o do valor final (exprnm2).
Se o sinal é positivo e o novo valor de varnn é menor ou igual a exprnm2, a execução continua para executar as instruções que seguem ao FOR. A mesma coisa acontece se o sinal de exprnm3 é negativo e o novo valor de varnm é maior ou igual a exprnm2.
Como a comparação ocorre depois de incrementar varnm, as instruções entre FOR e o NEXT serão executadas pelo menos uma vez. Se o NEXT não estiver presente, o Loop só será executado uma vez.
Em Integer BASIC varnm deve ser uma variável numérica e inteira, porém na versão BASIC APPLESOFT varnm pode ser uma inteira ou real. Em hipótese alguma não pode ser utilizado num FOR varnm alfanumérica (variável numérica).

OBSERVAÇÃO:
FOR varnm = valor inicial TO valor final STEP incremento
se o valor inicial < valor final o incremento do STEP deverá ser positivo.
Se o valor inicial > valor final o incremento do STEP terá que existir e ser negativo.

FOR tem por finalidade permitir que uma série de instruções seja executada um determinado número de vezes gerando um Loop automático.
A instrução FOR faz parte de uma instrução FOR-TO-NEXT, sendo utilizada para atribuir números a variáveis numéricas dentro da faixa especificada por FOR-TO.
O primeiro número logo após a instrução FOR é incrementado por 1 cada vez que se execute a instrução NEXT correspondente.
Ultrapassando-se o número que segue TO, prossegue a execução do programa na linha seguinte à instrução NEXT correspondente.

```
EXEMPLO 1
100 REM USANDO A INSTRUCAO FOR
110 FOR X = 1 TO 6
120 PRINT "SE ESTA MENSAGEM FOR IMPRESSA 6 VEZES"
130 NEXT X
140 PRINT
150 PRINT "A INSTRUCAO FOR FOI APROVADA"
160 END

EXECUÇÃO DO PROGRAMA
RUN
SE ESTA MENSAGEM FOR IMPRESSA 6 VEZES
SE ESTA MENSAGEM FOR IMPRESSA 6 VEZES
SE ESTA MENSAGEM FOR IMPRESSA 6 VEZES
SE ESTA MENSAGEM FOR IMPRESSA 6 VEZES
SE ESTA MENSAGEM FOR IMPRESSA 6 VEZES
SE ESTA MENSAGEM FOR IMPRESSA 6 VEZES
A INSTRUCAO FOR FOI APROVADA
```

Alguns computadores utilizam a instrução STEP para incrementar FOR-TO-NEXT por um outro valor que não seja 1, e para permitir a decrementação (alterar os números por ordem descendente)

Para maiores informações consulte STEP.

```
EXEMPLO 2
200 REM USANDO A INSTRUCAO FOR...NEXT
210 FOR X = 1 TO 100
220 PRINT X; " ";
230 NEXT
EXECUÇÃO DO PROGRAMA
1  2  3  4  5  6  7  8  9  10  11  12  13  14
15  16  17  18  19  20  21  22  23  24  25  26  27
 28  29  30  31  32  33  34  35  36  37  38  39
40  41  42  43  44  45  46  47  48  49  50  51  52
 53  54  55  56  57  58  59  60  61  62  63  64
65  66  67  68  69  70  71  72  73  74  75  76  77
 78  79  80  81  82  83  84  85  86  87  88  89
90  91  92  93  94  95  96  97  98  99  100
```

Se o seu computador não aceitar a instrução FOR, tecle e execute o seguinte programa

```
200 REM SUBSTITUINDO A INSTRUCAO FOR
310 X = 1
320 PRINT X; " ";
330 IF X = 100 THEN END
340 X = X + 1
350 GOTO 320
```

ORTOGRAFIA ALTERNATIVA

Alguns computadores que seguem a linha TRS-80, Nível I permitem utilizar F no lugar de FOR.

SE O SEU COMPUTADOR NÁO TIVER ESTA INSTRUÇÁO

Se o seu computador não aceitar a instrução FOR, poderá utilizar um laço contador para substituí-la. Substitua as linhas 110 e 130 no Exemplo 1 por:

```
110 X = 1
115 X = X+1
130 IF X = < 6 THEN 120
```

OBSERVAÇÁO:

Quando FOR for utilizado, convém lembrar que os microcomputadores que seguem a linha APPLE II operam aproximadamente duas e meia vezes mais rápido que os computadores que seguem a linha TRS-80. Loops deverão ser, portanto, reduzidos seguidos de um fator de 2.5.

VARIAÇÓES DE USO

Alguns computadores que aceitam a versão BASIC-PLUS-2, sob determinadas condições, permitem FOR-TO com o NEXT apenas implícito e não impresso de fato.

Exemplo:

```
100 PRINT X,SQR(X) FOR X = 1 TO 6
```

imprime uma tabela de valores dos números de 1 a 6, com as respectivas raízes quadradas.

VEJA TAMBÉM

NEXT, STEP

Função **FRAC • FRA**

SINTAXE GERAL

Número de linha LET variável = FRAC

FRAC é uma função usada no modo programa devendo vir precedida de uma instrução geralmente LET ou PRINT.

A função FRAC é utilizada por apenas algumas versões BASIC para isolar a parte fracionária de um número, em conjunto com o respectivo sinal.

120 F = FRAC(-17.583), por exemplo, dá a F o valor de -.583.

EXEMPLO 1

```
100 REM USANDO A FUNCAO FRAC
110 N = -15.75
120 F = FRAC(N)
130 PRINT "FRAC FOI APROVADO SE -.75 = ";F
140 END
```

EXECUÇÃO DO PROGRAMA

```
RUN
FRAC FOI APROVADO SE -.75 = -.75
```

SE O SEU COMPUTADOR NÃO TIVER ESTA FUNÇÃO

Se FRAC não foi aprovado, substitua a linha 120 por

```
120 F = N-SGN(N)*INT(ABS(N))
```

e rode (RUN) o exemplo novamente.

VEJA TAMBÉM

INT, FIX

Função FRE • $FREE • FRE(0)
FRE (STRING)

SINTAXE GERAL

Número de linha PRINT FREE (variável)

FRE é uma função usada geralmente no modo programa devendo ser precedida de uma instrução normalmente PRINT.

FRE(X) indica o espaço na memória disponível para o usuário.

FRE retorna o espaço livre disponível, o número de bytes de memória RAM. Não permitido no Integer BASIC.

A função FREE(string) é utilizada para substituir o número de bytes de espaço total para strings, atribuído mas não utilizado dentro da memória do computador. Qualquer carácter (contido entre aspas) ou variável de string pode ser utilizado com a função FRE.

O B$ na linha 140 abaixo é puramente arbitrário.

A maioria dos computadores com capacidade FRE reservam automaticamente 50 bytes de espaço para strings ao se ligar o computador.

```
EXEMPLO 1
100 REM USANDO A FUNCAO FRE
110 PRINT "DE ENTRADA EM QUALQUER COMBINACAO DE LETRAS E NUMEROS";
120 INPUT D$
130 PRINT "A QUANTIDADE DE ESPACO PARA STRING INUTILIZADO = ";
140 PRINT FRE(B$)
150 END

EXECUÇÃO DO PROGRAMA
RUN
DE ENTRADA EM QUALQUER COMBINACAO DE LETRAS E NUMEROS? 1 2 3 4 5 6 7 8 9
A QUANTIDADE DE ESPACO PARA STRING INUTILIZADO = 41
DE ENTRADA EM QUALQUER COMBINACAO DE LETRAS E NUMEROS? MICRO
A QUANTIDADE DE ESPACO PARA STRING UTILIZADO = 45
```

Determinados computadores utilizam números ou variáveis numéricos na função FRE para assinalar a quantidade total de memória que ainda está disponível (e não apenas a parte reservada para strings), de maneira semelhante à instrução MEM.

```
EXEMPLO 2
200 REM USANDO A FUNCAO FRE(MEM)
210 PRINT FRE(N); "BYTES DE MEMORIA AINDA DISPONIVEIS"
220 END

EXECUÇÃO DO PROGRAMA
RUN
48027 BYTES DE MEMORIA AINDA DISPONIVEIS
```

A quantidade de memória ainda disponível dependerá do tamanho da memória do computador.

Teclar NEW restabelece geralmente todas as variáveis String (e numéricas) de volta em nulo (e zero) de modo que a memória total fique novamente disponível.

ORTOGRAFIA ALTERNATIVA

Diversas versões BASIC utilizam FRE(0) para informar a quantidade total de memória ainda disponível.

Experimente FRE na linha 210 do Exemplo 2, para ver se seu computador o aceita.

VARIAÇÕES DE USO

FRE(exprnm) nos microcomputadores que seguem a linha APPLE II é disponível apenas na versão do BASIC APPLESOFT para retornar o número de bytes da memória presentemente disponível para o programa.

A memória disponível é aquela abaixo da área de armazenamento dos valores alfanuméricos (strings) e acima da área de armazenamento de variáveis subscritas (arranjos). Existindo mais de 32767 bytes de memória livres, a função FRE devolve um número negativo.

Some 65536 ao número para achar o tamanho real da memória disponível.

O FRE também limpa valores alfanuméricos não usados da área de armazenamento.

O valor de exprnm não é usado por FRE. Não é disponível para os micros que seguem a linha APPLE II na versão Integer BASIC.

VEJA TAMBÉM

MEM, CLEAR, $, NEW

SINTAXE GERAL

FP [,DN] [,SN] [,VN]

Este é um comando do DOS 3.3 utilizado nos microcomputadores que seguem a linha APPLE para colocar seu sistema na versão BASIC APPLESOFT. É usado apenas em modo imediato.

Se o seu computador tiver o cartão de linguagem, ou mais que 48kb, basta teclar INT que o carácter indicativo do Integer BASIC aparecerá na tela. Para retornar a versão BASIC APPLESOFT, tecle FP.

A fonte do APPLESOFT depende da linha de APPLE II (APPLE II, IIPLUS, IIE, IIC) utilizada, e dos opcionais que estão instalados.

a) com APPLE II PLUS, IIE, IIC, a linguagem está na ROM(memória só de leitura), não importa que tipos de opcionais existem.

b) se tiver o cartão APPLESOFT firmware instalado, o FP obtém a linguagem a partir dele, não importanto as chaves do cartão.

c) com o APPLE II Language System instalado, o FP pega o APPLESOFT dele.

d) em qualquer outro APPLE II, o FP procura o APPLESOFT no disco especificado (ou corrente). Se ele não existe lá, a mensagem LANGUAGE NOT AVAILABLE aparece.

FP representa um ponto flutuante. Se você usa ou está usando Integer BASIC e decide escrever um programa para balancear seu talão de cheque, você precisará ligar o APPLESOFT básico, o qual pode operar com números decimais.

A sintaxe usada é:

FP [,Dn] [,Sn] [,Vn]

onde:

[,Dn] onde n é 1 ou 2, especifica a drive do disco a ter acesso. Se omitido o DOS usa o último drive referenciado.

[,Sn] onde n é um número de 1 a 7, especifica o slot que contém o cartão de controle do drive. Se for omitido o DOS usa o último slot referenciado.

[,Vn] onde n é um número que vai de 0 a 254 e especifica o número do volume do disco a ter acesso.

O número de volume é um parâmetro associado a cada disquete de maneira análoga a que associamos um número a cada livro de uma enciclopédia ou a cada um dos volumes de uma coleção de livros. Este parâmetro pode ser usado para evitar que disquetes sejam regravados acidentalmente.

Se teclar INT enquanto estiver em Integer BASIC, perderá todos os programas na memória. O mesmo ocorrerá se estiver em APPLESOFT e teclar FP.

Quando você muda de Integer para APPLESOFT ou vice-versa, perde qualquer programa que porventura esteja na memória.

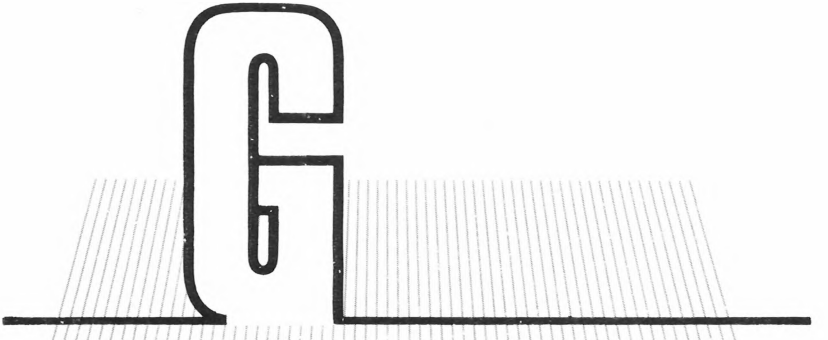
FP zera qualquer programa BASIC presente na memória.

OBSERVAÇÃO: Não use o comando RUN APPLESOFT para mudar

linguagens. Se fizer isso, o DOS testa o tipo de arquivo e coloca os pontos da memória para acomodar um programa em Integer BASIC, em vez de um programa APPLESOFT, em que pretendia trabalhar.

VARIAÇÕES DE USO
Não se tem conhecimento de nenhuma.

VEJA TAMBÉM
INT, CALL-151(no seu manual)



GE • GET
GO • GOSUB • GOS.
GOSUB-OF • GOTO • GO TO
GOT • G. • GOTO-OF
GR • GT • GRAD

Operador GE

GE é utilizado em determinados microcomputadores como abreviatura do sinal "maior do que ou igual a" (>=).
Para maiores informações veja >=

EXEMPLO 1

```
100 REM USANDO O OPERADOR GE(MAIOR DO QUE OU IGUAL A)
110 IF 50 >= 20 THEN 140
120 PRINT "O OPERADOR GE NÃO FOI APROVADO NA LINHA 110"
130 GOTO 180
140 IF 50 >= 50 THEN 170
150 PRINT "O OPERADOR GE NÃO FOI APROVADO NA LINHA 140"
160 GOTO 180
170 PRINT "O OPERADOR GE FOI APROVADO"
180 END
```

EXECUÇÃO DO PROGRAMA

```
RUN
O OPERADOR GE FOI APROVADO
```

VARIAÇÕES DE USO

Não se tem conhecimento de nenhuma.

VEJA TAMBÉM

>=, IF-THEN

Instrução GET

SINTAXE GERAL

Número de linha instrução GET, variável alfanumérica
ou
GET var

GET A$ é uma instrução utilizada geralmente no modo programa para produzir uma suspensão da execução do programa enquanto espera pela variável A$ ser introduzida (input). Não é necessário pressionar a tecla de entrada (RETURN, CR ou equivalente), para que o input seja aceito.
GET pode ser usado em modo imediato. Não é disponível em Integer BASIC.
Nos microcomputadores que seguem a linha TRS-80, a instrução similar é o INKEY$ porém INKEY$ não provoca a parada do programa. Você pode forçá-lo a esperar da seguinte maneira:
10 IF INKEY$ = " " THEN GOTO 10

GET é uma declaração do APPLESOFT utilizada por microcomputadores que seguem a linha APPLE II ou PET para aceitar um só carácter do teclado sem exibi-lo na tela e sem esperar para que se pressione a tecla RETURN. Utiliza-se de maneira semelhante a INKEY$.
Com uma variável numérica tal como GET A, GET aceita unicamente uma entrada numérica. Uma variável de string (tal como GET A$) aceitará entradas de qualquer tecla menos a tecla STOP.
A instrução GET nos microcomputadores que seguem a linha APPLE faz com que a execução do programa faça uma pausa até ser pressionada uma tecla.
Se a declaração GET especifica uma variável real ou inteira a entrada deve ser numérica. GET é geralmente usado para receber strings.
No computador PET, GET faz a "varredura" do teclado. Caso não encontre nenhuma tecla pressionada, armazena um carácter nulo e procede ao resto do programa. (Para maiores informações consulte INKEY$)

EXEMPLO 1

```
100 REM USANDO A INSTRUCAO GET
110 PRINT "TECLAR QUALQUER CARACTER"
120 GET D$
130 IF D$ = "" THEN 120
140 PRINT "VOCE ACABA DE PRESSIONAR A TECLA ";D;""
150 PRINT "TECLE ";D$;" DE NOVO PARA CONTINUAR"
160 GET B$
170 IF B$ = D$ THEN 120
180 GOTO 160
190 END
```

EXECUÇÃO DO PROGRAMA (utilizando-se X)

```
RUN
TECLAR QUALQUER CARACTER
VOCE ACABA DE PRESSIONAR A TECLA X
TECLE X DE NOVO PARA CONTINUAR
```

VARIAÇÕES DE USO
Muitos microcomputadores utilizam GET # para ler um registro em disco ou fita.

Exemplo: GET #2,REC%
READs informações armazenadas no registro REC%, a partir do arquivo #2.
Alguns BASIC (tais como o N-BASIC da NEC, o BASIC de cor Estendida do TRS-80, e o BASIC Nível III da Microsoft) proporcionam um GET que armazena informações exibidas em determinada seção da tela, devendo ser especificadas as localizações X,Y dos cantos opostos do retângulo de fronteiro, é mais o nome do arranjo em que o respectivo material vai ser armazenado.

Exemplo:
GET @ (10,8)-(25,14),A%
guarda no arranjo A% os caracteres e os símbolos gráficos das colunas de 10 a 25 nas linhas de 8 a 14. As informações podem ser devolvidas à tela mediante uma instrução PUT @ semelhante.
GET é utilizado pelos BASIC da Hewlett Packard para LOAD (chamar) um programa ou arquivo de dados a partir de disquete ou fita.
Por exemplo:
GET "PROG"
Apaga qualquer programa na memória e LOAD (chama) para PROG.
GET pode também ser utilizado para APPEND, um segmento de programa no final de um programa atualmente existente por especificar o primeiro número a ser atribuído ao programa que está sendo carregado.
GET "SUB1" ,500
irá renumerar o programa SUB1 à medida que for sendo carregado para a memória, a começar na linha 500. Todas as linhas do programa existente com números de linha inferiores a 500 serão retidas, e terão as linhas do programa SUB1 apenas no final.
A instrução GET A$ nos micros que seguem a linha APPLE II não requer os mesmos tipos de instruções dos que seguem a linha TRS-80.

VEJA TAMBÉM
INKEY$, KEY$, PUT, APPEND, @

Comando/Instrução **GO**

SINTAXE GERAL

Número de linha GO TO ou GOTO

GO pode ser utilizado tanto como comando no modo direto ou como instrução no modo programa; porém deve ser utilizado como parte da instrução GOTO ou GOSUB.
É reconhecido na maioria das versões do BASIC para provocar um desvio incondicional para a linha indicada.
A maioria dos computadores não se preocupa se existe um espaço depois do GO, e converte automaticamente para GOTO ou GOSUB. Outros (tais como o Nível I do TRS-80) não permitem o espaço.
GO tem geralmente significação tão-somente ao ser conjugado com outra palavra BASIC.

EXEMPLO 1

```
100 REM EXEMPLO GO NA INSTRUCAO GO TO
110 PRINT "A INSTRUCAO GO ";
120 GO TO 150
130 PRINT "NAO APROVOU"
140 GOTO 160
150 PRINT "APROVOU"
160 END
```

EXECUÇÃO DO PROGRAMA

```
RUN
A INSTRUCAO GO APROVOU
```

EXEMPLO 2

```
100 REM EXEMPLO DE GO NA INSTRUCAO GO SUB
110 GO SUB 1000
120 PRINT "APROVOU, AO SER UTILIZADO COM SUB"
130 GO TO 1050
1000 REM SUB-ROTINA
1010 PRINT "A INSTRUCAO GO ";
1020 RETURN
1050 END
```

EXECUÇÃO DO PROGRAMA

```
RUN
A INSTRUCAO GO APROVOU AO SER UTILIZADO COM SUB
```

VARIAÇÕES DE USO
GO é utilizado pela DATAPOINT como forma abreviada de GOTO, na qualidade de instrução e de comando direto. Como comando GO n, o computador transfere a execução do programa na linha n.

VEJA TAMBÉM
GOTO, GOSUB, IF-GOTO, ON-GOTO, GOTO-OF, ON-GOSUB, GOSUB-OF, CONT

Instrução **GOSUB • GOS.**

SINTAXE GERAL

GOSUB lin
ou
Número da linha GOSUB número de linha

GOSUB é utilizado como instrução no modo programa.
GOSUB XX é utilizado tanto nos equipamentos que seguem a linha APPLE quanto TRS-80 ou Sinclair; esta instrução permite executar uma sub-rotina que começa na linha XX. Convém observar o seguinte, tanto nos microcomputadores que seguem a linha APPLE ou TRS-80.
XX tem que ser uma constante numérica inteira e positiva, enquanto que nos microcomputadores que seguem a linha SINCLAIR poderá ser também uma variável numérica ou uma expressão desde que resulte num valor inteiro e positivo que represente um número de linha existente no programa .
Em Integer BASIC também aceita GOSUB exprnm.
GOSUB transfere a execução do programa principal para a execução de uma sub-rotina que tem início na linha indicada logo após a instrução GOSUB.

OBS.:As instruções GOSUB e RETURN funcionam em conjunto sendo que o GOSUB transfere a execução do programa principal para uma sub-rotina (programa secundário) e a instrução RETURN é utilizada no final da execução da sub-rotina devolvendo o controle da sub-rotina para o programa principal a partir da primeira instrução que vier após a instrução GOSUB.
As sub-rotinas podem aparecer em qualquer parte do programa, mas devem ser prontamente distinguíveis do programa principal. Deve-se, porém, tomar cuidado para que o processamento não entre inadvertidamente na sub-rotina; esta deve ser precedida de uma instrução END ou GOTO.

EXEMPLO 1

```
100 REM USANDO A INSTRUCAO GOSUB
110 FOR X = 1 TO 2
120 PRINT
130 NEXT X
140 PRINT "EXECUTO PROGRAMA PRINCIPAL"
150 PRINT "VOU A SUB-ROTINA"
160 GOSUB 1000
170 PRINT "AGORA VOLTEI AO PROGRAMA PRINCIPAL"
180 PRINT "F I M"
190 END

1000 FOR X = 1 TO 2
1010 PRINT
1020 NEXT X
1030 PRINT "EXECUTANDO A SUB-ROTINA"
```

```
1040 PRINT "RETORNO AO PROGRAMA PRINCIPAL"
1050 PRINT
1060 END

EXECUÇÃO DO PROGRAMA
RUN
EXECUTO PROGRAMA PRINCIPAL
VOU A SUBROTINA

EXECUTANDO A SUBROTINA
RETORNO AO PROGRAMA PRINCIPAL

AGORA VOLTEI AO PROGRAMA PRINCIPAL
F I M
```

Execute o programa acima para melhor entender o GOSUB e observe que a instrução END na linha 190 impede que o programa continue executando novamente as linhas linhas da sub-rotina.
OBS.: Apague a linha 190 e verifique o que acontece.

```
EXEMPLO 2
200 REM EXPONENCIACAO
210 PRINT "ENTRE COM VALOR DA BASE"
220 INPUT B
230 PRINT "ENTRE COM VALOR EXPOENTE"
240 INPUT N
250 GOSUB 2000
260 PRINT B " ELEVADO A "N; " E IGUAL A ";E
265 PRINT
270 GOTO 210
275 RETURN

2000 REM SUBROTINA CALCULAR BASE B ELEVADO EXPOENTE N
2010 LET E = 1
2020 FOR X = 1 TO N
2030 LET E = E * B
2040 IF N = 0 THEN LET E = 1
2050 NEXT X
2060 RETURN

EXECUÇÃO DO PROGRAMA
RUN
ENTRE COM VALOR DA BASE
? 5   (se for teclado 5)
ENTRE COM VALOR EXPOENTE
? 3   (se for teclado 3)
5 ELEVADO A 3 E IGUAL A 125

ENTRE COM VALOR DA BASE
?
```

OBS.: Este exemplo é válido para expoentes inteiros e positivos.

VARIAÇÕES DE USO
Alguns computadores que seguem a linha inglesa SINCLAIR tais com
TK-82, TK-85, NE-80, NE-8000, ZX-80, ZX-81, SINCLAIR-100
aceitam expressões variáveis na instrução GOSUB. Neste
computadores GOSUB X e até GOSUB X = 20 + 200 são instruçõ

aceitáveis. Entretanto devemos atribuir a X um valor apropriado durante a execução do programa. Sendo X um número inteiro de valor reduzido, GOSUB X*10 da um resultado semelhante em ON X GOSUB 100,200,300,...

SINTAXE ADICIONAL
Na versão Integer BASIC nos computadores que seguem a linha APPLE GOSUB exprnm.
Em Integer BASIC, uma expressão numérica pode ser colocada no lugar do número de linha. Se o cálculo de exprnm não resultar em um número de linha existente aparecerá a mensagem de erro:
BAD BRANCH ERR. Esta sintaxe permite simular a instrução ON GOSUB, que não é disponível em Integer BASIC.
EXEMPLO:
```
20 X = 10
30 GOSUB X
```

ORTOGRAFIA ALTERNATIVA
Agumas versões de BASIC utilizam GOS. como abreviatura de GOSUB.

VEJA TAMBÉM
ON-GOSUB, IF-GOSUB, RETURN

Instrução/Comando **GOSUB-OF**

SINTAXE GERAL

Número de linha, GOSUB variável, OF

GOSUB-OF pode ser utilizado como comando ou como instrucão no modo programa.
GOSUB-OF é um esquema de desvio para sub-rotinas múltiplas semelhantes a ON-GOSUB, utilizado por microcomputadores, da linha Hewlett Packard.
GOSUB Y OF 1000, 2000, por exemplo, faz com que a execução do programa principal se desvie para a sub-rotina em 1000 se Y tiver valor de 1 e para 2000 se o valor de Y for 2.

EXEMPLO 1

```
100 REM USANDO GOSUB-OF
110 PRINT "DE ENTRADA NOS NUMEROS 1,2 OU 3";
120 INPUT Y
130 PRINT "A INSTRUCAO GOSUB-OF ";
140 GOSUB Y OF 300,330,360
150 GOTO 110
300 REM SUB-ROTINA NO.1
310 PRINT "SE DESVIOU PARA A SUB-ROTINA NO.1"
320 RETURN
330 REM SUB-ROTINA NO.2
340 PRINT "SE DESVIOU PARA A SUB-ROTINA NO.2"
350 RETURN
360 REM SUB-ROTINA NO.3
370 PRINT "SE DESVIOU PARA A SUB-ROTINA NO.3"
380 RETURN
400 END
```

EXECUÇÃO DO PROGRAMA

```
RUN
DE ENTRADA NOS NUMEROS 1,2, OU 3? 1
A INSTRUCAO GOSUB-OF SE DESVIOU PARA A SUB-ROTINA NO.1
DE ENTRADA NOS NUMEROS 1,2, OU 3? 2
A INSTRUCAO GOSUB-OF SE DESVIOU PARA A SUB-ROTINA NO.2
DE ENTRADA NOS NUMEROS 1,2, OU 3? 3
A INSTRUCAO GOSUB-OF SE DESVIOU PARA A SUB-ROTINA NO.3
DE ENTRADA NOS NUMEROS 1,2, OU 3?
```

VARIAÇÕES DE USO

Não se tem conhecimento de nenhuma.

VEJA TAMBÉM

ON-GOSUB, GOTO-OF, ON-GOTO

Comando/Instrução **GOTO**
GO TO • GOT • G.

SINTAXE GERAL

GOTO número de linha
ou
Número da linha GOTO número de linha

GOTO pode ser utilizado como comando no modo direto ou como instrução no modo programa.
GOTO XX é comum em todos os microcomputadores na linguagem BASIC para executar um desvio incondicional para a linha de número XX. Entretanto, XX deve representar um número inteiro e positivo.
GOTO...significa "vá para..."
GOTO determina que o programa "salte" para uma linha indicada, diferente da próxima, e continue a execução do programa a partir dessa linha.
GOTO altera a execução normal do programa, que é de seguir da linha de menor número para a linha de número maior, desviando incondicionalmente para um número de linha especificado.
GOTO é usado como comando no modo direto (em substituição ao RUN) para reiniciar um programa já executado sem "zerar" as variáveis.
GOTO usado como instrução permite que se desvie, "salte" de uma linha para outra do programa.

EXEMPLO 1
```
100 REM USANDO GOTO COMO INSTRUCAO
110 PRINT "ESTE E ";
120 GOTO 150
130 PRINT "INSTRUCAO GOTO"
140 GOTO 170
150 PRINT "UM EXEMPLO DA ";
160 GOTO 130
170 END
```

EXECUÇÃO DO PROGRAMA
```
RUN
ESTE E UM EXEMPLO DA INSTRUCAO GOTO
```

EXEMPLO 2
```
200 GOTO 260
210 LET Y = X * X
230 PRINT "O QUADRADO DO NUMERO ";
240 PRINT "TECLADO E "Y
250 END
260 PRINT "ENTRE COM UM NUMERO? "
270 INPUT X
280 GOTO 210
```

EXECUÇÃO DO PROGRAMA

```
RUN
ENTRE COM UM NUMERO?
? 5 (se for teclado 5)
O QUADRADO DO NUMERO TECLADO E 25
```

OBS.: Em muitos casos GOTO é utilizado em conjunto com outras palavras-chave.

ORTOGRAFIAS ALTERNATIVAS
GO TO, GOT, G.
experimente uma destas abreviaturas no Exemplo 1, na linha 120 e verifique se o seu computador aceita.

Sintaxe Adicional no Integer BASIC

GOTO exprnm

Nós microcomputadores que seguem a linha APPLE II na versão Integer BASIC é permitido colocar uma expressão numérica no lugar do número de linha. Se o número de linha calculado não existir, aparecerá uma mensagem de erro.
Esta sintaxe do GOTO permite simular a instrução ON GOTO, que não é disponível na versão Integer BASIC.

VEJA TAMBÉM
GOTO-OF, IF-GOTO, ON-GOTO

Instrução **GOTO-OF**

SINTAXE GERAL

Número de linha, GOTO, variável, OF incremento.

GOTO-OF é uma instruçao utilizada pelos microcomputadores Hewlett Packard para desvios múltiplos que incorporam numa única instrução uma série de testes IF-THEN. É uma instrução utilizada no modo programa.
GOTO Y OF 100, 200, 300, por exemplo, instrui o microcomputador a se desviar para as linhas 100,200, e 300 caso o valor do número inteiro de Y seja 1,2 ou 3 respectivamente. Sendo o INT Y inferior a 1 ou superior a 3, os testes neste exemplo falham todos e a execução passa para a próxima linha do programa.
O valor INT de Y não pode ultrapassar o número de possíveis desvios da instrução.
A maioria dos microcomputadores aceita tanto GO TO (duas palavras) como GOTO (uma palavra), ao passo que poucos aceitam GOTO como duas palavras.

EXEMPLO 1

```
100 REM USANDO GOTO-OF
110 Y = 2
120 GOTO Y OF 130,150
130 PRINT "GOTO-OF NAO APROVOU"
140 GOTO 160
150 PRINT "GOTO-OF APROVOU"
160 END
```

EXECUÇÃO DO PROGRAMA

```
RUN
GOTO-OF APROVOU
```

VARIAÇÕES DE USO
Não se tem conhecimento de nenhuma.

VEJA TAMBÉM
ON-GOTO, ON-GOSUB, IF-THEN, INT, GOSUB-OF

Comando/Instrução **GR**

SINTAXE GERAL

Número da linha GR

GR pode ser utilizado no modo direto como comando ou como instrução no modo programa para a execução de gráficos em baixa resolução, em uma tela de 40x40.
GR é utilizado em microcomputadores que seguem a linha APPLE II, assim como o MICROENGENHO, AP II, MAXXI, DACTRON e outros.
Os microcomputadores que seguem a linha TRS-80 como o CP-500, DGT 100, DGT 1000, DISMAC 8000, e outros, não têm o comando (ou instrução) GR.
GR não vem seguido de parâmetros e pode ser utilizado tanto como comando ou instrução de programa para mudar a operação do computador do modo TEXT para o modo gráfico.
Este comando coloca a tela no modo gráfico de baixa resolução (40 por 40), deixando 4 linhas para texto na parte inferior. A tela é limpa e o cursor é movido para a janela de texto.
Pode-se converter para modo gráfico em tela completa (40 por 48), depois de executado GR com o comando POKE-16302, ou o equivalente comando POKE 49234,0.
Se GR segue um desses comandos POKE, voltamos a ter novamente o modo gráfico mais quatro linhas de texto. Depois de um comando GR, COLOR é colocado em zero. Estando no modo HGR2, GR limpa a tela toda da memória, mas deixa na tela a página.2 do modo gráfico de baixa resolução e texto. Para retornar para o modo normal, tecle TEXT.
Em programas, use TEXT antes de passar de HGR2 para GR.
GR deve ser executado antes de se utilizarem as instruções especiais para gráficos, como PLOT, HLIN-AT e VLIN-AT.
GR pode também ser utilizado para limpar a tela antes de se iniciar nova exibição de gráficos. Cada vez que se executa GR, o computador apaga toda a tela.

EXEMPLO 1

```
100 REM USANDO GR EM TELA DE BAIXA RESOLUCAO
110 GR
120 COLOR = RND (16) * 16
130 FOR Y = 0 TO 39
140 PLOT Y,Y
150 NEXT Y
160 GOTO 120
```

EXECUÇÃO DO PROGRAMA

```
RUN
```

Se o computador aceitou a instrução GR e se o seu vídeo for colorido, aparecerá uma linha em diagonal que mudará de cor aleatoriamente.

VARIAÇÕES DE USO

GR a porção gráfica da tela é colocada em preto, o cursor e

posicionado na janela de texto, e color é colocado em 0 (preto).
Se executado enquanto HGR está em atividade, o GR se comporta normalmente.
Entretanto, se HGR2 estiver em processo, o sistema dará acesso à página 2 de texto e gráficos em baixa resolução. Nesta situação nada que for digitado aparecerá na tela. Para voltar à situação normal digite TEXT.
Use sempre TEXT nos programas, ao passar HGR2 para GR.
Pode-se pular para a forma de tela cheia (40x48), em gráfico de baixa resolução, com o comando/instrução POKE -16302,0 após executar GR.
Tudo aquilo que for digitado após uso no modo direto será colocado na tela como pontos coloridos nas quatro últimas linhas da tela do vídeo, porém a execução será normal. Para que seja devolvida a janela de TEXTO, devemos digitar POKE-16302,0.

VEJA TAMBÉM
TEXT, COLOR, HLIN-AT, VLIN-AT, PLOT, CLS

Operador **GT**

GT é utilizado e reconhecido por apenas alguns microcomputadores (assim como os da linha TEXAS INSTRUMENTS) como palavra alternativa do sinal "maior do que"(>).
Para mais informações, veja >

EXEMPLO 1

```
100 REM USANDO OPERADOR GT (MAIOR DO QUE)
110 IF 30 GT 15 THEN 140
120 PRINT "O OPERADOR GT NAO APROVOU"
130 GOTO 150
140 PRINT "O OPERADOR GT APROVOU"
150 END
```

EXECUÇÃO DO PROGRAMA

```
RUN
O OPERADOR GT APROVOU
```

VARIAÇÕES DE USO
Não se tem conhecimento de nenhuma.

VEJA TAMBÉM
>, IF-THEN, >=, <, <=, =, <>, EQ, GE, LE, LT, NE

Instrução/Comando **GRAD**

SINTAXE GERAL

Número de linha GRAD

GRAD pode ser utilizado como comando no modo direto ou como instrução no modo programa.
GRAD é utilizado em apenas alguns microcomputadores de bolso Sharp/TRS-80 e a série 4050 da Tektronix para calcular em GRADs em vez de radianos. (100 grads = 90 graus).
A maioria dos computadores está no modo radiano ao serem ligados, mas alguns têm a capacidade de calcular funções trigonométricas em graus e alguns deles podem utilizar grads.

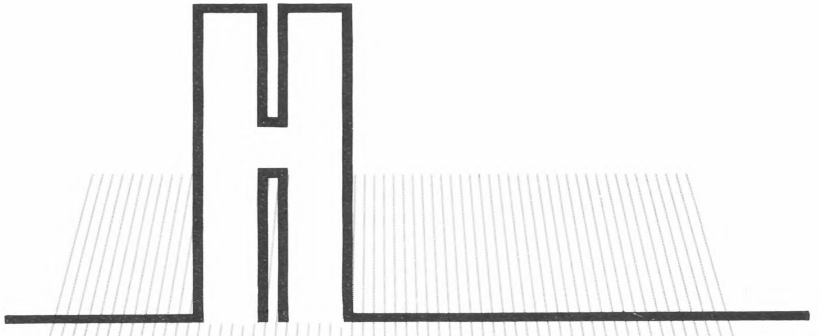
EXEMPLO 1

```
100 REM USANDO A INSTRUCAO GRAD
110 R = SIN(40)
120 PRINT "O SENO DE 40 RADIANOS E " ;R
130 GRAD
140 G = SIN(40)
150 PRINT "O SENO DE 40 GRADS E  ";G
160 END
```

EXECUÇÃO DO PROGRAMA

```
RUN
O SENO DE 40 RADIANOS E 745114
O SENO DE 40 GRADS E 587785
```

VEJA TAMBÉM
RAD, DEG, ACS, ASN, ATN, COS, SIN, TAN



HCOLOR = X • HGR
HGR2 • HLIN-AT • HPLOT
HTABX • HTAB (x)
HIMEM • HOME

Instrução **HCOLOR = X**

SINTAXE GERAL

Número da linha = número da cor

Esta instrução é reconhecida na versão BASIC APPLESOFT nos microcomputadores que seguem a linha APPLE II porém não está disponível na versão Integer BASIC.
HCOLOR coloca o gráfico de alta resolução na cor especificada pelo valor de HCOLOR, que deve estar entre 0 e 7, inclusive. A associação das cores e seus valores é a seguinte:

0 preto 1
1 verde (depende da tv)
2 violeta (depende da tv)
3 branco 1
4 preto 2
5 laranja (depende da tv)
6 azul (depende da tv)
7 branco 2

HCOLOR = exprnm até a instrução HCOLOR seguinte, todas as instruções HPLOT e DRAW serão executadas na cor especificada.
Os códigos de cores serão listados conforme valores acima.
Um ponto em alta resolução, plotado com HCOLOR = 2 (branco) será azul se a coordenada X do ponto for par; verde se X for ímpar, e branco, somente se ambos , (X,Y) e (X+1,Y), são plotados.
Este procedimento é normal nos aparelhos de TV.

EXEMPLO

```
100 REM USANDO A INSTRUCAO HCOLOR
120 HGR
130 HCOLOR = 3
140 X = PDL(0)
150 Y = PDL(1)
160 IF Y >159 THEN Y = 159
170 HPLOT X,Y
180 GOTO 140
```

HCOLOR não é mudado por HGR, HGR2 ou RUN. Até que o primeiro comando HCOLOR seja executado, a cor do gráfico de alta resolução é indeterminada.

```
HPLOT X,Y
HPLOT TO X,Y
HPLOT X,Y TO Z,W ! TO U,V !
```

O símbolo ! ! indica que o item é opcional ou pode ser repetido.
HPLOT X,Y plota um ponto cuja coordenada X é o valor da expressão X e a coordenada Y o valor da expressão Y. A cor do ponto é determinada pelo último comando HCOLOR executado.
HPLOT TO X,Y causa uma linha a ser plotada desde o último ponto plotado até (X,Y). A cor desta linha é determinada pela cor do último ponto plotado, mesmo que o valor de HCOLOR tenha sido

alterado a partir do "plotamento" anterior. Se nenhum ponto foi plotado, nenhuma linha é desenhada.
HPLOT X,Y TO Z,W TO U,V faz uma linha desde (X,Y) até (Z,W) usando a cor especificada pelo último comando HCOLOR. O traçado das linhas pode ser estendido com a inclusão do opcional TO U,V e limite da tela e os 239 caracteres por instrução devem ser respeitados.
O simples comando:
HPLOT 0,0 TO 279,0 TO 279,159 TO 0,159 TO 0,0 pode plotar uma margem retangular na tela de alta resolução.
Y e Z devem ter o valor de 0 até 279.
Y e W devem ter valor de 0 até 191.
Uma tentativa de plotar um ponto cujas coordenadas excedem estes limites causa mensagem ?VALOR ILEGAL:ERRO
Se a tela está 7 no modo gráfico de alta resolução mais 4 linhas para texto, então uma tentativa de plotar pontos com coordenadas Y, valendo de 160 até 191, não tem efeito visível.

VARIAÇÕES DE USO
Não se tem conhecimento de nenhuma.

VEJA TAMBÉM
COLOR, HGR, HGR2, GR

Instrução **HGR • HGR2**

SINTAXE GERAL

Número da linha HGR

HGR é uma instrução do BASIC APPLESOFT, para microcomputadores da linha APPLE, para colocar a tela do monitor na posição de alta resolução para gráficos.
Esta instrução não é disponível na versão Integer BASIC.
Nos microcomputadores que seguem a linha TRS-80, o procedimento mais aproximado seria disponível mediante o uso de POKE explicado nos manuais do nível II.
Não há parâmetros. Coloca a tela no modo gráfico de alta resolução (260 por 160 pontos), deixando quatro linhas de texto na parte inferior . A tela é limpa e a página 1 da memória (BK-16K) é mostrada. HCOLOR não é mudado por este comando. A memória do texto na tela não é afetada. O uso de HGR deixa uma "janela" de texto na tela completa, mas somente as quatro linhas inferiores do texto são visíveis abaixo do gráfico. O cursor não será visível se estiver acima das quatro linhas inferiores, isto é, dentro da "janela", e se tornará visível somente quando for removido para as quatro linhas inferiores.
A tela pode ser convertida totalmente para o modo gráfico (280 por 192 pontos) se depois da execução de HGR seguir um comando POKE; para tal uso tecle POKE-16302,0 ou POKE 49234,0 que é equivalente. Se HGR seguir um dos comandos POKE, teremos de volta o modo gráfico de alta resolução mais texto.

EXEMPLO - ANIMAÇÃO GRÁFICA

```
100 REM USANDO A INSTRUCAO HGR2
110 TEXT: HOME: NORMAL: SPEED = 255
120  DEF  FN(X)  =  SIN  (X*3.14/180):  REM FUNCAO SENO PARA GRAUS
DECIMAIS
130  DEF FN CS(X) = COS(X*3.14/180): REM FUNCAO CO-SENO PARA GRAUS
DECIMAIS
140  VTAB 10: INPUT "ANGULO :";AN$: AN = VAL (AN$): IF AN = 0 THEN
140
150  VTAB  14: INPUT "INCREMENTO:"; IN$: IN = VAL (IN$): IF IN = 0
THEN 150
160  HGR2: HCOLOR = 3: REM PREPARA A PAGINA DE ALTA RESOLUCAO PARA
GRAFICOS
170 XV = 140: YV = 96: A = 0: R = 0: REM DEFINE VALORES INICIAIS
180 A = A+AN: R = R+IN: REM INCREMENTA VALORES
190 X = XV+R*(FN CS(A)): REM DEFINE ABCISSA
200 Y = YV+R*(FN SN(A)): REM DEFINE ORDENADA
210 IF X <0 OR X >200 OR Y <0 OR Y> 192 THEN END
220 HPLOT XV, YV TO X,Y: REM TRACA LINHA
230 XV = X: YV = Y: GOTO 180
```

Em microcomputadores que seguem a linha APPLE II e com menos de 32K BYTES de memória, não se pode usar HGR e o DOS ao mesmo tempo desde que eles ocuparão necessariamente a mesma área de memória.

Também não se pode usar HGR com APPLESOFT em disquete ou cassete, visto que o interpretador APPLESOFT ocupa parte da página gráfica 1 de alta resolução.

VARIAÇÕES DE USO
HGR2 converte a tela inteira para modo gráfico de alta resolução (280 x 192), sendo mostrada a página 2 de gráfico em alta resolução.
HGR2 não afeta a memória de tela em baixa resolução (texto).
Entretanto não se consegue ver o que é digitado, apesar de ser executado normalmente.

OBSERVAÇÃO
Não tente gerar uma janela de texto com POKE - 16301,0; isto iria mostrar a página gráfica 2, enquanto que os comandos digitados vão para a página 1 tornando-os invisíveis.
Também não se pode usar HGR2 e o DOS de forma simultânea, a menos que o sistema tenha mais que 36K BYTES.
HGR2 não será disponível em sistema com menos de 24K de memória.
Em sistemas com 24K, coloque HIMEM: em 16384 antes de usar HGR2 para proteger o programa e as variáveis dos gráficos e vice-versa.

VEJA TAMBÉM
COLOR, HCOLOR, GR

Instrução **HLIN-AT**

SINTAXE GERAL

Número da linha HLIN col1, col2 AT lin

HLIN-AT pode ser usado como comando no modo direto ou como instrução no modo programa.
HLIN-AT é utilizada nas versões do BASIC de microcomputadores como característica especial para se exibir uma LINha Horizontal até determinada fileira da tela. HLIN quando executada, traça uma linha horizontal a partir da posição col1 para um ponto col2 na posição horizontal da linha especificada na tela.
EX:
HLIN 0,39 AT 0
desenha uma linha horizontal na parte mais alta da tela, da margem esquerda até a màrgem direita.
Utiliza-se também a sintaxe HLIN X, Y AT Z.
O comprimento da linha horizontal é determinado por dois números que seguem a instrução HLIN. Esses números indicam os limites entre os quais a linha se vai estender. A linha pode se estender em qualquer comprimento entre as colunas 0 e 39.
O número que segue AT representa o número da fileira que a linha deve ocupar, podendo esse número ir de 0 até 39.
HLIN 12,30, AT 33, por exemplo, instrui o computador a riscar uma linha horizontal entre a coluna 12 e a coluna 30 AT fileira 33.
A instrução GRÁficos deve ser executada antes que o computador possa aceitar a instrução HLIN-AT (ver GR). A cor da linha é determinada pela última instrução COLOR executada (veja color)

EXEMPLO 1

```
100 REM USANDO A INSTRUCAO HLIN-AT
110 GR
120 Y = 0
130 FOR X = 0 TO 39
140 COLOR = Y
150 HLIN 0,39 AT X
160 Y = Y + 1
170 IF Y < 16 THEN 190
180 Y = 0
190 NEXT X
200 END
```

EXECUÇÃO DO PROGRAMA

Se o computador aceitou a instrução HLIN-AT, a tela deve estar com 39 linhas horizontais em diversas cores.
A BASIC APF não utiliza AT na sua instrução HLIN. Em vez disto, é utilizada uma vírgula. O formato e a cor a serem utilizados se declaram nas instruções SHAPE e COLOR antes de se utilizar HLIN.

OBSERVAÇÕES: HLIN X,Y AT Z
Usado no modo gráfico de baixa resolução, HLIN desenha uma linha desde a coordenada (X,Z) até a coordenada (Y,Z).
A cor é determinada pelo último comando COLOR executado.

O valor da expressão X e da expressão Y deve estar nos limites de 0 a 39, e a expressão Z deve estar nos limites de 0 a 47 ou a mensagem ?VALOR ILEGAL:ERRO aparecerá.
O valor da expressão X pode ser maior, igual ou menor do que o valor da expressão Y.
Se HLIN é usado quando o sistema está no modo TEXT, ou no modo GR com expressão Z valendo de 40 a 47, então uma linha de caracteres será colocada onde a linha de pontos seria plotada.
Este comando não tem efeito visível quando usado no modo gráfico de alta resolução.

VARIAÇÕES DE USO
Na versão Integer BASIC col1 deve ser menor ou igual a col2 ou a mensagem RANGER ERR aparecerá na tela.
HLIN não é aceito nos microcomputadores que seguem a linha TRS-80 podendo entretanto ser traduzido para esses computadores assim:
o que trocará uma linha do ponto X para o ponto Y na posição vertical Y (Z)

VEJA TAMBÉM
GR, COLOR, PLOT, VLIN-AT, TEXT

Comando **HPLOT**

SINTAXE GERAL

```
HPLOT colh,linh
HPLO TO colh,linh
HPLOT colh1,linh1 TO colh2,linh2 {TO colh3,linh3...}
```

HPLOT X,Y

plotará um ponto na tela gráfica de alta resolução. A cor do ponto é determinada pelo último comando HCOLOR executado.

Esta é uma declaração do APPLESOFT, trata-se de um comando para gráficos de alta resolução equivalente a SET(X,Y). Não existe o mesmo comando no TRS-80 sem modificações. O comando no TRS-80 chamado PRINT @ funciona como substituto adequado do comando HPLOT.

A primeira forma do comando coloca um ponto colorido na tela e na posição especificadà. A cor do ponto é determinada pela última declaração COLOR executada.

A segunda forma do comando desenha uma linha colorida a partir do último ponto colocado até as coordenadas colh e linh. Se não houve ainda nenhum desenho desde que foi dado o comando HGR ou HGR2, nada será desenhado. A cor da linha é dada pela última declaração HCOLOR executada.

A terceira forma do comando traça uma linha colorida usando a cor especificada pelo último comando HCOLOR. O traçado das linhas pode ser estendido com o opcional TO colh3,linh3...

Pode haver qualquer número de pares de coordenadas desde que elas caibam em uma linha de programa.

Uma tentativa de plotar um ponto cujas coordenadas excedam estes limites causará uma mensagem de erro.

Deve-se executar uma declaração HGR ou HGR2 antes de HPLOT. De outra forma, pode-se destruir programas ou variáveis.

Não disponível no Integer BASIC.

HPLOT X,Y

Coloca ponto colorido na coordenada horizontal X e vertical Y. X varia de 0 a 279; Y de 0 a 159 (HGR) ou 191 (HGR2). A coordenada 0,0 é o canto superior esquerdo.

HPLOT X1,Y1 TO X2,Y2

Desenha uma linha do ponto X1,Y1 ao ponto X2,Y2. O comando pode ser estendido a pontos adicionais..

TO XN,YN

EXEMPLO 1

```
100 REM USANDO O COMANDO HPLOT
110 HGR
120 HCOLOR= 3
130 GOSUB 500
140 HPLOT A,B
150 GOSUB 500
160 HPLOT TO A,B
170 GOTO 130
500 A = PDL(0)/.912
510 B = PDL(1)/1.6
520 RETURN
```

EXECUÇÃO DO PROGRAMA
RUN
Este programa desenha linhas em sua tela.

VARIAÇÕES DE USO
Não se tem conhecimento de nenhuma.

VEJA TAMBÉM
PLOT, HLIN, VLIN, SET, RESET

Instrução **HTABX • HTAB (x)**

SINTAXE GERAL

Número da linha HTAB col
ou
Número da linha HTAB(exprnm)

Esta instrução é reconhecida na versão BASIC APPLESOFT dos microcomputadores que seguem a linha APPLE II.
HTAB(X) movimentará o cursor e a próxima posição "PRINT" horizontalmente para a posição X da linha (PRINT) que estiver sendo exposta.
O cursor se move para a direita ou esquerda para a coluna especificada pelo valor de col; sem limpar qualquer carácter na tela.

EXEMPLO 1
```
100 REM USANDO A INSTRUCAO HTAB
110 HOME
120 E$ = CHR$(32): INVERSE
130 FOR X = 1 TO 40: VTAB 8: HTAB X: PRINT E$: NEXT X
140  FOR  X  = 8 TO 18: VTAB X: HTAB 1: PRINT E$: VTAB X: HTAB 40:
PRINT E$;: NEXT X: NORMAL
150 PRINT
160 PRINT "APERTE QUALQUER TECLA P/CONTINUAR"
170 GET X$
180 PRINT "OBRIGADO..."
```

HTAB assume que a linha na qual o cursor está localizado tem 255 posições, de 1 até 255, sem levar em conta a largura do texto a ser colocado. Posições 1 até 40 são para a linha atual; posições 41 até 80 são para a próxima linha abaixo, e assim por diante. O comando HTAB move o cursor para a posição X da margem esquerda em diante, da linha atual. HTABs movimentam-se relativamente à margem esquerda do texto da tela, mas independentemente da largura da linha.
HTAB 0 move o cursor para a posição 256. Se X é negativo ou maior que 255, a mensagem ?VALOR ILEGAL:ERRO é mostrada.

ORTOGRAFIA ALTERNATIVA
Na versão Integer BASIC, usa-se a instrução TAB (X).
Em outros micros usa-se TAB (X) ou PRINT @.

VARIAÇÕES DE USO
Não se tem conhecimento de nenhuma.

VEJA TAMBÉM
PRINT @, VTAB, HTAB

Comando **HIMEM**

SINTAXE GERAL

HIMEM exprnm

Este comando é aceito nas versões BASIC dos microcomputadores que seguem a linha APPLE.

HIMEM coloca à disposição o mais alto número de endereço da memória para uso do programa em BASIC, incluindo área de variáveis.

HIMEM determina a posição mais alta de memória de leitura e gravação (RAM) disponível para o programa BASIC. O sistema operacional em disco (DOS) sempre reside acima do HIMEM se está presente. Com a declaração HIMEM pode-se reservar espaço para sub-rotinas em linguagem de máquina e tabelas de forma gráfica de alta resolução. Também se pode proteger a área de RAM reservada para tela gráfica de alta resolução.

HIMEM primeiramente é preenchido com a posição de memória mais alta dos microcomputadores que seguem a linha APPLE II (por exemplo, 49151 em sistemas de 48K). O DOS reside na parte mais alta da memória, de forma que ele ajusta o HIMEM aproximadamente 10.800 bytes abaixo quando é carregado. Cada arquivo de buffer adicional que é carregado via MAXFILES abaixa o HIMEM outros 595 bytes. Se o programa em APPLESOFT usa cordões, valores são guardados iniciando na posição resultante de HIMEM; trabalhando de cima para baixo. Consulte o mapa de memória no manual do seu microcomputador.

O valor de exprnm deve estar na faixa entre -65535 até 65535 (-32767 até 32767 em Integer BASIC) ou ocorre uma mensagem de erro.

Não se deve colocar HIMEM indicando posição acima da maior posição de memória disponível, porque nesse caso se corre o risco de ter armazenamento de variáveis em posição de memória não existente.

Pode-se ver o valor corrente de HIMEM usando as instruções apropriadas, como listado abaixo:

```
PRINT PEEK(116)*256+PEEK(115) para APPLESOFT
PRINT PEEK(77)*256 X PEEK(76) para Integer BASIC
```

HIMEM não é afetado pelos comandos de manutenção NEW, RUN ou CLEAR.

Se for dado HIMEM menor que LOMEM ou não se deixar espaço suficiente para o programa ocorrerá uma mensagem de erro.

Só pode ser usado em modo imediato no Integer BASIC.

VARIAÇÕES DE USO

Não se tem conhecimento de nenhuma.

VEJA TAMBÉM

LOMEM

Comando/Instrução **HOME**

SINTAXE GERAL

HOME
ou
número de linha HOME

HOME pode ser utilizado como comando no modo imediato ou como instrução no modo programa.
HOME, declaração do APPLESOFT limpa toda impressão na tela e move o cursor para a posição mais à esquerda da tela.
HOME limpa a tela sem apagar o programa da memória.
HOME comando ou instrução de computadores que seguem a linha APPLE II (americano).
HOME usado como instrução de programa para limpar a tela antes de uma apresentação gráfica.

EXEMPLO 1 (Usando HOME como comando direto)

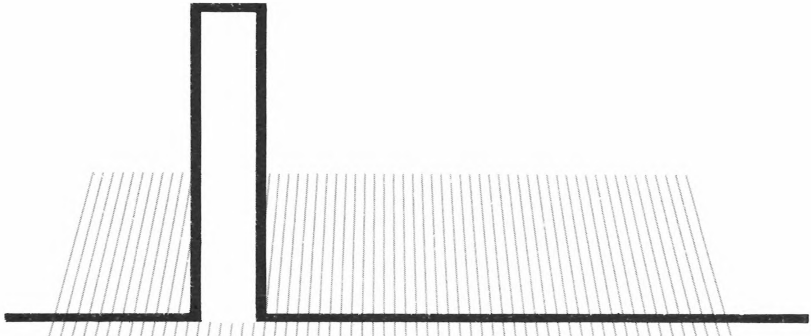
tecle HOME em seguida aperte a tecla
(RETURN).
OBS.: Toda impressão que estiver na tela será apagada e o cursor se posicionará no canto superior esquerdo da tela.

EXEMPLO 2

```
200 REM USANDO A INSTRUCAO HOME
210 PRINT "SE A TELA FOR LIMPA"
220 PRINT "E NAO APARECER MENSAGEM ERRO "
230 PRINT "O SEU COMPUTADOR ACEITA "
240 PRINT "A INTRUCAO <HOME>"
250 FOR Y = 1 TO 1000 STEP 0.2
260 NEXT Y
270 HOME
```

VARIAÇÕES DE USO
Em outros microcomputadores assim como aqueles que seguem a linha TRS-80, HOME deve ser substituído por CLS para limpar a tela. O cursor em ambos os casos vai para o canto superior da tela.
Nos microcomputadores que seguem a linha APPLE II na versão Integer BASIC deve ser utilizado para limpar a tela CALL-936

VEJA TAMBÉM
CLS, CALL-936



IF • IF . . . GOTO . . .
IF . . . LET . . . • IF . . . THEN . . .
IMAGE • INDEX • IN # • INIT
INKEY$ • INP • INPUT • IN. • I.
INPUTLINE • IMAGE • INDEX • IN
INPUT $ • INPUT 1 • INSTR
INT • I • INVERSE

Introdução Modificadora **IF**

SINTAXE GERAL

Número da linha IF nome da VARIÁVEL, condição.

IF é uma instrução modificadora que toma uma decisão após um teste para verificar qual caminho deve seguir a execução do programa.

IF é utilizado como instrução e faz parte das instruções de desvio IF-THEN...,IF-LET...,IF-GOTO...,IF-GOSUB etc.

IF indica a variável a ser testada por um dos operadores condicionais de igualdade ou desigualdade: =, <, >, <=, >=, <>.

IF encaminha para a instrução indicada após a instrução THEN se a condição entre o IF e o THEN for verdadeira; sendo falsa, o programa continuará normalmente na linha seguinte.

NOTA: Algumas versões do BASIC assim como o BASIC-Plus 2 utilizam o IF como modificador da maioria das outras instruções.

Por exemplo: 10 A = B IF A < B

A instrução de atribuição LET A = B será executada tão-somente se o valor corrente de A for inferior ao valor de B.

Alguns computadores permitem omitir a instrução THEN se a variável após o IF for seguida diretamente por um operador.

EXEMPLO 1

```
100  REM  USANDO  IF COM THEN IMPLICITO NAS LINHAS 120, 150 E
999
110 LET A = 3
120 IF A = 3 GOTO 150
130 PRINT "A LINHA 120 NAO FUNCIONOU"
140 GOTO 999
150 IF A = 3 GOSUB 900
160 GOTO 999

900 REM SUBROTINA
910  PRINT  "AS  LINHAS  120 E 150 FUNCIONARAM, SERA QUE ISTO
OCORRE TAMBEM NA LINHA 999"
920 RETURN
999 IF A = 3 END
```

EXECUÇÃO DO PROGRAMA

```
RUN
AS  LINHAS 120 E 150 FUNCIONARAM, SERA QUE ISTO OCORRE TAMBEM
NA LINHA 999
```

OBS.: Para se verificar adicionalmente a capacidade IF do computador VEJA TAMBÉM os exemplos de:

IF-THEN, IF-GOTO, IF-LET, ELSE

Instrução **IF... GOTO...**

SINTAXE GERAL

IF condição GOTO número de linha
Número da linha IF condição GOTO número de linha
IF...GOTO e uma instrução reconhecida por alguns microcomputadores para atender esta sintaxe.
IF...GOTO... e utilizado como instrução de desvio condicional no modo programa. Se a condição entre o IF e o GOTO for verdadeira, o computador executará a instrução GOTO.
São utilizados para teste da condição de igualdade ou desigualdade os seguintes operadores relacionais:

```
 =   IGUAL A
 <   MENOR DO QUE
 >   MAIOR DO QUE
<=   MENOR OU IGUAL
=<   IGUAL OU MENOR
>=   MAIOR OU IGUAL
=>   IGUAL OU MAIOR
<>   DIFERENTE DE
><   DIFERENTE DE
```

EXEMPLO 1

```
100 REM USANDO A INSTRUCAO IF...GOTO...
110 INPUT B
120 IF B > 5 GOTO 150
130 PRINT B " NAO E MAIOR QUE 5"
140 GOTO 170
150 PRINT B " E MAIOR QUE 5"
160 PRINT "A INSTRUCAO IF...GOTO... FUNCIONOU"
170 END
```

EXECUÇÃO DO PROGRAMA

```
RUN
? 6   (se teclarmos 6 para o INPUT B)
6 E MAIOR QUE 5
A INSTRUCAO IF...GOTO... FUNCIONOU
```

NOTA: Deve-se ter muita cautela com os computadores que permitem o uso de instruções múltiplas. Pois o IF...GOTO... devem ser colocados em último lugar numa linha que tenha instruções múltiplas, pois, sendo verdadeiro o teste, o computador executará o desvio incondicional do GOTO, e não sendo verdadeiro o computador executará a próxima linha numerada.
OBS.: Experimente... 120 IF B > 5 GOTO 150 : PRINT "NAO EXECUTO"
A instrução PRINT após os dois pontos não poderá ser executada.

ORTOGRAFIAS ALTERNATIVAS
Alguns computadores permitem formas abreviadas de IF...GOTO... como IF...GOT... ou IF...G.

VARIAÇÕES DE USO

Alguns interpretadores permitem a utilização da instrução THEN no lugar do GOTO.

EXPERIMENTE: 120 IF B > 5 THEN 150

VEJA TAMBÉM

IF-THEN..., GOTO, GOSUB, IF, IF-LET...

Instrução **IF... LET...**

SINTAXE GERAL

Número da linha IF condição LET variável numérica ou alfanumérica

Com esta sintaxe IF...LET...é reconhecido em apenas algumas versões BASIC de microcomputadores.

IF...LET... é utilizado como instrução condicional no modo programa. Se a condição entre o IF e o LET for verdadeira o computador executará a instrução LET.

São utilizados para teste de condição de igualdade ou desigualdade os seguintes operadores relacionais:

```
=    IGUAL A
<    MENOR DO QUE
>    MAIOR DO QUE
<=   MENOR OU IGUAL
=<   IGUAL OU MENOR
>=   MAIOR OU IGUAL
<>   DIFERENTE DE
><   DIFERENTE DE
```

EXEMPLO 1

```
100 REM USANDO A INSTRUCAO IF...LET...
110 LET A = 60
120 IF A > 50 LET N = 15
130 PRINT "N = ";N
140 PRINT "IF...LET...FUNCIONA SE O VALOR DE N FOR 15"
150 END
```

EXECUÇÃO DO PROGRAMA

```
RUN
N = 15
IF...LET...FUNCIONA SE O VALOR DE N FOR 15
```

NOTA: Experimente o mesmo exemplo modificando as linhas 110 e 120

```
110 A = 60
120 IF A > 50 N = 15
```

EXPERIMENTE TAMBÉM:

```
120 IF A > 50 THEN N = 15
              e
120 IF A > 50 THEN LET N = 15
```

VARIAÇÕES DE USO

Os computadores são uniformes no seu uso da instrução IF...THEN... A maioria permite omitir o LET ao passo que outros permitem omitir o LET porém exigem o uso do THEN.

VEJA TAMBÉM

IF, IF-THEN..., IF-GOTO..., LET

Instrução/Comando **IF... THEN...**

SINTAXE GERAL

IF condição THEN instrução
ou
número da linha IF condição THEN instrução

IF...THEN... estas duas instruções permitem testar a veracidade, ou não, de uma expressão ou variável; se for constatado que a condição é verdadeira, será executada a instrução que se segue ao THEN. Caso contrário o microcomputador executará a instrução da próxima linha.

Este par de instruções funciona de maneira semelhante na maioria das versões BASIC e sistemas com um carácter condicional:

Se tal condição acontecer...

IF... THEN... podem ser utilizados como comandos no modo direto, porém são mais utilizados como instruções no modo programa.

IF... THEN... formam um par de comandos ou de instruções que é empregado sempre em conjunto e funciona da seguinte maneira:

IF(Se) tal condição acontecer THEN(Então) execute esta instrução.

IF... THEN... permitem que o computador possa tomar uma decisão referente ao fluxo do programa baseado no resultado desenvolvido por uma condição.

A condição é algo que vai ser trabalhado para se saber se é verdadeira ou falsa; se aparecer como verdadeira, a instrução depois do THEN será executada, de outro modo, passar-se-á por cima dela.

As condições mais úteis comparam dois números ou duas "strings" (série de caracteres alfanuméricos): podem verificar se dois números são iguais ou se um é maior que o outro; e podem também verificar se duas "strings" são iguais ou se uma segue a outra por ordem alfabética.

A condição pode ser uma relação de igualdade ou de desigualdade em que são utilizados os símbolos abaixo:

=	IGUAL
<	MENOR DO QUE
>	MAIOR DO QUE
=<	IGUAL OU MENOR DO QUE
=>	IGUAL OU MAIOR DO QUE
<>	DIFERENTE DE

EXEMPLO 1

```
100 REM USANDO IF...THEN COMO COMANDO DIRETO
110 PRINT "IF...THEN... MODO DIRETO"
120 END
```

Se um comando direto IF...THEN... seguido de uma instrução GOTO número de linha for dado e já existir um programa, na memória RAM, que tenha a linha especificada na instrução após THEN, o computador executará, aquela instrução se a condição for

verdadeira (mesmo que ainda não tenha sido executado o programa).

```
IF A=0 THEN GOTO 110
(experimente no seu computador)
```

OBS.: Se não existir na memória do computador o programa com a linha 110, ocorrerá um erro "UL ERROR" Linha Indefinida.

```
EXEMPLO 2
200 REM USANDO IF...THEN COMO INSTRUCAO
210 PRINT "NUMERO", "MAIOR ATÉ AGORA"
220 INPUT A
230 LET M = A
240 PRINT A, M
250 INPUT A
260 IF M < A THEN LET M = A
270 GOTO 240

EXECUÇÃO DO PROGRAMA
RUN
NÚMERO        MAIOR ATÉ AGORA
? 5 (se entrar com valor cinco)
 5             5
? 3 (se entrar com valor três)
 3             5
? 10 (se entrar com valor dez)
 10            10
? 8 (se entrar com valor oito)
 8             10
```

OBS.: A parte mais importante é a linha 260, que atualiza M, se o seu valor anterior for menor que o novo número introduzido A.

A maioria dos computadores são uniformes na utilização da instrução THEN, muitos porém permitem que se omita a instrução THEN se a instrução IF for seguida diretamente por um operador matemático.

Deve-se ter muita cautela com os interpretadores que permitem o uso de linhas de instrucão múltiplas. IF...THEN... devem ser colocados em último lugar numa linha que tenha instruções múltiplas, pois, sendo verdadeiro o teste, o computador se desvia para a instrução indicada, se falso recai para a próxima linha numerada.

```
EXEMPLO 3
300 REM INSTRUCOES MULTIPLAS
305 PRINT "ENTRE COM UM NUMERO DE 1 A 10"
310 INPUT X
320 IF X = 10 THEN GOTO 340: PRINT "X=10"
330 GOTO 310
340 END
```

NOTA: Se for executado o exemplo acima, a instrução PRINT "X=10" não poderá ser executada jamais, pois se X = 10 o programa passa para a linha 340, senão, vai para a próxima linha do programa.

NOTA: Quando estiver usando IF para testar a igualdade para um valor que seja o resultado de um cálculo com PONTO FLUTUANTE, lembre-se de que a representação interna do valor pode não ser exata. Portanto, o teste deve ser feito em relação à faixa na qual a precisão do valor pode variar.

ORTOGRAFIAS ALTERNATIVAS

Alguns computadores aceitam IF-THE ou IF-T... ou simplesmente IF... para serem utilizados como formas abreviadas de IF-THEN...

IF-THEN-ELSE... A instrução que o microcomputador deve executar caso a expressão seja falsa.

Portanto, se na instrução IF-THEN... não usar ELSE e a expressão for falsa, o microcomputador executará a próxima linha do programa.

EXEMPLO 4

```
400 REM USANDO IF...THEN...ELSE...
410 INPUT "APERTE A TECLA DE NUMERO 5 "; N
420 IF N=5 THEN PRINT "MUITO OBRIGADO" ELSE PRINT "VOCE NAO APERTOU O NUMERO 5"
430 GOTO 410
```

EXECUÇÃO DO PROGRAMA

```
RUN
APERTE A TECLA DE NUMERO 5? 3
VOCE NAO APERTOU O NUMERO 5
APERTE A TECLA DE NUMERO 5? 5
MUITO OBRIGADO
APERTE A TECLA DE NUMERO 5?
```

OBS: A versão BASIC APPLESOFT não aceita IF...THEN...ELSE

IF...THEN... A versão APPLESOFT, tem problemas se o último carácter não-espaço que precede o THEN é a letra A.

O A combinado com o T forma a palavra reservada AT. Para evitar esse problema pode-se fechar com parênteses toda a expressão, incluindo o A.

VEJA TAMBÉM

IF, IF-GOTO, IF-LET, ELSE, AND, OR, NOT, GOTO, GOSUB, STOP, END.

Instrução **IMAGE**

A instrução IMAGE é utilizada por apenas algumas versões BASIC para especificar o formato de impressão que deve acompanhar uma instrução PRINT USING.

A instrução IMAGE é uma declaração de imagem ou de formato de conversão de dados.

EX:

```
60 IMAGE 3D,3X,2D.2D,3X,7A
```

São os seguintes os caracteres de formatação que podem ser utilizados numa instrução IMAGE:

X define os espaços em branco.
D define os caracteres numéricos.
A define os caracteres alfanuméricos.
E define o formato exponencial.
S define se o sinal é + ou -.
, / (vírgula e barra) são utilizadas como separadoras de formato.
O / também gera uma nova linha.

As constantes de string entre aspas podem ser utilizadas em qualquer lugar na linha de IMAGE.

Para saída na impressora é necessário notar que o número máximo de caracteres por linha varia, dependendo do tipo de impressora.

VARIAÇÕES DE USO

Não se tem conhecimento de nenhuma.

VEJA TAMBÉM

PRINT USING

Função **INDEX**

SINTAXE GERAL

Número de linha LET variável = INDEX variáveis.

INDEX é uma função usada geralmente no modo programa devendo ser precedida de uma instrução como LET ou PRINT.

INDEX nos indica a posição inicial do primeiro carácter num string que faz parte de um string maior. A posição é contada a partir do lado esquerdo.

INDEX se assemelha a INSTR, conforme utilizado em outros microcomputadores.

EXEMPLO 1

```
100 REM USANDO A FUNCAO INDEX
110 X$ = "MICRO"
120 Y$ = "APPLE"
130 K = INDEX(X$,Y$)
140 IF K <> 5 THEN 180
150 Y$ = "COMPUTADOR"
160 K = INDEX(X$,Y$)
170 IF K <>0 THEN 200
180 PRINT "INDEX FOI APROVADO"
190 GOTO 3999
200 PRINT "INDEX NAO FOI APROVADO"
3999 END
```

EXECUÇÃO DO PROGRAMA

```
RUN
INDEX FOI APROVADO
```

SE O SEU COMPUTADOR NÃO TIVER ESTA FUNÇÃO

Caso o exemplo tenha falhado experimente INSTR ou POS no lugar de INDEX. Se nenhuma das duas funcionar no seu computador, utilize a seguinte sub-rotina.

```
3000 GOTO 3999
3060 REM SUBROTINA ENTRADA N, X$, Y$, SAIDA K
3062 REM
3064 L = LEN(Y$)
3066 FOR K = N TO LEN(X$)-L+1
3068 IF Y$ = MID$ (X$,K,L) THEN 3074
3070 NEXT K
3072 K = 0
3074 RETURN
```

Para utilizar a sub-rotina com este exemplo, efetue as seguintes alterações:

```
125 N = 1
130 GOSUB 3060
160 GOSUB 3060
```

VARIAÇÕES DE USO

Não se tem conhecimento de nenhuma.

VEJA TAMBÉM

INSTR, POS

Comando/DOS **IN #**

SINTAXE GERAL

IN#s

É um comando utilizado em microcomputadores que seguem a linha APPLE II para seleção de dispositivos e E/S (entrada ou saída) e outros complementos.

IN#s este comando refere-se a uma porta periférica dentro do APPLE II designada por S. Seleciona a entrada do conector S. É usado para especificar qual periférico será abastecido por um comando INPUT. Os periféricos podem ser de 1 até 7, indicados por S.

O slot 0 (zero) não é um dispositivo periférico.

IN# 0 especifica o teclado como dispositivo de entrada. Se não existir periférico acoplado ao slot especificado, o sistema ficará bloqueado até o acionamento de RESET.

EX. IN#6

Obtém uma entrada subseqüente do dispositivo conectado no slot 6. Normalmente é recomendado colocar a interface para as duas primeiras drives no slot 6.

Este comando, quando executado, fará com que seja acionada a drive e tendo nela um disquete este será lido sendo introduzido o DOS na parte superior da RAM para que todos os comandos do DOS do sistema operacional possam ser utilizados.

Se não houver nenhum disquete na drive o sistema ficará com a luz vermelha da drive em uso até que seja usada a tecla RESET.

Sempre que o DOS estiver presente na memória do APPLE II, IN# será considerado um comando do DOS, requerendo PRINT e CHR$(4) em modo programado.

VARIAÇÕES DE USO

Em microcomputadores que seguem a linha TRS-80 não existe equivalente sem uma interface para expansão e um equipamento associado a ela. A depender do comando que esteja sendo transferido do computador pelo comando, será possível simular a mesma função usando o comando INPUT# X com um gravador cassete acoplado. Os manuais do TRS-80 originais costumam trazer instruções sobre o assunto.

VEJA TAMBÉM

PR#s

Comando/DOS **INIT**

SINTAXE GERAL

INIT nomearq (,Dn) (,Sn) (,Vn)

INIT nome do arquivo é um comando do DOS 3.3 (sistema operacional de disquete) utilizado para a inicialização de um disquete virgem, nos microcomputadores que seguem a linha APPLE II.

Inicialização consiste em formatar um disquete virgem e colocar no disquete uma cópia do DOS.

INIT nomearq geralmente exige um programa em BASIC qualquer, o qual será executado automaticamente sempre que este for carregado no DOS 3.3.

PROCEDIMENTOS PARA INIÇIALIZAÇÃO

1) Carregar o DOS 3.3 com o disquete mestre;

2) Teclar NEW e RETURN (CR);

3) Retirar o disquete mestre e colocar o disquete virgem a ser inicializado;

4) Criar um programa de inicialização com informações importantes, as quais ajudarão a identificá-lo com mais facilidade;

EXEMPLO

```
100 REM INICIALIZANDO UM DISQUETE
110 HOME
120 VTAB 8:HTAB 1
130 PRINT "DISQUETE INICIALIZADO NO DOS 3.3"
140 PRINT "UNITRON AP II COM 48K"
150 PRINT "EM:23 OUTUBRO 84"
160 PRINT "SUSYANE DE SOUZA PEREIRA"
170 PRINT "ALUNA DO INSTITUTO SULLIVAN"
180 END
```

5) Rode agora o programa e melhore a formatação np vídeo se for necessária;

6) Digite agora INIT HELLO e dê RETURN (o acionador da drive entrará em movimento aproximadamente por 40 segundos e após este tempo o cursor (prompt) aparecerá no vídeo e a inicialização estará completada).

HELLO ficou sendo o nome do programa, porém poderia ter sido dado um outro nome qualquer de até 30 caracteres começado por uma letra. Pode-se incluir quaisquer caracteres, exceto vírgulas e caracteres de controle.

O programa presente na memória é guardado no disquete sob o nome de arquivo DADO. Este programa será o programa de saudação e será executado automaticamente sempre que o disquete for carregado.

Ao disquete é associado o número de volume (Vn) com que ele é inicializado; se não se informar nenhum número de volume (de 1 a 254), associa-se ao disquete o número de volume 254.

Vn o (n) número de volume (1 a 254) pode ser usado para evitar que disquetes sejam regravados acidentalmente.

Suponhamos que você possua um sistema de movimentos mensais

gravado em disquete, onde os registros de cada mês estão em diferentes disquetes, todos de volumes diferentes.

Portanto, quando você for entrar com os dados de um certo mês, terá de se certificar se foi especificado o volume certo dado ao disquete, caso contrário as informações não serão escritas e ocorrerá a mensagem de erro: VOLUME ILEGAL.

Dn e Sn (se o número da drive (1 ou 2) não for especificado o drive 1 é assumido como correto). A primeira interface de drive é usualmente conectada no slot de número 6 e geralmente não precisa ser especificada. Devemos especificar o número do slot (soquete de 1 a 7) caso ele esteja em números diferentes.

OBSERVAÇÃO:

O comando INIT nomearq deve ser usado no modo direto e sendo um comando do sistema operacional com disquete, o DOS já deve estar carregado na parte superior da RAM.

VARIAÇÕES DE USO

Não se tem conhecimento de nenhuma.

VEJA TAMBÉM

LOCK, LOAD, UNLOCK

Função **INKEY$**

SINTAXE GERAL

Número de linha, instrução IF função INKEY$

INKEY$ é uma função utilizada no modo programa geralmente precedida pela instrução LET, PRINT ou IF.

A função INKEY$ é utilizada para ler um carácter do teclado cada vez que se execute essa função.

Esta função faz uma varredura no teclado e fornece uma string de um carácter que foi teclado antes da varredura.

Ao contrário da instrução INPUT, INKEY$ não detem a execução enquanto se espera para que seja pressionada a tecla ENTER RETURN ou equivalente. O computador continua simplesmente a "circular" até que receba uma mensagem do teclado. Até que seja apertada uma tecla do teclado, INKEY$ dá simplesmente a leitura de um string "vazio" (isto é, código ASCII de 0).

Uma vez que INKEY$ não espera para você dar entrada num carácter a partir do teclado, e dê a instrução ENTER, é colocada geralmente num laço de programa para poder varrer repetidamente o teclado em busca de uma tecla que tiver sido apertada.

Esta função é muito útil quando se deseja colocar os valores sem precisar apertar a tecla ENTER, RETURN ou equivalente.

Por Exemplo:

```
100 PRINT "APERTE UMA LETRA"
110 A$ = INKEY$
120  IF  A$ = "" THEN 110 ELSE PRINT "VOCE APERTOU A TECLA "; A$ "NAO FOI?"
130 GOTO 100
```

A função INKEY$ procura repetidamente a letra no teclado para satisfazer a condição da instrução IF-THEN. Ao se dar entrada na letra X, fica satisfeita a condição da instrução IF-THEN, e o computador desvia o controle para a linha 120.

EXEMPLO 1

```
100 REM USANDO INKEY$
110 CLS
120 PRINT "PRESSIONE QUALQUER TECLA NO TECLADO"
130 X$ = INKEY$
140 IF X$ = "" GOTO 130
150 PRINT "VOCE ACABA DE PRESSIONAR A TECLA ";X$;
160  PRINT:PRINT  "PRESSIONE A TECLA ";X$;" MAIS UMA VEZ PARA COMECAR DE NOVO"
170 IF INKEY$ = X$ GOTO 110
180 GOTO 170
190 END
```

EXECUÇÃO DO PROGRAMA

```
RUN
PRESSIONE QUALQUER TECLA NO TECLADO
VOCE ACABA DE PRESSIONAR A TECLA T
```

PRESSIONE A TECLA T MAIS UMA VEZ PARA COMEÇAR DE NOVO

ORTOGRAFIAS ALTERNATIVAS

INKEY$ esta função nos computadores que seguem a linha APPLE tem a mesma função que GET (string) exceto que GET (string) pára a execução do programa até uma tecla ser apertada e INKEY$ não. Nos micros da linha APPLE, para se ter o mesmo efeito, usar:

X = PEEK (-16384): IF X >127 THEN GOTO XXX

(IF X >127 significa que alguma tecla foi digitada).

Se o conteúdo deste endereço for maior do que 127, isto indica que a tecla foi pressionada. Neste caso, o valor encontrado neste endereço será o valor ASCII da tecla mais 128, isto é:

conteúdo de -16384 = ASC ("tecla pressionada") + 128.

VARIAÇÕES DE USO

Não se tem conhecimento de nenhuma.

VEJA TAMBÉM

INPUT, IF-THEN, INPUT$, GET

Função **INP**

SINTAXE GERAL

Número da linha PRINT INP(número inteiro de 0 a 255)

INP é uma função utilizada no modo programa geralmente precedida pela instrução PRINT, sendo aceita por alguns interpretadores BASIC.

INP (porta) abre o canal de INPUT para um PORT (slot).

INP(porta) esta função fornece um valor byte da porta especificada. Há 256 portas numeradas de 0 a 255.

A instrução INP é utilizada para ler o valor decimal de um BYTE de informações em determinada porta do computador, podendo o valor em BYTES ser qualquer número inteiro positivo desde 0 até 255.

Exemplo: 10 PRINT INP(15) a função INP lê o byte que está na porta 15 e a instrução PRINT faz com que seja impresso o valor (em decimal) no vídeo.

INP(porta) geralmente a porta 255 já é usada pelo computador para a interface de cassete.

PRINT INP (X), por exemplo, imprime o valor decimal do BYTE na porta X.

INP é ferramenta útil para montagem de portas com respeito a determinada condição como, por exemplo, um pedido de entrada proveniente de um dispositivo periférico distante. Outras aplicações poderiam incluir as leituras recebidas de sensores remotos, num sistema de aquecimento de água por energia solar, etc.

EXEMPLO 1

```
100 REM USANDO FUNCAO INP
110 FOR A = 0 TO 255
115 PRINT "O VALOR DECIMAL DO BYTE NA PORTA No. ";
120 PRINT A;" E"; INP(A)
130 NEXT A
140 END
```

EXECUÇÃO DO PROGRAMA

```
RUN
O VALOR DECIMAL DO BYTE NA PORTA No. 0 E 63
O VALOR DECIMAL DO BYTE NA PORTA No. 1 E 15
                  .
                  .
O VALOR DECIMAL DO BYTE NA PORTA No. 255 É 127
```

VARIAÇÕES DE USO

INP é utilizado em certas versões do PDP-8 como abreviatura da instrução INPUT; nos micros que seguem a linha APPLE II usa-se IN# (número slot)

VEJA TAMBÉM

OUT, PEEK, POKE, INPUT, PIN

Instrução **INPUT • IN. • I.**

SINTAXE GERAL

Número da linha INPUT variável
ou
Número da linha INPUT variável,variável,...,variável

INPUT é utilizado apenas como instrução no modo indireto (programa).

Esta instrução está disponível em todas as versões BASIC para qualquer equipamento, para aquisição de dados a serem introduzidos em uma ou mais variáveis, numéricas ou alfanuméricas, simples ou subscritas, via teclado.

Quando uma linha de programa que contém uma instrução INPUT é encontrada, esta instrução provoca uma parada na execução do programa até que o usuário dê entrada em um "INPUT" solicitado.

INPUT não pode ser usado no modo direto.

INPUT é uma instrução de entrada que permite atribuir um valor a uma variável, através do teclado, durante a execução do programa. É um meio de comunicação entre o operador e o computador durante o processamento.

INPUT pode especificar uma variável numérica ou alfanumérica (String), ou ainda uma lista de variáveis numéricas e/ou alfanuméricas (Strings) a serem introduzidas no programa. As variáveis deverão estar separadas por vírgulas.

Exemplo: 100 INPUT X,A$,Y,B$

Quando for executada a instrução INPUT da linha 100, o computador gera um endereço de memória para a variável numérica X, coloca na tela do vídeo um sinal de interrogação(?) e espera que o operador introduza um valor numérico para X; em seguida gera um endereço de memória para a variável alfanumérica A$ apresentando novamente na tela interrogação (??) esperando que seja introduzido um valor alfanumérico para a variável A$, e assim sucessivamente para as demais variáveis do INPUT.

OBS.: Não podemos entrar com uma expressão dentro de um valor numérico, deve-se entrar com uma constante numérica simples. Se introduzirmos mais elementos de dados que a instrução INPUT especifica, o computador mostrará a mensagem:

? EXTRA IGNORADO
No exemplo 100 INPUT X,A$,Y,B$

A instrução INPUT pede que se entre com quatro valores (conforme suas variáveis: numéricas ou alfanuméricas). Em alguns computadores os valores podem ser colocados de uma só vez separados por vírgula - em outros devem ser entrados um por vez.

Convém observar que em alguns computadores a instrução INPUT só admite uma única variável. No exemplo acima seria necessário utilizar quatro vezes a instrução, uma para cada variável:

```
100 INPUT X
102 INPUT A$
104 INPUT Y
106 INPUT B$
```

EXEMPLO 1

```
100 REM USANDO A INSTRUCAO INPUT
110 PRINT "TECLE SEU NOME E SUA IDADE"
120 INPUT N$,I
130  PRINT  "OLA ," N$; " VOCE TEM NO MINIMO" I * 365; " DIAS
DE VIDA"
140 END
```

EXECUÇÃO DO PROGRAMA

```
RUN
TECLE SEU NOME E SUA IDADE
? SONIA
?? 39
OLA  SONIA VOCE TEM NO MINIMO 14235 DIAS DE VIDA
```

VARIAÇÕES DE USO

Muitos interpretadores BASIC permitem que a instrução INPUT possa ser utilizada executando as capacidades tanto de PRINT como de INPUT. Assim podemos incluir uma "MENSAGEM PRONTA" na instrução INPUT, tornando mais fácil entrar com os dados corretamente.

A "MENSAGEM PRONTA" deve seguir imediatamente o INPUT, estar entre aspas e seguida por um ponto e vírgula(;).

EXEMPLO 2

```
200 REM USANDO A INSTRUCAO INPUT/PRINT
210 INPUT "TECLE SEU NOME E SUA IDADE "; N$,I
220  PRINT  "OLA, ";N$; " VOCE TEM NO MINIMO " I * 365 " DIAS
DE VIDA "
230 END
```

EXECUÇÃO DO PROGRAMA

```
RUN
TECLE SEU NOME E SUA IDADE  ROSSANA
?? 41
OLA, VOCE TEM NO MINIMO 14965 DIAS DE VIDA
```

Experimente rodar no seu computador este exemplo e verifique se a linha 210 foi executada. Se foi executada, observe que nenhuma instrução PRINT precedeu a instrução INPUT, ambas as funções PRINT/INPUT foram conjugadas na linha 210.

ORTOGRAFIAS ALTERNATIVAS

Alguns computadores permitem IN como abreviatura de INPUT, outros aceitam INP e versões BASIC para o Nível I dos computadores TRS-80 aceitam I.

Nas versões BASIC APPLESOFT e INTEGER BASIC a sintaxe geral em que o INPUT aceita a mensagem pronta para esclarecimento apresenta-se da seguinte forma:

APPLESOFT

```
INPUT "MENSAGEM"; VAR, VAR
```

INTEGER BASIC

```
INPUT "MENSAGEM"; VAR,VAR
```

Se forem digitados na entrada caracteres inaceitáveis (ou seja, letras em valores numéricos) aparecerão as mensagens de advertência:

REENTER (para APPLESOFT) e
SINTAX ERR e reexecuta a instrução INPUT a partir de seu início.

VEJA TAMBÉM
INPUT1, INP, INKEY$, LINEINPUT, INPUTLINE, INPUT$

Instrução **INPUTLINE**

SINTAXE GERAL

Número da linha INPUTLINE variável alfanumérica.

INPUTLINE é utilizado como instrução no modo programa sendo aceito por apenas alguns interpretadores BASIC.

INPUTLINE se assemelha a INPUT, sendo utilizada em BASIC AVANÇADO (estendido) ou em algumas versões BASIC, para se aceitar uma LINHA inteira de entrada, atribuindo-a a uma única variável de string. INPUTLINE faz sugestão ao usuário (mediante um ponto de exclamação) da mesma forma que INPUT. O string que é dado como "INPUT" pode incluir vírgulas, dois pontos, e outros caracteres especiais sem necessidade de serem incluídos entre aspas.
INPUTLINE se assemelha a LINEINPUT.

```
EXEMPLO 1
100 REM USANDO A INSTRUCAO INPUTLINE
110 PRINT "TECLAR SEU SOBRENOME, PRIMEIRO":
120 INPUTLINE X$
130 PRINT "ALO, "N$;", EU SOU UMA MAQUINA"
140 END
```

```
EXECUÇÃO DO PROGRAMA
RUN

TECLAR SEU SOBRENOME, PRIMEIRO: XAVIER, ANTONIO
ALO, XAVIER, ANTONIO EU SOU UMA MAQUINA
```

VEJA TAMBÉM
LINEINPUT, INPUT

Função **INPUT $**

SINTAXE GERAL

Número de linha INSTRUÇÃO (LET) = INPUT$(número de caracteres)

INPUT$ é uma função utilizada no modo programa devendo ser precedida por uma instrução (geralmente o LET). Esta função é reconhecida por apenas alguns interpretadores BASIC.

INPUT$ (N) é função BASIC Microsoft utilizada para ler determinado número de caracteres a partir do teclado sem os exibir na tela. A tecla ENTER, CR ou RETURN não precisa ser pressionada depois de se dar entrada no último carácter mas o número correto de caracteres deve ser teclado antes que o programa possa prosseguir.

EXEMPLO: X$= INPUT(4) faz parar a execução do programa enquanto não se der entrada a quatro caracteres.

EXEMPLO 1

```
100 REM USANDO A FUNCAO INPUT$( )
110 PRINT "TECLAR QUATRO CARACTERES"
120 B$ = INPUT$(4)
130 PRINT "VOCE ACABA DE TECLAR";B$
140 END
```

EXECUÇÃO DO PROGRAMA

```
RUN
TECLAR QUATRO CARACTERES
D
C
V
R
```

VOCÊ ACABA DE TECLAR DCVR

OBS: Esta função é encontrada em algumas versões do BASIC funcionando apenas em alguns computadores. (Verifique o manual do seu computador.)

VARIAÇÕES DE USO

Algumas vezes nos vemos em situação difícil quando necessitamos introduzir dados, em nossos programas, que contenham vírgulas, dois pontos, carácter de controle e outros; como por exemplo endereços, horários etc... Isto ocorre porque o comando INPUT não aceita este tipo de entrada, originando uma string truncada e uma mensagem de erro "?EXTRA IGNORED".

Além disso, caso não tenhamos uma certa familiarização com o uso de nosso equipamento, o INPUT pode ter características próprias de edição, pode nos levar a certos tropeços. Estes por sua vez podem ser originados quando estamos voltando o cursor para o início do campo e ultrapassamos seu limite inicial, ou então quando digitamos mais de 255 caracteres (máximo permitido

pelo comando), ou ainda quando sobrepomos o limite final do campo. Estes "pequenos" detalhes poderão causar a interrupção do programa e a destruição da formatação da tela.

É a isso que se propõe a rotina abaixo, chamada INPUT$; não é um comando novo, mas sim uma série de testes sobre cada carácter que é digitado levantando sua validade ou não.

Esta rotina pode ser incluída em qualquer programa facilmente pois, por estar escrita em BASIC e não em linguagem de máquina, não requer nenhum procedimento inicial diferente.

Inicialmente colocamos o cursor na posicão onde deverá ser digitado o campo. Então, na linha do programa que está sendo usado, onde se encontra INPUT "";A$ COLOCAMOS gosub 150:A$=L$.

Apesar do uso do comando GET, esta rotina é suficientemente rápida para não permitir que haja truncamento no campo digitado. As teclas de recuar e avançar o cursor podem ser usadas normalmente, só que agora o movimento do cursor se limita aos caracteres já digitados, não ocasionando a extrapolação dos limites do campo. Em adição podem ser utilizados CTRL-X e CTRL-C para se posicionar o cursor respectivamente na primeira e na última posição digitada. Após termos o campo desejado sem erros, digitamos RETURN, sendo que todos os caracteres que estiverem após a posição do cursor e pertençam ao campo (caracteres errados) serão apagados com a formatação da tela permanecendo intacta.

A rotina tem um alarme a partir do 249ºcarácter digitado e qualquer carácter que seja introduzido apos o 255º será rejeitado e não haverá interrupção da rotina. Entretanto, este limite poderá ser alterado para a quantidade necessária em cada caso. Basta que alteremos o valor de L na linha 150, pois H é a posição inicial do campo a ser digitado. Isto previne quanto à possível destruição da formatação da tela.

Na linha 241 foi dada a opção para se digitar o colchete esquerdo "{", não disponível no teclado, utilizando-se CTRL-K. O mesmo ocorre nas linhas 242 e 243 onde obtemos a barra invertida " \ " e o carácter sublinhando "-" digitando-se respectivamente CTRL-B e CTRL-E.

A rotina propriamente dita está compreendida entre as linhas 150 e 340. Para que ela pudesse servir para teste foram incluídas as linhas 140,380,390 e 400.

Esta rotina também está habilitada a ler dados de arquivo em disco contendo vírgulas. Esta opção é acessada através de GOSUB 350: A$=L$. Recomenda-se o cuidado para que a rotina de leitura envie Carriage Return mais CTRL-D antes de fechar o arquivo, para evitar problemas relativos ao uso do GET na leitura de arquivos em disco.

Como sugestão, define-se D$ como CHR$(13)+CHR$(4).

Após o uso de ambas as rotinas, o conteúdo do campo estará em L$, para que seja feita sua transferência antes do novo uso da rotina.

```
EXEMPLO
130 REM USANDO COMANDO INPUT$ EXPANDIDO
140 GOTO 380
150 H = PEEK(36)+1: V = PEEK(37)+1: L$ = "": L = 256-H
160 GET A$: IF A$ = CHR$(13) THEN FOR K = 1 TO LEN (R$): PRINT " ";: NEXT K: HTAB H: X = FRE (0): RETURN
210 IF A$ = CHR$(8) THEN R$ = "X"+ RIGHT$ (L$,1) + R$: IF L$ > "" THEN H = H -1: L$ = MID$ (L$,1,LEN(L$)-1)
```

```
     220  IF  A$  =  CHR$(21) AND R$ > "" THEN H = H + 1: L$ =L$ +
LEFT$ (R$,1)
     230  IF  A$ = CHR$(24) THEN H = H - LEN (L$): R$ = "X" + L$ +
R$: L$ = ""
     240  IF  A$ = CHR$(3) THEN H = H + LEN (R$): L$ = L$ + R$: R$
= ""
     241 IF A$ = CHR$(11) THEN A$ = CHR$(91)
     242 IF A$ = CHR$(5) THEN A$ = CHR$ (92)
     243 IF A$ = CHR$(2) THEN A$ = CHR$(95)
     250 IF A$ < CHR$(32) THEN A$ = "": H-1
     320  IF LEN (L$) > L-8 THEN PRINT CHR$(7);: IF LEN (L$) + LEN
(A$) = L THEN PRINT CHR$(7);: GOTO 160
     330 PRINT A$;: H = H+1: VTAB V: HTAB H: L$ = L$+ A$
     340 R$ = MID$ (R$,2): GOTO 160
     350 L$ = "" M$ = CHR$(13)
     360 GET A$: IF A$ <> M$ THEN L$ = L$ + A$: GOTO 360
     370 X = FRE (0): RETURN
     380 HOME: VTAB 5: HTAB 5: PRINT "TESTE : ";
     390 GOSUB 150: A$ = L$
     400 VTAB 15: HTAB 13: PRINT A$
     410 END
```

ORTOGRAFIAS ALTERNATIVAS
Não se tem conhecimento de nenhuma.

VEJA TAMBÉM
INKEY$, KEY$, GET, INPUT

Instrução **INPUT 1**

SINTAXE GERAL

Número da linha INPUT1 variável

INPUT1 é utilizado como instrução no modo programa sendo aceito por apenas alguns interpretadores BASIC.

A instrução INPUT1 é utilizada por alguns computadores norte-americanos e europeus na versão MAXI-BASIC, de maneira semelhante às instruções INPUT, com a diferença de que INPUT1 detém o retrocesso e o espaçamento da linha depois que os dados INPUT tiverem sido atribuídos a uma variável.

EXEMPLO 1

```
100 REM USANDO A INSTRUCAO INPUT1
110 PRINT "ENTRE COM UM VALOR PARA A VARIAVEL N"
120 INPUT1 N
130 PRINT "VARIAVEL N = "; N
140  PRINT  "INPUT1  FOI  SATISFATORIO  SE SEU MICRO ACEITA A
VARIAVEL N = "; N
150 PRINT "FOREM IMPRESSAS NA MESMA LINHA QUE O SINAL DE ?"
160 END
```

EXECUÇÃO DO PROGRAMA (Utilizando-se 12)

```
RUN
ENTRE COM UM VALOR PARA A VARIAVEL N
? 12 VARIAVEL N = 12
INPUT1  FOI  SATISFATORIO  SE SEU MICRO ACEITA A VARIAVEL N =
12
FOREM IMPRESSAS NA MESMA LINHA QUE O SINAL DE ?
```

VARIAÇÕES DE USO

Não se tem conhecimento de nenhuma.

VEJA TAMBÉM

INPUT, INP, INPUTLINE, LINEINPUT

Função **INSTR**

SINTAXE GERAL

Número de linha LET variável = INSTR variável

INSTR é uma função usada geralmente no modo programa devendo ser precedida de uma instrução como LET ou PRINT.

INSTR(I,X$,Y$) é função de string que localiza a posição inicial da primeira ocorrência do string Y$ dentro do string X$. Busca a primeira ocorrência da cadeia Y$ e X$ e devolve à posição em que a ocorrência for encontrada.

O parâmetro I, opcional, estabelece a posição para iniciar a busca. I deve entrar na faixa de 0 a 255.

Se X$ for nulo ou se Y$ não puder ser encontrado, INSTR devolve 0. Se Y$ for nulo, INSTR devolve I ou 1(um).

```
EXEMPLO 1
100 REM USANDO A FUNCAO INSTR
110 X$ = "ABCEDB"
120 Y$ = "B"
130 PRINT INSTR (X$,Y$);INSTR (4,X$,Y$)
```

```
EXECUÇÃO DO PROGRAMA
RUN
2  6
```

```
EXEMPLO 2
200 REM USANDO A FUNCAO INSTR(I,X$,Y$)
210 X$ = "MICRO"
220 Y$ = "APPLE"
230 K = INSTR(4,X$,Y$)
240 IF K <> 7 THEN 270
250 PRINT "INSTR(I,X$,Y$) NAO FOI APROVADA"
260 GOTO 280
270 PRINT "INSTR(I,X$,Y$) FOI APROVADA"
3999 END
```

```
EXECUÇÃO DO PROGRAMA
RUN
INSTR(I,X$,Y$) FOI APROVADA
```

SE O SEU COMPUTADOR NÃO TIVER ESSA FUNÇÃO

Se falharam ambos os testes, experimente as funções POS e INDEX.

Se seu computador utiliza as funções MID$ e LEN, pode ser utilizada a seguinte sub-rotina:

```
3000 GOTO 3999
3060 REM *SUBROTINA INSTR* INPUT I,X$,Y$ SAIDA K
3062 REM UTILIZA TAMBEM L INTERNAMENTE
3064 L = LEN(Y$)
3066 FOR K = N TO LEN(X$)-L+1
3068 IF Y$ = MID$(X$,K,L) THEN 3074
```

```
3070 NEXT K
3072 K = 0
3074 RETURN
```

Para utilizar essa sub-rotina em conjunto com o exemplo 2 efetue as seguintes mudanças:

```
225 N = 4
230 GOSUB 3060
```

VEJA TAMBÉM
POS, INDEX, MID$, LEN

Função **INT • I**

SINTAXE GERAL

Número de linha LET variável = INT(argumento)
OU
INT (exprnm)

INT é uma função usada geralmente no modo programa devendo ser precedida de uma instrução como LET ou PRINT.

Esta função é aceita na maioria dos microcomputadores que seguem a linha SINCLAIR, TRS-80 ou APPLE, sendo que neste último não é disponível na versão INTEGER BASIC, por esta versão registrar automaticamente inteiros.

INT(X) esta função retorna o maior número inteiro igual ou menor que o número ou expressão X.

A função INTeger é utilizada para arredondar os números até seu valor de número inteiro. Na BASIC os números são sempre arredondados para baixo. O número inteiro permanece o mesmo seja qual for o valor dos números removidos da direita da vírgula decimal, exceção feita ao fato de que, sendo convertido um número inteiro em número negativo, o resultante é arredondado para o próximo número inteiro inferior. Por exemplo, INT(-4.65) se converte em 5.

Existem certos limites quanto ao tamanho do número que alguns computadores podem processar com a função INT. Certos microcomputadores não aceitam um número inferior a -32768 ou superior a +32767.

EXEMPLO 1

```
100 REM USANDO A FUNCAO INT
110 READ Y
120 PRINT "O VALOR DE NUMERO INTEIRO DE "; Y;
130 Y = INT(Y)
140 PRINT "E "; Y
150 IF Y = 180 THEN 180
160 GOTO 110
170 DATA 3.44,5.987,.65,-9.54,98765.324,-12345.986,123.87
180 END
```

EXECUÇÃO DO PROGRAMA

```
RUN
O VALOR DE NUMERO INTEIRO DE 3.44 E 3
O VALOR DE NUMERO INTEIRO DE 5.987 E 5
O VALOR DE NUMERRO INTEIRO DE .65 E 0
O VALOR DE NUMERO INTEIRO DE -9.54 E -10
O VALOR DE NUMERO INTEIRO DE 98765.3 E 98765
O VALOR DE NUMERO INTEIRO DE -12346 E -12346
O VALOR DE NUMERO INTEIRO DE 123.87 E 123
```

ORTOGRAFIA ALTERNATIVA

I. é utilizada pela maioria das Tiny BASIC como abreviatura de INT.

Adicionalmente, os microcomputadores que seguem a linha TRS-80 nos fornecem duas funções que são:

CINT(X) e FIX(X). A primeira faz a mesma coisa que o INT(X), porém com a ressalva de que X deve estar entre -32768 e +32767.

Esta função torna-se, assim, mais rápida. O FIX(X) será igual a INT(X) se X for positivo. Se X for negativo, FIX(X) será igual a INT(X)+1.

VARIAÇÕES DE USO

INT pode ser utilizado nos computadores que seguem a linha APPLE II para colocá-lo na versão INTEGER BASIC. Neste caso, qualquer programa INTEGER BASIC não está presente (por exemplo, num APPLE II PLUS sem o Language Card, a mensagem LANGUAGE NOT AVAILABLE aparecerá no vídeo).

INT, neste caso, deverá ser usado somente no modo direto.

VEJA TAMBÉM

CINT, FIX

Instrução/Comando **INVERSE**

SINTAXE GERAL

INVERSE ou
Número de linha, INVERSE.
INVERSE pode ser utilizado tanto como comando no modo direto como instrução no modo programa.

INVERSE é utilizado pelo APPLE II como comando ou como instrucão, para exibir a saída em tela no modo INVERSE (invertido).
INVERSE nos microcomputadores que seguem a linha APPLE II faz com que qualquer texto impresso sob seu controle fique no vídeo em letras pretas sobre um fundo branco. O comando ou instrução NORMAL do BASIC APPLESOFT faz com que a tela volte à posição anterior.
Todas as saídas subseqüentes a INVERSE, dadas com PRINT, aparecerão em caracteres pretos com fundo branco. As mensagens de erro também são afetadas. Entretanto, os caracteres mostrados na tela do vídeo devido às instruções INPUT não são afetados,bem como os caracteres mostrados anteriormente.
O INVERSE atua alterando um pouco o código ASCII, assim sendo qualquer carácter enviado para o disquete em Vídeo Reverso será guardado em códigos incorretos. Ao serem lidos de volta aparecerão errados.
INVERSE não é disponível na versão INTEGER BASIC.

EXEMPLO 1

```
100 REM USANDO INVERSE
110 PRINT"ISTO DEMONSTRA ";
120 INVERSE
130 PRINT "IMPRESSAO INVERTIDA,"
140 END
```

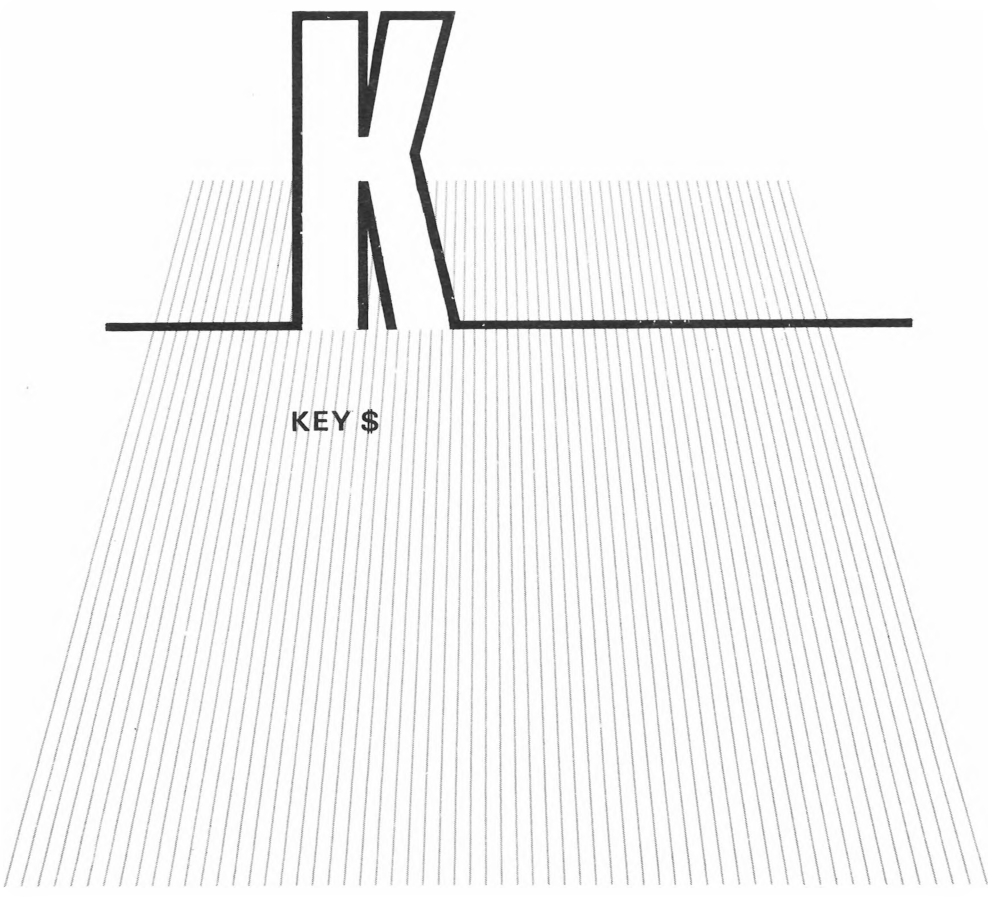
Para rodar este programa, limpe a tela e tecle RUN

EXECUÇÃO DO PROGRAMA

```
RUN
ISTO DEMONSTRA IMPRESSAO INVERTIDA
```

VARIAÇÕES DE USO
Não se tem conhecimento de nenhuma.

VEJA TAMBÉM
NORMAL, FLASH


KEY $

Função **KEY $**

SINTAXE GERAL

Número de linha, LET variável, KEY$

KEY$ é uma função usada geralmente no modo programa devendo ser precedida de uma instrução como LET ou PRINT.

A função KEY$ é utilizada em apenas algumas versões BASIC para dar entrada num carácter sem exibi-lo na tela. KEY$(n) aceita apenas um dentre três valores diferentes de n.

Verifique o manual do seu computador para saber se ele aceita esta função.

KEY$(0) lê o teclado, KEY$(1) lê o controle de jogo da mão direita e KEY$(2) lê o controle de jogo da mão esquerda.

A$ = KEY$(0), por exemplo, faz a varredura do teclado à procura de qualquer tecla que tiver sido pressionada. Se não tiver sido pressionada nenhuma tecla, A$ = ""(nulo), caso contrário A$ "lê" o carácter que for pressionado. KEY$(0) equivale a INKEY$.

EXEMPLO 1

```
100 REM USANDO KEY$(0)
110 CALL 17046 LIMPA A TELA EM BASIC APF
120 PRINT "TECLAR QUALQUER CARACTER"
130 X$ = KEY$(0)
140 IF X$ = "" THEN 130
150 PRINT "VOCE ACABA DE TECLAR ";X$
160 PRINT "PARA REPETIR, TECLAR NOVAMENTE ";X$;"
170 IF KEY$ (0) = X$ THEN 110
180 GOTO 170
190 END
```

EXECUÇÃO DO PROGRAMA (utilizando-se @)

```
RUN
TECLAR QUALQUER CARACTER
VOCÊ ACABA DE TECLAR  @
PARA REPETIR, TECLAR NOVAMENTE @
```

R$ = KEY$(1) verifica o controle de jogos da mão direita. Se não for pressionada nenhuma tecla, R$ = "". Sendo pressionada uma das teclas numéricas 0-9, R$ passa a representar esse carácter numérico.

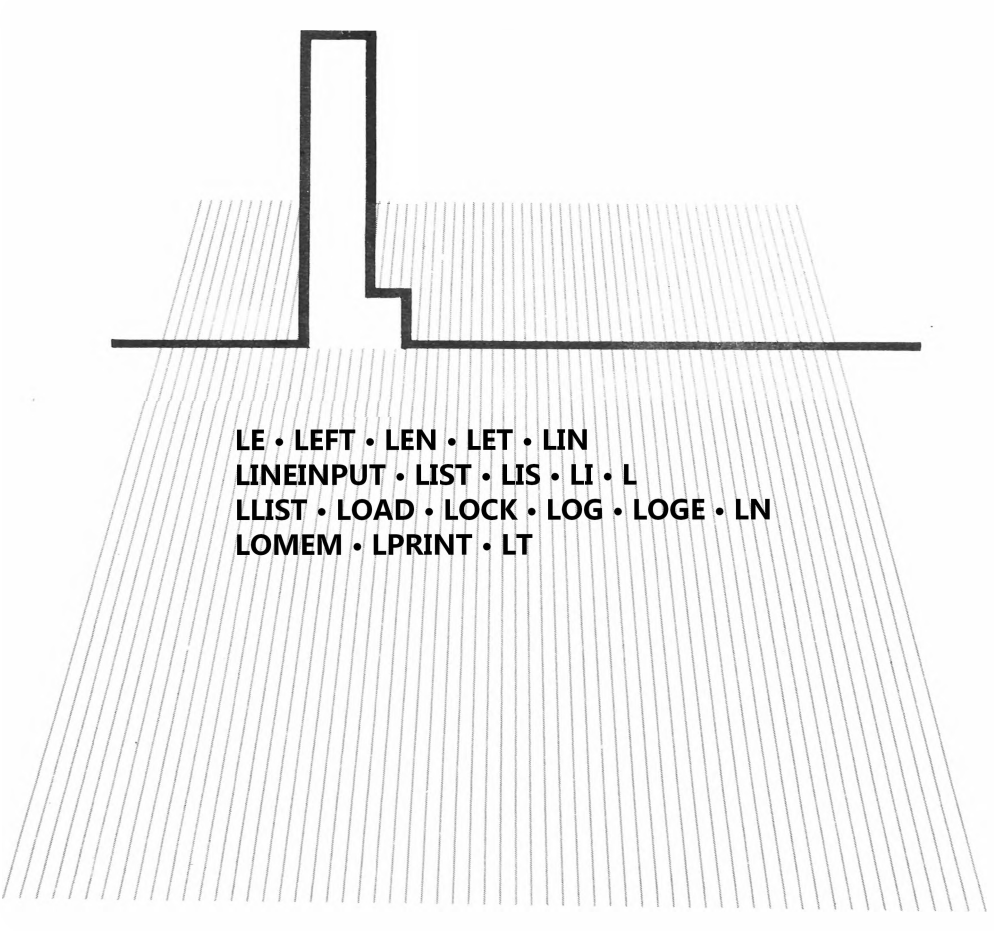
Finalmente, R$ recebe N, S, E ou W, para indicar em que sentido a maçaneta de direção foi empurrada. L$ = KEY$(2) lê o controle de jogo da mão esquerda de maneira semelhante.

ORTOGRAFIA ALTERNATIVA

KEY é utilizada por Texas Instruments TI99/4 numa instrução CALL (isto é, tecla CALL) para fazer a mesma coisa que KEY$(0).

VEJA TAMBÉM

INKEY$, PDL, GET, INPUT



LE • LEFT • LEN • LET • LIN
LINEINPUT • LIST • LIS • LI • L
LLIST • LOAD • LOCK • LOG • LOGE • LN
LOMEM • LPRINT • LT

Operador Lógico LE

SINTAXE GERAL

Número de linha IF valor, LE valor THEN expressão.

Os operadores lógicos fazem comparações lógicas e são normalmente usadas em instruções IF/THEN.

LE é utilizado em alguns computadores da linha TEXAS INSTRUMENTS como palavra alternativa para o sinal de "menor do que ou igual a" <=.

EXEMPLO 1

```
100 REM USANDO LE (MENOR DO QUE OU IGUAL A)
110 IF 20 LE 30 THEN 140
120 PRINT "O OPERADOR LE NAO FOI APROVADO NA LINHA 110"
130 GOTO 180
140 IF 30 LE 30 THEN 170
150 PRINT "O OPERADOR LE NAO FOI APROVADO NA LINHA 140
160 GOTO 180
170 PRINT "O OPERADOR LE FOI APROVADO"
180 END
```

EXECUÇÃO DO PROGRAMA

```
RUN
O OPERADOR LE FOI APROVADO
```

VARIAÇÕES DE USO

Não se tem conhecimento de nenhuma.

VEJA TAMBÉM

<=, <, >=, >, =, <>, EQ, LT, GT, GE, NE, IF-THEN

Função LEFT

SINTAXE GERAL

LEFT$ (expr$,exprnm)
Número de linha LET variável, função LEFT$ (variável alfanumérica, número)
LEFT$ (X$,X)

LEFT$ é uma função usada geralmente no modo programa devendo ser precedida de uma instrução como LET ou PRINT.

LEFT$ (X$,X) esta função é aceita na maioria das versões BASIC e faz com que o microcomputador reconheça certo número de caracteres de X$, a começar do lado esquerdo.

LEFT$ (expr$,exprnm) retorna o carácter mais à esquerda do valor contido na variável alfanumérica FXPR$, sendo que a exprnm deve representar um número inteiro entre 1 e 255.

Esta função não é disponível na versão INTEGER BASIC da linha APPLE II.

A função LEFT$ (string,X) é utilizada para extrair um número específico (X) de caracteres de string, a começar com o carácter mais para a esquerda no string.

PRINT LEFT$ ("ACEITANDO",6), por exemplo, imprime as letras ACEITA, que são os últimos 6 caracteres deixados no string ACEITANDO.

O string deve estar contido entre aspas ou listado como variável de string. O número de caracteres (X) pode ser expresso como uma variável, número ou operação aritmética, sendo que por vezes uma virgula deve separar o string do número.

Se o valor de (X) for fracionário, o microcomputador encontra automaticamente seu valor de número inteiro.

EXEMPLO 1

```
100 REM USANDO FUNCAO LEFT$
110 A$ = "AMANHA"
120 B$ = LEFT$ ("ACEITANDO",6)
130 PRINT LEFT$ (A$,1);" FUNCAO LEFT$ FOI ";B$
140 END
```

EXECUÇÃO DO PROGRAMA

```
RUN
A FUNÇÃO LEFT$ FOI ACEITA
```

ORTOGRAFIA ALTERNATIVA

Alguns BASIC, tais como MAX BASIC e DEC BASIC-PLUS, utilizam o LEFT (string,n) ao invés de LEFT$.

VARIAÇÕES DE USO

Não se tem conhecimento de nenhuma.

VEJA TAMBÉM

PRINT, RIGHT$, MID$, CHR$, STR$, STRING$, INKEY$, INSTR, PDS, SEG$, DEFSTR

Função LEN • L

SINTAXE GERAL

LEN (var alfa)
Número de linha PRINT função LEN (variável alfanumérica)

LEN (variável valor alfanumérico) e uma função usada geralmente no modo programa devendo ser precedida de uma instrução como LET ou PRINT.

LEN ("X") esta função retorna o número de caracteres na cadeia "X".

A função LEN é utilizada para medir o LENgth (comprimento) de strings, por contar o número de caracteres entre aspas ou atribuído a variáveis de string, retorna o comprimento do valor de uma variável STRING (alfanumérica).

10 PRINT LEN ("RETURN") deve imprimir 6, o número de letras na palavra RETURN.

Esta função não é disponível nas versões do BASIC para microcomputadores da linha TRS-80 no nível I.

LEN (expr$) conta o número de caracteres de expr$, incluindo todos os espaços e caracteres não imprimíveis.

Se houve alguma concatenação e expr$ tiver mais que 255 caracteres aparecerá uma mensagem de erro:
STRING TOO LONG ERROR

EXEMPLO 1

```
100 REM USANDO A FUNCAO LEN
110 PRINT "TECLAR QUALQUER COMBINACAO DE LETRAS E NUMEROS"
120 INPUT X$
130 PRINT "VOCE DEU ENTRADA EM ";X$" QUE CONTEM ";
140 PRINT LEN(X$);" CARACTERES"
150 END
```

EXECUÇÃO DO PROGRAMA (utilizando-se RPY765)

```
TECLAR QUALQUER COMBINACAO DE LETRAS E NUMEROS
? RPY765
VOCE DEU ENTRADA EM RPY765 QUE CONTEM 6 CARACTERES
```

ORTOGRAFIA ALTERNATIVA

O computador britânico ACORN ATOM permite L. como abreviatura de LEN.

VARIAÇÕES DE USO

Não se tem conhecimento de nenhuma.

VEJA TAMBÉM

ASC, FRE, LEFT$, MID$, RIGHT$, STR$, VAL, SEG$

Comando/Instrução **LET**

SINTAXE GERAL

LET VAR = EXPR
ou
Número da linha LET variável= Constante, Variável ou Expressão.

LET VARIÁVEL - Pode ser utilizado como comando no modo direto ou como instrução no modo programa.
LET VARIÁVEL - Atribui um valor numérico ou alfanumérico a uma variável numérica ou alfanumérica .
LET variável é aceita nas versões BASIC como instrução ou comando de atribuição de uma variável. Nos microcomputadores que seguem a linha SINCLAIR, o seu uso é obrigatório em declarações de atribuição . Porém, nos micros que seguem a linha APPLE ou TRS-80 e alguns outros o seu uso é opcional.
LET nao faz apresentação na tela, mas deixa a variável preparada para uso ou apresentação, se solicitada por uma outra instrução.

EXEMPLO 1

```
100 REM USANDO INSTRUCAO LET
110 LET A = 7
120 LET B = 5
130 LET C = (A + B)/2
140 END
```

EXECUÇÃO DO PROGRAMA

```
RUN
```

OBS.: Na execução do programa nenhum valor será apresentado na tela mas estarão armazenados na memória os valores para A=7, B=5 e C=6 (sendo que no cálculo da EXPRESSÃO C=(A+B)/2 o valor armazenado e apresentado será o resultado após efetuadas todas as operações indicadas à direita de C "nome da variável")

Para verificar isto use: (no modo direto)

```
PRINT A
PRINT L
PRINT C
```

EXEMPLO 2

```
200 REM LET ATRIBUINDO VALORES ALFANUMERICOS
210 LET A$ = "SEI "
220 LET B$ = "USAR "
230 LET C$ = "A INSTRUCAO LET"
240 PRINT A$;
250 PRINT B$;
260 PRINT C$
270 END
```

EXECUÇÃO DO PROGRAMA

```
RUN
```

SEI USAR A INSTRUÇÃO LET

OBS.: Quando se atribuem valores alfanuméricos a uma variável alfanumérica estes valores devem vir entre ASPAS.

Em grande parte das versões BASIC usadas atualmente, o LET é opcional, isto é, as variáveis podem ser declaradas sem escrever a instrução LET. (110 A = 7)

VARIAÇÕES DE USO:

Não se tem conhecimento de nenhuma.

Função/Instrução **LIN**

S:NTAXE GERAL

Número de linha LIN (X)

A instrução LIN(X) (utilizada nas versões do BASIC 2000 da Hewlett-Packard e nas BASIC Opus 1 e Opus 2 do Digital Group) faz com que o microcomputador pule um número específico (X) de linhas na impressora antes de imprimir a próxima linha.

```
EXEMPLO 1
100 REM USANDO LIN
110 PRINT "A INSTRUCAO LIN FOI APROVADA"
120 LIN (3)
130 PRINT "SE 3 LINHAS FOREM PULADAS ANTES QUE ESTA LINHA
SEJA IMPRESSA"
140 END
```

```
EXECUÇÃO DO PROGRAMA
RUN
A INSTRUCAO LIN FOI APROVADDA



SE 3 LINHAS FOREM PULADAS ANTES QUE ESTA LINHA SEJA IMPRESSA
```

SE O SEU COMPUTADOR NÃO TIVER ESTA FUNÇÃO OU INSTRUÇÃO

Se o seu computador não tiver LIN(X), pode-se facilmente simular a mesma substituindo-se LIN(X) numa série de instruções (X) de PRINT.

Por exemplo, substitua a linha 120 no exemplo 1 e coloque as linhas 115 e 122.

```
115 FOR R = 1 TO 3
120 PRINT
122 NEXT R
```

Uma vez que cada instrução PRINT provoca um espaçamento de linha, essas linhas farão com que o computador realize a mesma operação de LIN(3).

VARIAÇÕES DE USO

Não se tem conhecimento de nenhuma.

VEJA TAMBÉM

PRINT, VTAB

Instrução **LINEINPUT**

SINTAXE GERAL

Número de linha instrução LINEINPUT variável.

LINEINPUT é utilizada como instrução no modo programa.
LINEINPUT se assemelha a INPUT. Tem como finalidade aceitar toda uma linha de entrada (até 254 caracteres) para uma variável de string, sem delimitadores.
Não se imprime um ponto de interrogação, a menos que faça parte da cadeia de indicação. Toda entrada desde o fim da indicação até o retorno de carro é atribuída à variável de string. O string que dá como "input" pode incluir vírgulas, dois pontos e outros caracteres especiais sem necessidade de colocá-los entre aspas. Será aceito qualquer carácter menos o retrocesso.

```
EXEMPLO 1
100 REM USANDO A INSTRUCAO LINEINPUT
110 PRINT "TECLAR SEU NOME - DE FAMILIA, PRIMEIRO";
120 LINEINPUT N$
130 PRINT "ALO, ";N$;", EU SOU UMA MAQUINA"
140 END

EXECUÇÃO DO PROGRAMA
RUN
TECLAR SEU NOME - DE FAMILIA, PRIMEIRO DE PAULA, CIDA
ALO, DE PAULA, CIDA, EU SOU UMA MAQUINA
```

LINEINPUT tem capacidade de imprimir a sua própria mensagem de instigação ("prompt"). Apague a linha 110 e mude a linha 120 para:

```
120 LINEINPUT "TECLAR SEU NOME - DE FAMILIA, PRIMEIRO ";N$
```

Os resultados devem ser os mesmos de antes.

ORTOGRAFIA ALTERNATIVA
Alguns computadores permitem o uso da forma abreviada, .INPUT, no lugar de LINEINPUT.

VEJA TAMBÉM
INPUTLINE, INPUT

Comando/Instrução **LIST**

LIS • LI • L

SINTAXE GERAL

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

LIST pode ser utilizado como instrução no modo programa, mas é mais usado como comando no modo direto para apresentar a listagem de todo o programa que estiver na mesma RAM, de forma seqüencial pelos números de suas linhas .LIST X - lista o programa da linha X ate a última linha existente no programa.

LIST este comando funciona de maneira semelhante nas linhas, APPLE, TRS-80, SINCLAIR ou PC(IBM), porém quando seguido de indicações de números de linhas e operadores pode apresentar certas particularidades.

Nos computadores que seguem as linhas TRS-80 e APPLE vamos encontrar LIST X-Y (ou ainda LIST X,Y (APPLE), e LIST-X ou ainda LIST,X no APPLE).

No primeiro caso, são listadas as linhas que vão do número X ao número Y.

No segundo, o programa é listado a partir da sua primeira linha até a linha X. Nas diversas linhas de microcomputadores LIST X causará o aparecimento apenas da linha especificada.

O comando LIST é utilizado para exibir cada linha do programa na ordem numérica em que aparece no programa. Alguns computadores (ou terminais) farão rolar o programa inteiro a não ser que sejam detidos por determinada funcão de tecla (Control C, Control S, SHIFT @, etc.). Outros param depois de exibir as primeiras 12, 16, 24 ou mais linhas, avançando em seguida à razão de uma ou mais linhas adicionais cada vez que se pressione a tecla seta-ascendente, seta-descendente ou outra tecla apropriada.

O comando LIST pode também ser utilizado em conjunto com um número de linha para especificar um ponto de partida outro que não no princípio. Muitos computadores aceitam também números de início e fim.

LIST 100-150 ou LIST100-150, por exemplo, farão listar apenas aquelas linhas de programa com números que vão de 100 a 150.

```
EXEMPLO 1
100 REM USAMDO LIST
110 REM ESTE COMANDO
120 REM EXIBIRA TODA
130 REM LINHA DO PROGRAMA
140 PRINT "LIST APROVOU"
150 END
```

EXECUÇÃO DO PROGRAMA

```
RUN
LIST APROVOU
```

TECLAR LIST 140-150, ao que seu computador deve imprimir:

```
140 PRINT "LIST APROVOU"
150 END
```

Se o seu computador não aceitar as limitações de números de linha, experimente dar entrada em LIST 140.

Deve ser impressa a linha 140. Caso esse teste falhar, experimente dar entrada em LIST sem números de linha.

Alguns computadores irão exibir a linha 110 e todas as linhas que seguem, ao se dar entrada em LIST 110. Se o seu computador aceitar essa instrução, utilize LIST 110-110 ou LIST 110,110 para listar apenas a linha 110.

Experimente os seguintes comandos:

```
LIST
LIST -130
LIST 130-
```

Se o seu computador aceitou estes comandos LIST, LIST- deve ter listado o programa inteiro, LIST 130-, o programa a começar com a linha 130, e LIST-130 o programa a começar com a primeira linha e findar com a linha 130. Se esses comandos falharem, experimente utilizar vírgulas (,) no lugar de tracinhos (-).

LIST é aceito por alguns computadores como instrução de programa. Para testar isto no seu computador, adicione ao EXEMPLO 1 a linha seguinte:

```
145 LIST
```

Se LIST for aceito como instrução de programa, irá imprimir:

```
RUN
LIST APROVOU
100 REM USANDO LIST
110 REM ESTE COMANDO
120 REM EXIBIRA TODA
130 REM LINHA DO PROGRAMA
140 PRINT "LIST APROVOU"
145 LIST
150 END
```

ORTOGRAFIA ALTERNATIVA

Estão em uso diversas abreviaturas, inclusive LIS, LI (Texas Instruments 990) e L. (Tiny BASIC)

VEJA TAMBÉM

LLIST

Comando/Instrução **LLIST**

SINTAXE GERAL

Número de linha LLIST

LLIST pode ser utilizado como comando no modo direto ou como instrução no modo programa.

Sua finalidade é fornecer a listagem do programa através da impressora, nos microcomputadores que seguem a linha APPLE II deve-se usar PR#1 e então LIST.

LLIST é utilizado na BASIC Microsoft para listar o programa atualmente residente na memória, por meio da impressora ao invés da tela, LLIST é utilizada mais freqüentemente como comando, mas pode também ser utilizada como instrução dentro de um programa.

Funcionam com LLIST todas as opções disponíveis a LIST.

LLIST supoe uma impressora com 132 posições de impressão.

O BASIC sempre volta ao nível de comando após a execução de um LLIST. As opções para LLIST são as mesmas para LIST.

LLIST 150- lista todas as linhas de 150 ate o fim.

LLIST-1000 lista todas as linhas desde o número mais baixo até 1000.

LLIST 150-1000 lista as linhas de 150 até 1000 inclusive.

Uma vez que a listagem está sendo produzida em papel e não na tela de vídeo, não vai parar depois de 12, 16 ou 24 linhas (conforme pode ocorrer com LIST) mas irá continuar até o final do programa ou até que se pressionem as teclas BREAK, RESET, Control C, etc.

ADVERTÊNCIA: A impressora tem que estar ligada ao computador e estar pronta para operar antes de se teclar o comando LLIST.

EXEMPLO 1

```
100 REM USANDO O COMANDO LLIST
110 REM O COMANDO LLIST
120 REM IRA IMPRIMIR ESTE PROGRAMA
130 REM NA IMPRESSORA
140 END
```

EXECUÇÃO DO PROGRAMA

Teclar LLIST deve produzir na sua impressora a listagem acima. Teclar LLIST 110-120 deve imprimir

```
110 REM O COMANDO LLIST
120 REM IRA IMPRIMIR ESTE PROGRAMA
```

Teclar LLIST -110 deve listar apenas as duas primeiras linhas, ao passo que LLIST -130 deve listar apenas as últimas duas linhas. Para verificar se LLIST pode ser utilizada como instrução de programa, acrescente:

135 LLIST ao exemplo 1 e execute um RUN. Se LLIST for aceita, deverá ser impresso o **seguinte:**

```
100 REM USANDO O COMANDO LLIST
110 REM O COMANDO LLIST
120 REM IRA IMPRIMIR ESTE PROGRAMA
130 REM NA IMPRESSORA
135 LLIST
140 END
```

VEJA TAMBÉM

LIST, LPRINT

Comando **LOAD**

SINTAXE GERAL

LOAD

LOAD deve ser utilizado no modo programa para provocar um sinal do microcomputador para o periférico em busca de dados. A execução pode ser imediata ou deferida.

Nos microcomputadores que seguem a linha TRS-80 LOAD deve ser substituído por CLOAD

O comando LOAD é utilizado para carregar um programa no computador a partir de fita cassete.

Programa de Teste

Dê entrada no programa ao computador a partir do teclado, e armazene-o em seguida em fita cassete. (Ver os detalhes na seção SAVE).

```
100 REM PROGRAMA DE TESTE LOAD
110 PRINT "ESTE PROGRAMA TESTA A CARACTERISTICA LOAD"
120 END
```

Depois que o programa tiver sido registrado em fita cassete, apague a memória do computador pelo comando NEW, ou qualquer outro que seja apropriado.

Rebobine a fita, regule o gravador em seguida no modo play e tecle o comando LOAD.

O motor do gravador cassete é controlado pelo computador que o liga e desliga antes e depois do ciclo LOAD. A cassete deve "tocar" o programa, fazendo o LOAD para o computador.

Liste o programa para ter certeza de que o programa armazenado na memória do computador é idêntico àquele a que se deu entrada originalmente. Se tudo parecer bem, faça rodar (RUN) o programa.

Caso se deseje, pode-se dar ao programa um nome ao ser SAVE (conservado). Outros programas serão então ignorados, sendo Carregado (LOAD) o programa especificado, ao ser encontrado.

Exemplo: LOAD "ROSA"

EXECUÇÃO DO PROGRAMA

```
ESTE PROGRAMA TESTA A CARACTERÍSTICA LOAD
```

VARIAÇÕES DE USO

LOAD "nome de arquivo" é utilizado freqüentemente para carregar (LOAD) um programa antes conservado em disquete. É preciso atribuir um nome de arquivo.

Microcomputadores que seguem a linha APPLE

SINTAXE

```
LOAD nomearq [,Dn] [,Sn] [,Vn]
```

O programa de nomearq é carregado a partir do disquete. Se a operação for bem-sucedida, o programa anterior da memória é apagado.

Se o arquivo não existir no drive Dn do slot Sn, a mensagem

FILE NOT FOUND aparece.

Se o disquete no drive é slot Sn não tiver o Vn, o erro VOLUME MISMATCH aparecerá no vídeo. Dn, Sn, e Vn podem ser especificados em qualquer ordem. Se Dn e Sn são omitidos, o último drive ou slot referenciado e usado.

VO é adotado quando Vn estiver ausente. LOAD nomearq é um comando do DOS e deve ser precedido de PRINT CHR(4) e ser usado no modo programa.

VEJA TAMBÉM

CLOAD, SAVE, CSAVE, LIST, NEW

Comando/DOS **LOCK**

SINTAXE GERAL

LOCK nomearq [,Dn] [,Sn] [,Vn]

LOCK protege um arquivo em disco de eliminação acidental. É um comando DOS, requerendo PRINT e CTR-D em modo programado.

Habitualmente procura-se evitar que arquivos sejam destruídos: o comando LOCK se encarrega disso.

Exemplo: ARQ.1

protege o programa ARQ.1 de possíveis acidentes. Uma vez protegido, um arquivo não pode ser eliminado ou ter seu nome trocado até que seja desprotegido pelo comando UNLOCK.

Um arquivo protegido é indicado pela listagem emitida pelo comando CATALOG; aparecerá um asterisco(*) à esquerda do tipo do arquivo.

Nenhum programa pode ser guardado com o nome igual ao de um programa protegido.

Dn,Sn,Vn podem ser especificados em qualquer ordem. Se Dn e Sn são omitidos, o último drive ou slot referenciado é usado.

VARIAÇÕES DE USO

Não se tem conhecimento de nenhuma.

VEJA TAMBÉM

UNLOCK, INIT, SAVE, LOAD

Função LOG • LOGE • LN

SINTAXE GERAL

Número de linha LOG (exprnm)
ou
Número de linha LET variável = LOG (X)

LOG é uma função usada geralmente no modo programa devendo ser precedida de uma instrução como LET ou PRINT.

A função LOG(X) funciona de maneira idêntica nas diversas linhas de microcomputadores em que é aceita. A expressão LOG devolve o logaritmo natural da expressão. Esta função não está disponível nos microcomputadores que seguem a linha TRS-80 de nível 1, e nem na versão INTEGER BASIC da linha APPLE.

A função LOG(X) calcula o logaritmo natural de qualquer número (X) cujo valor seja superior a 0. Para logaritmos a base comum (10), veja LOG10 ou CLG.

Se X for zero ou negativo, aparecerá a mensagem ILLEGAL QUANTITY ERROR.

EXEMPLO 1

```
100 REM USANDO A FUNCAO LOG
110 PRINT "DE ENTRADA NUM NUMERO POSITIVO ";
120 INPUT R
130 L = LOG(R)
140 PRINT "O LOGARITMO NATURAL DE " ;R;" E ";L
3999 END
```

EXECUÇÃO DO PROGRAMA

```
RUN
DE ENTRADA NUM NUMERO POSITIVO? 80
O LOGARITMO NATURAL DE 80 E 4.38203
```

ORTOGRAFIA ALTERNATIVA

Alguns BASICs utilizam IN (como, por exemplo, a BASIC de Micropolis), alguns utilizam LOGE, e uns poucos (tais como DEC-10) utilizam todas as três formas do logaritmo natural.

SE SEU COMPUTADOR NÃO TIVER NENHUMA DELAS:
Se todos falharem, use a seguinte sub-rotina:

```
3000 GOTO 3999
3150 REM SUB-ROTINA DE LOGARITMO NATURAL INPUT X, SAIDA L
3152 REM UTILIZA INTERNAMENTE P,M,N,T,S
3154 IF X > 0 THEN 3160
3156 PRINT "LOGARITMO NAO DEFINIDO EM  ";X
3158 STOP
3160 P = 1
3162 M = 2
3164 N = .5
3166 T = X
3168 S = 0
```

Comando **LOMEM**

SINTAXE GERAL

LOMEM:exprnm

LOMEM determina o limite inferior de memória disponível para variáveis de programas BASIC.

A expressão LOMEM é usada nos microcomputadores que seguem a linha APPLE II para evitar que gráficos em alta resolução não interfiram com a área de dados de variáveis da memória. Não existe a mesma função no TRS-80 e usualmente não é necessária uma conversão.

Estabelece a menor posicão de memória RAM disponível para as linhas de programa e variáveis no BASIC, e usualmente o endereço de início da memória para a primeira variável do programa. Este comando tem em conta a proteção de variáveis do gráfico de alta resolução.

Quando o interpretador APPLESOFT é carregado do disquete ou cassete, ele sempre reside em RAM abaixo de LOMEM.

Cada vez que se adiciona uma linha de programa ou se modifica uma linha existente, LOMEM é ajustado para cima ou para baixo. Apagando-se um programa com NEW ou CTRL-B, também se altera LOMEM.

O valor da exprnm deve estar entre -65535 e +65535 (-32767 até +32767 na versão Integer BASIC) ou ocorrerá um erro.

Para ver o atual valor de LOMEM escreva:

PRINT PEEK(106)*256+PEEK(105)

Na versão BASIC APPLESOFT, se LOMEM: é colocado com um valor maior que HIMEM: menor que o valor já existente de LOMEM: ou menor que a posição mais alta de memória usada para o sistema operacional corrente ou programa , ocorre uma mensagem de erro.

VARIAÇÕES DE USO

Não se tem conhecimento de nenhuma.

VEJA TAMBÉM

HIMEM

Comando/Instrução **LPRINT**

SINTAXE GERAL

Número de linha LPRINT lista da expressões

LPRINT pode ser utilizada como COMANDO no modo direto ou como instrução no modo programa; tem como finalidade imprimir dados na impressora de linhas.

LPRINT supõe uma impressora com 132 caracteres de impressão.

LPRINT é utilizado principalmente nos microcomputadores que seguem a linha TRS-80 para desviar o display da tela para a impressora; nos microcomputadores que seguem a linha APPLE II deve-se usar PR#1.

LPRINT é utilizado por alguns BASIC da Microsoft para enviar uma instrução PRINT a uma impressora ao invés de encaminhá-la para a tela. LPRINT pode ser utilizado quer como comando direto quer como instrução de programa.

Todas as opções disponíveis à instrução PRINT estão também disponíveis à LPRINT, com a exceção de @. Estão incluídas TAB, impressão em zonas com vírgulas, impressão concentrada, com ponto e vírgula, e LPRINT USING.

Advertência: uma impressora tem que estar ligada com o computador e estar pronta para operar antes que se rode (RUN) um programa com utilização da instrução LPRINT.

```
EXEMPLO 1
100 REM USANDO LPRINT
110 LPRINT "VOCE TEM UM EXCELENTE MICRO"
120 LPRINT
130 X = 30
140 LPRINT TAB(X) ; X ; X+4 ; X+8
150 LPRINT "E UMA BOA IMPRESSORA"
160 LPRINT "TUDO ME PARECE ESTAR MUITO BOM,"
170 LPRINT USING "VOU LHE DAR $####,## POR ELA";999,50

EXECUÇÃO DO PROGRAMA
VOCE TEM UMA EXCELENTE IMPRESSORA !
                                30  34  38
TUDO ME PARECE ESTAR MUITO BOM,
VOU LHE DAR $999,50 POR ELA
```

VEJA TAMBÉM

PRINT, PRINT USING, TAB, ,(VIRGULA), ;(PONTO E VIRGULA), LLIST

Operador LT

SINTAXE GERAL

Número de linha IF expressão LT expressão... THEN...

LT é considerado operador condicional usado dentro das instruções IF...THEN...

LT é reconhecido em poucas versões BASIC e utilizado em alguns computadores como palavra alternativa pelo sinal de "menor do que" <.

Para mais informações veja <.

EXEMPLO 1

```
100 REM USANDO O OPERADOR LT (MENOR DO QUE)
110 IF 50 LT 100 THEN 140
120 PRINT "O OPERADOR LT NAO FOI ACEITO"
130 GOTO 150
140 PRINT "O OPERADOR LT FOI ACEITO"
150 END
```

EXECUÇÃO DO PROGRAMA

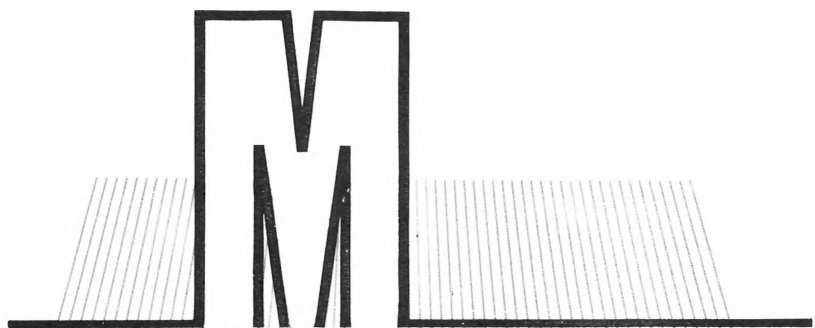
```
O OPERADOR LT FOI ACEITO
```

VARIAÇÕES DE USO

Não se tem conhecimento de nenhum.

VEJA TAMBÉM

>,<,<=,>=, =,<>,EQ, GE, GT, LE, NE, IF-THEN



MAN • MAT INPUT • MAT PRINT
MAT READ • MAT TRN
MAXFILES • MEM • M.
MID$ • MID • MIN • MOD • MON

Comando **MAN**

SINTAXE GERAL

MAN

MAN é um comando devendo ser utilizado no modo direto.
MAN é reconhecido na versão Integer BASIC dos computadores que seguem a linha APPLE para terminar a numeração automática de linhas iniciadas com o comando AUTO.
Procedimento para testar o comando MAN.
Se o computador estiver na versão Integer BASIC coloque no modo de numeração automática de linhas e pressione RETURN.
EXEMPLO : AUTO 100 (complete com a instrução e pressione RETURN)

```
> 100 PRINT "USANDO O COMANDO MAN"
> 110 PRINT "NA VERSAO INTEGER BASIC"
> 120 PRINT "DIGITE O COMANDO AUTO"
> 130 PRINT "COMPLETE AS LINHAS"
> 140 PRINT "E APERTE RETURN, CR OU EQUIVALENTE"
> 150 PRINT "PARA SAIR DA NUMERACAO AUTOMATICA"
> 160 PRINT "APERTE SIMULTANEAMENTE CTRL X"
> 170 PRINT "EM SEGUIDA DIGITE O COMANDO MAN"
```

VARIAÇÕES DE USO
Não se tem conhecimento de nenhum.

VEJA TAMBÉM
AUTO

Instrução **MAT INPUT**

SINTAXE GERAL

Número de linha MAT INPUT lista de variáveis

MAT INPUT é uma instrução reconhecida por poucas versões BASIC e usada geralmente no modo programa.

MAT INPUT atribui valores a um arranjo, onde as matrizes são lidas através do teclado. Uma instrução DIM estabelece o número de elementos da matriz. O número total de elementos de uma matriz redimensionada não pode ultrapassar o número total já estabelecido em uma instrução DIM.

Como uma declaração normal INPUT, o MAT INPUT imprime uma interrogação (?) no terminal como sinal de que está pronto a receber um valor para o primeiro elemento da matriz.

```
EXEMPLO 1
100 DIM A(2,4)
110 MAT INPUT A
120 END

EXECUÇÃO DO PROGRAMA
RUN
? 9,-4,3,5,5,3,2,0
```

Depois que a linha 110 for executada, a matriz A contém os seguintes valores.

9	-4	3	5
5	3	2	0

DIM estabelece o número de elementos da matriz que podem ter valores atribuídos à matriz.

Por exemplo:

```
100 DIM A(3)
120 MAT INPUT A
```

A instrução DIM permite à variável A utilizar os elementos de arranjo denominados A(0) até A(3). MAT INPUT atribui valores para as células A(1) até A(3).

Se tiverem que ser preenchidos numa única declaração todos os elementos, deve-se teclar uma vírgula depois de cada valor antes de pressionar a tecla RETURN ou ENTER. Se a cada elemento da matriz não se atribuiu um valor antes de pressionar a tecla RETURN ou ENTER, o computador irá imprimir um ponto de interrogação duplo (??), como sinal de que são necessários mais valores. Da mesma forma que ocorre com uma instrução INPUT comun, podem ser lançados valores um de cada vez, seguidos em cada caso pelo RETURN.

A instrução MAT INPUT atribui valores a cada coluna vertical

na primeira fileira de variáveis de duas dimensões antes de atribuir valores à fileira horizontal seguinte.

Por exemplo:

```
100 DIM A(2,3)
120 MAT INPUT A
```

O computador atribui valores aos elementos das variáveis de arranjo A(1,1), A(1,2) e A(1,3), antes de A(2,2) e A(2,3).

A maioria dos computadores que aceitam MAT INPUT permite estabelecer o tamanho do arranjo mediante a instrução MAT INPUT, contanto que não sejam utilizados mais de 10 elementos. Se o arranjo tiver que ter mais de dez elementos, deverá ser DIMensionado.

Por exemplo:

```
100 DIM B(20,20)
120 MAT INPUT A(3,5)
130 MAT INPUT B(15,11)
```

VARIAÇÕES DE USO

Não se tem conhecimento de nenhuma.

VEJA TAMBÉM

MAT PRINT, MAT READ, FOR, INPUT, DIM

Instrução **MAT PRINT**

SINTAXE GERAL

Número de linha MAT PRINT, nome da matriz

MAT PRINT é uma instrução reconhecida por poucas versões BASIC e usada geralmente no modo programa.

MAT PRINT é utilizado para imprimir os valores armazenados em elemento especificado de uma matriz (variável subscrita de duas dimensões). O número de elementos impressos é determinado pelo valor DIMensionado atribuído à matriz.

Por exemplo:

```
100 DIM A(3)
110 MAT PRINT A
```

Imprime os três valores atribuídos ao arranjo "A" (A(1) até A(3). Os dados que foram guardados na matriz podem ser colocados na declaração MAT PRINT.

```
EXEMPLO 1
100 DIM A(3,4)
110 MAT READ A
120 MAT PRINT A
130 DATA 1,2,3,4,5,6,7,8,9,10,11,12
140 END

EXECUÇÃO DO PROGRAMA
RUN
 1
 2
 3
 4
 5
 6
 7
 8
 9
10
11
12
```

Observe que cada elemento da matriz A foi impresso em linhas separadas, MAT PRINT somente dá saída a uma matriz para cada declaração. Parte da matriz pode ser impressa subscrevendo-se o nome da matriz. Por exemplo, depois da linha 110 a matriz foi colocada como segue:

1	2	3	4
5	6	7	8
9	10	11	12

Se quiser imprimir a parte superior esquerda da matriz, a declaração MAT PRINT deverá ser:

```
120 MAT PRINT A (2,3)
RUN
1
2
3
5
6
7
```

MAT PRINT não redefine a matriz, mas indica qual célula na porção superior esquerda da matriz deve ser impressa.

A maioria dos computadores com capacidade MAT PRINT permitem uma vírgula depois da instrução. Se o nome da matriz é seguido por uma vírgula, os elementos são impressos através da linha com um elemento em cada zona, com formato aberto.

Se o nome da matriz é seguido por um ponto-e-vírgula os elementos serão impressos numa forma compactada.

```
EXEMPLO 2
200 DIM A(3,4)
210 MAT READ A
220 MAT PRINT A,
230 MAT PRINT A;
240 DATA 1,2,3,4,5,6,7,8,9,10,11,12
250 END

EXECUÇÃO DO PROGRAMA
RUN
 1      2      3      4
 5      6      7      8
 9     10     11     12
MATRIZ GERADA POR MAT PRINT A,

 1  2  3  4
 5  6  7  8
 9 10 11 12
MATRIZ GERADA POR PRINT MAT A;
```

A instrução MAT PRINT pode imprimir o conteúdo de matrizes com mais de uma dimensão. O número de elementos na primeira dimensão especifica o número de fileiras a serem impressas, ao passo que o número de elementos na segunda coluna determina o número de colunas.

Por exemplo:

```
DIM A(2,3)
MAT PRINT
```

ORTOGRAFIA ALTERNATIVA

Alguns computadores permitem a impressão formatada de arranjos com a instrução MAT PRINT USING.

VARIAÇÕES DE USO

Não se tem conhecimento de nenhuma.

VEJA TAMBÉM

MAT INPUT, MAT READ, ,(VIRGULA), ;(PONTO-E-VIRGULA), FOR, PRINT, DIM, PRINT USING

Instrução **MAT READ**

SINTAXE GERAL

Número de linha MAT READ, lista variáveis

MAT READ é uma instrução reconhecida por poucas versões BASIC é usada geralmente no modo programa.

MAT READ é utilizada para ler valores de instrução DATA, atribuindo-os a uma matriz (variáveis subscritas de duas dimensões). A instrução DIM estabelece o tamanho da matriz. Mais que uma matriz pode ser lida e gerada numa declaração MAT READ, mas os nomes das matrizes devem ser separados por vírgulas.

MAT READ X(3,4),Y,Z

É feita a aquisição de 3 matrizes, X,Y e Z, sendo X dimensionada com 3 linhas e 4 colunas, se já não o foi no programa, ou redimensionada, se já foi dimensionada previamente.

MAT READ A(5),B

Faz a aquisicão de duas matrizes, A e B, sendo dimensionada a matriz A com cinco linhas e uma coluna, se não o foi anteriormente, e redimensionada, se já tiver sido dimensionada no programa.

EXEMPLO 1

```
100 DIM A(3,4),B(2,3)
110 MAT READ A,B
120 DATA 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18
130 END
```

EXECUÇÃO DO PROGRAMA

Na linha 110 a matriz A será preenchida primeiro e depois a matriz B. Então, após a execução da linha 110, as duas matrizes aparecerão como segue:

1	2	3	4
5	6	7	8
9	10	11	12

MATRIZ A

13	14	15
16	17	18

MATRIZ B

Se não houver suficientes dados para satisfazer ambas as matrizes A e B, uma mensagem de erro aparecerá no vídeo.

Matrizes podem ser redimensionadas quando nós carregamos os

valores da matriz.

```
EXEMPLO 2
200 DIM A(10,10)
210 MAT READ A(2,3)
220 DATA 10,20,30,40,50,60
230 END
```

Originalmente nós reservamos uma matriz 10x10 na memória. Quando carregamos valores para a matriz A, ela é redimensionada como uma matriz 2x3.

Especificamos a nova dimensão numa declaração MAT READ.

Depois que a linha 110 é executada, a matriz A tem a seguinte fórmula:

10	20	30
40	50	60

A instrução MAT READ atribui valores a cada coluna na primeira fileira de variáveis da matriz com duas dimensões, antes de atribuir valores à fileira seguinte:

Por exemplo:

```
100 DIM A(2,3)
110 MAT READ A
```

O computador lê seis valores da instrução DATA e os atribui a elementos de variáveis do arranjo A(1,1), A(1,2) e A(1,3) antes de tratar de A(2,1), A(2,2) e A(2,3).

A maioria dos computadores que aceita MAT READ permite que o tamanho da matriz seja estabelecido pela instrução MAT READ se não forem utilizados mais de 10 elementos. Havendo necessidade de mais de 10 elementos num arranjo, deverá ser DIMensionado.

```
EXEMPLO
110 DIM B(20,20)
120 MAT READ A(3,5)
130 MAT READ B(15,11)
```

Para fazer o teste desta característica no seu computador, omita linha 110 no exemplo 1 e mude a linha 120

```
120 MAT READ A(3,4)
```

Se o seu computador aceitar essa característica, a execução do programa não deve mudar.

VARIAÇÕES DE USO

Não se tem conhecimento de nenhuma.

VEJA TAMBÉM

MAT PRINT, MAT INPUT, READ, DATA, DIM, FOR

Instrução **MAT TRN**

SINTAXE GERAL

Número de linha MAT, nome matriz = 1 TRN (nome matriz 2)

MAT TRN é uma instrução reconhecida por poucos interpretadores BASIC.

MAT TRN troca linhas por colunas de uma matriz (com duas dimensões).

Sendo A um arranjo de 3x4 e B um arranjo de 4x3, MAT B = TRN(A) irá formar a matriz B de tal maneira que a primeira fileira da matriz A se torne a primeira coluna da matriz B, a segunda fileira de A se torne a segunda coluna de B, e assim sucessivamente.

EXEMPLO 1

```
        1   2   3   4
M =     5   6   7   8
        9  10  11  12
```

100 MAT N = TRN(M)
causara:

```
        1   5   9
        2   6  10
N =     3   7  11
        4   8  12
```

Onde M é uma matriz 3x4 e N uma matriz 4x3.

A instrução MAT M = TRN(M) será legal somente se ambas as dimensões de M forem iguais (isto é, se M for uma matriz quadrada)

VARIAÇÕES DE USO

Alguns computadores permitem que a palavra MAT seja facultativa.

VEJA TAMBÉM

MAT, DIM, FOR, NEXT

Comando/DOS **MAXFILES**

SINTAXE GERAL

MAXFILES N

O comando MAXFILES especifica o número máximo de arquivos permitidos que podem ser ativados a qualquer tempo.

N deve ser uma constante na faixa de um(1) a 16 ou um erro aparecerá.

É um comando DOS e requer PRINT e CHR$(4) no modo programado.

O sistema operacional DOS suporta no máximo 16 arquivos abertos ao mesmo tempo.

Quando executado, o MAXFILES reserva 595 bytes de memória (um buffer de arquivo) para cada arquivo. Quando se carrega o DOS, faz automaticamente MAXFILES 3, de tal modo que três arquivos, no máximo, poderão estar ativos ao mesmo tempo, a menos que um outro MAXFILES explicitado seja executado. Contudo, na maioria dos casos, três arquivos são suficientes.

Todos os comandos do DOS exceto o MAXFILES usam o buffer de arquivo quando são executados. Assim, o número máximo de arquivos que se pode ter aberto na prática é um a menos do limite de MAXFILES.

Quando se buscam dados armazenados no disco, o DOS traz 256 bytes, e os coloca na parte de entrada do "buffer". Quando um programa se propõe escrever dados no disquete, os caracteres são armazenados na parte de saída do "buffer" de arquivo até que 256 bytes fiquem acumulados, sendo então enviados para o disquete de uma vez.

Quando três arquivos permanecem ativos ao mesmo tempo, significa que 1785 bytes de memória são reservados para três "buffer" de arquivo.

MAXFILES N é usado geralmente no modo direto, mas se for necessário usar MAXFILES dentro de um programa APPLESOFT, ele deve ser usado na primeira linha (0), para que seja o primeiro comando a ser executado.

EXEMPLO:

MAXFILES 5

Para usar MAXFILES em um programa Integer BASIC, deve-se criar um arquivo EXEC.

VARIAÇÕES DE USO

Não se tem conhecimento de nenhuma.

VEJA TAMBÉM

DIM, HIMEM

Função **MEM • M.**

SINTAXE GERAL

Número de linha LET variável, função MEM

MEM é uma função podendo ser utilizada no modo direto ou no modo programa devendo ser precedida de uma instrução como LET ou PRINT.

1- Esta função é reconhecida nos microcomputadores que seguem a linha TRS-80.

2- A função MEM fornece o número total de bytes desprotegidos e não utilizados da memória. Não inclui o espaço utilizado para armazenagem de strings.

MEM é utilizada geralmente no nível de comando, com um comando PRINT para exibir a quantidade de bytes ainda não utilizados de MEMória que ainda permanecem no computador. MEM pode também ser utilizada numa instrução de programa, para se evitar erros de OM (Memória Totalmente Ocupada), através da colocação de menos espaços strings, ou de dimensionamento de menores matrizes etc.

MEM não necessita de nenhum argumento.

EX. 10 IF MEM <80 THEN 90

A instrução da função MEM precedida da instrução PRINT no modo programa é utilizada para se descobrir a quantidade de memória não-utilizada para armazenagem, programas, variáveis, strings, stack, ou sendo reservada para arquivos objetos.

EXEMPLO 1

```
100 REM USANDO MEM
110 PRINT MEM; "BYTES DE MEMORIA AINDA RESTAM"
120 END
```

EXECUÇÃO DO PROGRAMA (tipica)

```
RUN
48035 BYTES DE MEMORIA AINDA RESTAM
```

(As quantidades de memória disponível dependerão evidentemente do tamanho da memória do seu computador.)

ORTOGRAFIA ALTERNATIVA

M. é utilizado no TRS-80 Nível I como abreviatura de MEM.

FREE(string) ou FREE(número) devolve a quantidade de memória livre. (O mesmo que MEM.)

VARIAÇÕES DE USO

Não se tem conhecimento de nenhuma.

VEJA TAMBÉM

FRE, CLEAR

Função **MID$ • MID**

SINTAXE GERAL

Número de linha LET variável = MID$ (X$,P,N)
PRINT MID$ ou (X$,P,N)

MID$ é uma função usada geralmente no modo programa devendo ser precedida de uma instrução como LET ou PRINT.

MID$("...",P,N) ou

MID$(X$,P,N), função aceita nos microcomputadores das linhas APPLE e TRS-80.

Em ambos os casos a função provocará a leitura da string começando com o carácter na posição P, passando a ler os N(números de) caracteres indicados.

Esta função inexiste na versão BASIC para TRS-80 Nível 1 ou na versão INTEGER BASIC da linha APPLE.

OBSERVAÇÃO:

A variável ou expressão alfanumérica(X$) não deve exceder a 255 caracteres e os argumentos numéricos PEN devem estar entre 1 e 255 (inteiros).

A função MID$ (string,P,N) é utilizada para isolar um número específico de caracteres de strings, que distam P (uma posição) caracteres do carácter mais para a esquerda no string.

PRINT MID$ (COMPUTADOR,4,3), por exemplo, imprimirá as letras 'PUT', que são os três caracteres no MIDdle (meio) a começar do quarto carácter de string a partir da esquerda.

O string deve estar entre aspas ou ser atribuído a uma variável de string. O número de caracteres e a posição inicial podem ser expressos como variáveis, números ou operações aritméticas. Cada argumento na função MID$ deve estar separado por uma vírgula. Se o valor de P ou N for decimal, o computador converte o mesmo automaticamente para número inteiro.

```
EXEMPLO 1
100 REM USANDO A FUNCAO MID$
110 A$ = "REAPROVADA"
120 B$ = MID$(A$,3,8)
130 PRINT MID$("MAIS",2,1)" FUNCAO MID$ FOI ";B$
140 END

EXECUÇÃO DO PROGRAMA
RUN
A FUNCAO MID$ FOI APROVADA
```

Sendo omitida a extensão (N), a maioria das versões BASICS irá simplesmente isolar o resto do string. Existem algumas, no entanto (tais como BASIC TDL), que isolam apenas o carácter na respectiva posição especificada. Outras não permitem omitir valores.

```
EXEMPLO 2
200 REM USANDO MID$(A$,N)
210 PRINT MID$("SULLIVAN",5)
250 END
```

Será que imprimiu IVAN, ou uma mensagem de erro?

MID$ é colocado do lado esquerdo de alguns interpretadores, (tais como BASIC-80, BASIC TDL etc.) para modificar o conteúdo de uma variável de string.

120 MID$(A$,3,5)=B$, por exemplo, substitui cinco caracteres de A$ pelos primeiros cinco caracteres de B$, começando pela terceira posição a partir do lado esquerdo de B$. Se B$ não contiver cinco caracteres, serão inseridos espaços em branco em A$. Se a extensão for omitida (5, no nosso exemplo), todo o resto de A$ será substituído por B$.

Por exemplo, o MID$(X$,3,1) se refere a um string de um carácter que inicia pelo terceiro carácter de L$.

Se nao for determinado argumento, o string que inicia na posição P será selecionado no exemplo abaixo. Os três primeiros dígitos de um número telefônico local são chamados prefixos da central.

```
EXEMPLO 3
100 REM USANDO MID$
110 INPUT "CODIGO DE AREA E NUMERO ";TEL$
120 PRE$ = MID$(TEL$,4,3)
130 PRINT "O NUMERO E ";", PREFIXO DO NUMERO CHAMADO"
140 END
```

EXECUÇÃO DO PROGRAMA

Este programa examina um número telefônico completo (código de área e os últimos dígitos) e seleciona o prefixo deste número:

Ex: DDD(discagem direta a distância)
021 541-3933

ORTOGRAFIA ALTERNATIVA

Alguns interpretadores (tais como a BASIC-PLUS da DEC e a BASIC-V da Harris) aceitam MID no lugar de MID$ para isolar substrings.

SE O SEU COMPUTADOR NÃO TIVER ESTA FUNÇÃO

A maioria dos computadores dispõe de meios de isolar substrings se não tiverem a função MID$. Alguns utilizam SEG4, ao passo que SUBSTR é empregado em outros.

Os computadores que requerem o dimensionamento das variáveis de string (tais como North Start, Hewlett Packard, etc) utilizam subscritos para isolar os caracteres do string. Por exemplo:

```
100 DIM A$(8)
110 A$ = "ABCDEFGHIJKLMNOP"
120 PRINT A$(5,5)
130 END
Imprime EFGHI
```

Na versão BASIC os micros que seguem a linha do Sinclair requerem que a linha 120 seja:

```
120 PRINT A$(5 TO 5)
```

VARIAÇÕES DE USO

Não se tem conhecimento de nenhuma.

VEJA TAMBÉM

PRINT, RIGHT$, LEFT$, CHR$, SPACE$, STR$, STRING$, INKEY$

Função **MIN**

SINTAXE GERAL

Número de linha LET variável = variável função MIN

MIN é uma função usada geralmente no modo programa devendo ser precedida de uma instrução como LET ou PRINT.

Esta função é reconhecida por apenas algumas versões BASIC.

A função MIN é utilizada para determinar qual de dois valores é o maior.

Y = A MIN 8, por exemplo, atribui a Y o valor de A se A for menor do que 8. Caso contrário, Y = 8.

```
EXEMPLO 1
100 REM USANDO MIN
110 A = 5
120 Y = A MIN 8
130 IF Y = 5 THEN 160
140 PRINT "MIN NAO FOI APROVADO"
150 GOTO 170
160 PRINT "MIN FOI APROVADO"
170 END
```

EXECUÇÃO DO PROGRAMA

```
RUN
MIN FOI APROVADO
```

VARIAÇÕES DE USO

Alguns computadores utilizam a forma MIN(A,B) para o mesmo fim. Alguns são capazes de achar o menor valor numa matriz utilizando MIN.

SE O SEU COMPUTADOR NÃO TIVER ESSA FUNÇÃO

O valor MINimo (Y) de dois números pode ser determinado por esta fórmula:

```
Y = (A+B-ABS(A-B))/2
```

ou por

```
120 Y = A
122 B = Y
124 IF A <= B THEN 130
```

O valor MINimo duma matriz pode ser determinado pelo programa seguinte:

```
EXEMPLO 2
100 DIM B(6)
110 FOR X = 1 TO 6
120 READ B(X)
130 NEXT X
140 N = B(1)
150 FOR X = 2 TO 6
160 IF N <= B(X) THEN 180
170 N = B(X)
180 NEXT X
```

```
190 PRINT "O VALOR MINIMO E ";N
200 DATA 2,6,15,1,9,-7
210 END
```

EXECUÇÃO DO PROGRAMA

```
RUN
O VALOR MINIMO E -7
```

VEJA TAMBÉM

ABS, <=

Função **MOD**

SINTAXE GERAL

Número de linha LET variável = Y MOD X

MOD é uma função usada geralmente no modo programa devendo ser precedida de uma instrução como LET ou PRINT.

Esta função é reconhecida por apenas algumas versões BASIC.

A MOD B é utilizada em alguns microcomputadores para calcular o restante aritmético (MODulo) depois que o valor de A for dividido pelo valor de B.

PRINT 10 MOD 6, por exemplo, imprime o número 4, que é o remanescente ao se dividir 10 por 6.

Alguns computadores convertem automaticamente em número inteiro o valor do MODulo.

PRINT 12,5 MOD 5, por exemplo, talvez imprima o número 2 (o valor de número inteiro do remanescente de 2,5).

EXEMPLO 1

```
100 REM USANDO MOD
110 N = 12 MOD 5
120 IF N = 2 THEN 150
130 PRINT "A FUNCAO MOD NAO FOI ACEITA"
140 GOTO 160
150 PRINT "A FUNCAO MOD FOI ACEITA"
160 END
```

EXECUÇÃO DO PROGRAMA

```
RUN
A FUNÇÃO MOD FOI ACEITA
```

SE O SEU COMPUTADOR NÃO TIVER ESSA FUNÇÃO

MOD é cômodo, mas não insubstituível. Aqui está, apresentada passo por passo, uma maneira de contornar

```
110 N = 12/5
112 N = N-INT(N)
114 N = INT(N*5)
```

Uma forma mais generalizada da equação vem a ser:

```
110 N = INT(A-B*INT(A/B))
```

Substitua 12 por A e 5 por B e experimente com o exemplo.

VARIAÇÕES DE USO

Alguns computadores (tais como Harris BASIC-V) utilizam MOD(X,Y) para calcular o valor de X MODulo Y.

VEJA TAMBÉM

INT, FIX

Comando/DOS **MON**

SINTAXE GERAL

MON [C] [,I] [,O]

MON C,I,O é um comando do DOS 3.3 (Sistema Operacional do Disquete) utilizado nos microcomputadores que seguem a linha APPLE II. O comando MON faz visíveis no monitor de vídeo os comandos do DOS e/ ou fluxos de dados. É um comando DOS, e requer PRINT e CHR$(4) quando usado em modo programado. Os comandos do DOS e as informações trocadas entre o computador e o disquete normalmente ficam invisíveis na tela de vídeo. Para monitorizar esta ação use o comando MON.

MON [C] [,I] [,O]

Os três parâmetros dizem o que será mostrado na tela de vídeo.

C mostra todos os comandos do DOS.

I mostra quantidade de dados que entra, isto é, informação que é enviada do disquete para o computador.

O mostra saída de dados, isto é, informação que é enviada do computador para o disquete.

Os argumentos (razão) são:

C para comandos

I para input

O para output

Os parâmetros C,I,O podem aparecer em qualquer ordem e em qualquer combinação, dependendo da informação que se deseje no monitor. As vírgulas são opcionais, por exemplo: MONICO

Pelo menos um dos segmentos deve estar presente ou então o comando MON será ignorado. O comando MON permite a monitorização de várias informações, e seu efeito permanece até que seja executada uma declaração NOMON, FP ou INT. O comando MON é útil na depuração de programas.

EXEMPLO 1

```
100 REM RELACAO DE NOMES
110 D$ = CHR$(4)
120 PRINT D$; "MON,C,I,O"
130 PRINT D$;"OPEN NOMES"
140 PRINT "ISTO VAI PARA O VIDEO"
150 PRINT D$;"WRITE NOMES"
160 PRINT "CLOVIS"
170 PRINT "ROSSANA"
180 PRINT "SUSY"
190 PRINT "SIMONE"
200 PRINT D$;"CLOSE NOMES"
210 PRINT D$;"NOMON C,I,O"
220 END
```

VARIAÇÕES DE USO

Não se tem conhecimento de nenhuma.

VEJA TAMBÉM

NOMON

NE • NEM • N
NEXT • NEX • N. • NOFLOW
NOMON • NORMAL • NOT
NOTRACE • NUM • NUM$

Comando/Operador **NE**

SINTAXE GERAL

NE

NE é utilizado geralmente no modo direto em alguns computadores (tais como o TL990) como abreviatura do comando NEW e do operador relacional "não-igual a". É reconhecido como NEW ao ser utilizado no modo comando, e como <> ao ser utilizado como instrução de programa.

EXEMPLO 1

```
100 REM USANDO NE(NEW) COMO COMANDO
110 PRINT "NE OU NEW APAGA O PROGRAMA DA MEMORIA RAM"
120 END
```

EXECUÇÃO DO PROGRAMA

LISTe o programa para ter certeza de que se deu entrada no mesmo. Tecle NE para apagar o exemplo 1 e tecle novamente, LIST, para ter certeza de que o programa foi "apagado".

EXEMPLO 2

```
200 REM USANDO NE COMO OPERADOR RELACIONAL
210 X = 15
220 IF X NE 25 THEN 250
230 PRINT "O OPERADOR NE NAO FOI APROVADO"
240 GOTO 260
250 PRINT "O OPERADOR NE FOI APROVADO"
260 END
```

EXECUÇÃO DO PROGRAMA

```
RUN
O OPERADOR NE FOI APROVADO
```

VARIAÇÕES DE USO

Não se tem conhecimento de nenhuma.

VEJA TAMBÉM

NEW, <>, <, >, <=, >=, =, EQ, GE, GT, LE, LT, IF-THEN

Comando/Instrução **NEW • NE • N**

SINTAXE GERAL

NEW ou número de linha NEW.

NEW é utilizado como comando no modo direto e não tem aplicação como instrução no modo programa, a não ser para oferecer proteção com senha.

O comando NEW apaga o(s) programa(s) BASIC armazenado(s) na memória. Não apaga, contudo, o interpretador em si. NEW é utilizado normalmente ao se ter de dar entrada num programa novo ao computador, apagando-se o programa existente.

Este comando apaga qualquer programa que exista na memória, apaga suas linhas, zera as variáveis numéricas e anula as variáveis alfanuméricas (strings). Entretanto, ele não modifica os espaços strings colocados por uma instrução CLEAR, nas versões BASIC para TRS-80.

Nos microcomputadores que seguem a linha APPLE, o comando NEW também reajusta o LOMEM:, porém HIMEM:COLOR e HCOLOR não são afetados. E na versão INTEGER BASIC o NEW só pode ser usado no modo direto.

EXEMPLO 1

```
100 REM USANDO NEW
110 PRINT "NEW IRA APAGAR ESTE PROGRAMA NA MEMORIA RAM"
120 END
```

EXECUÇÃO DO PROGRAMA

LISTe o programa para ter certeza de que se lhe deu entrada.

Tecle NEW para apagar o exemplo, e tecle novamente LIST para ter certeza de que o programa foi apagado.

Alguns computadores talvez utilizem SCRATCH ou SCR em vez de NEW.

ORTOGRAFIA ALTERNATIVA

Alguns computadores aceitam NE ou N como abreviaturas de NEW.

VARIAÇÕES DE USO

Existem certos microcomputadores (tais como os da linha Sinclair) que utilizam NEW n para apagar o programa e ao mesmo tempo estabelecem a quantidade de memória disponível para BASIC. Caso deva estar presente na memória um programa em linguagem de máquina juntamente com um programa em BASIC, é de boa praxe (ou até necessário) reservar a parte de cima da memória para o programa em linguagem de máquina; "n" é número decimal.

VEJA TAMBÉM

CLEAR, SCRATCH

Instrução **NEXT • NEX • N.**

SINTAXE GERAL

Número de linha NEXT variável contadora
ou
Número de linha NEXT

NEXT é instrução usada geralmente no modo programa, como par do FOR...TO...STEP...

A instrução NEXT termina o LOOP automático iniciado com a instrução FOR. Quando a instrução NEXT é executada , a variável (contadora) do LOOP é incrementada do valor especificado na declaração STEP do FOR. (FOR variável = valor inicial TO valor final step incremento).

Se o valor incrementado com o step é menor que o valor final, o programa continua executando a instrução que segue ao FOR; se o valor incrementado é maior que o valor final do LOOP, a execução será desviada para a próxima instrução após o NEXT.

NEXT não pode ser utilizado sem o FOR, causará uma mensagem de erro.

POR EXEMPLO

```
100 FOR X = 1 TO 5 STEP 1
110 NEXT X
120 END
```

Na quinta vez que se execute a instrução NEXT, o valor de X terá sido incrementado para 6, que ultrapassa o alcance da instrução FOR, de 5, fazendo com que o computador "caia para frente", para a linha 120.

Se executar no modo direto PRINT X
Isto poderá ser confirmado

EXEMPLO 1

```
100 REM USANDO NEXT
110 FOR R = 1 TO 3
120 PRINT R,
130 NEXT R
140 PRINT
150 PRINT "A INSTRUCAO NEXT FOI APROVADA"
160 END
```

EXECUÇÃO DO PROGRAMA

```
RUN
 1      2      3
A INSTRUÇÃO NEXT FOI APROVADA
```

Uma vez que a instrução NEXT volta tão-somente à instrução FOR anterior, que utiliza a mesma variável, é possível, no caso da maioria dos computadores, fazer uso de instruções FOR-NEXT "aninhados".

EXEMPLO 2

```
200 REM USANDO INSTRUCOES NEXT ANINHADOS
```

```
210 FOR X = 1 TO 3
220 PRINT "X=";X
230 FOR Y = 1 TO 2
240 PRINT "Y=";Y
250 NEXT Y
260 NEXT X
270 PRINT "O PROGRAMA NEXT FOI APROVADO"
280 END

EXECUÇÃO DO PROGRAMA
RUN
X = 1
Y = 1
Y = 2
X = 2
Y = 1
Y = 2
X = 3
Y = 1
Y = 2
O PROGRAMA NEXT FOI APROVADO
```

Muitos computadores permitem a execução de uma instrução NEXT que não contém variável. Nesse caso, o computador retorna a instrução FOR anterior (seja qual for a variável associada) enquanto não tenha ultrapassado seu alcance determinado.

Em algumas versões BASIC para TRS-80 ou a versão BASIC APPLESOFT pode-se usar NEXT sem identificação do nome da variável contadora.

A variável adotada por ausência é aquela do FOR que iniciou mais recentemente e que ainda está em efeito.

NEXT sem variável é executado mais rapidamente que o correspondente com variável.

Para testar essa característica, rode o segundo exemplo depois de eliminar as variáveis X e Y das instruções NEXT nas linhas 150 e 160. A rodada de amostra deve permanecer a mesma.

Alguns computadores permitem que NEXT especifique mais de uma variável. Para se terminar uma alçada aninhada tríplice, por exemplo, pode ser utilizado NEXT K, J, I (com as variáveis na ordem inversa das instruções FOR correspondentes).

ORTOGRAFIA ALTERNATIVA

Alguns computadores permitem utilizar NEX e N. como abreviaturas de NEXT.

VARIAÇÕES DE USO

Alguns computadores (tais como DEC BASIC-PLUS-2) permitem que NEXT seja implícito sob determinadas circunstâncias. Escreve-se FOR, mas não NEXT.

Exemplo: 130 PRINT X,X*X FOR X=1 TO 5

Na versão INTEGER BASIC para os microcomputadores que seguem a linha APPLE, o NEXT não pode ser utilizado no modo direto.

VEJA TAMBÉM

FOR

Comando/Instrução **NOFLOW**

SINTAXE GERAL

Número de linha, comando NOFLOW

NOFLOW é utilizado como comando no modo direto ou como instrução no modo indireto. Porém não tem aplicação como instrução no modo programa.

O comando NOFLOW é reconhecido e utilizado por apenas algumas versões BASIC tais como BASIC Micropolis para anular a função "trace" (rastrear).

NOFLOW pode ser utilizado como instrução de programa para desligar a operação "trace" em determinadas áreas especificadas dentro do programa.

EXEMPLO 1

```
100 REM USANDO NOFLOW
110 PRINT "AS PRIMEIRAS TRES LINHAS DESTE PROGRAMA"
120 NOFLOW
130 PRINT "SAO IMPRESSAS COM O RASTREAMENTO LIGADO,"
140 PRINT "ESTA LINHA E IMPRESSA COM O RASTREAMENTO DESLIGADO"
150 END
```

EXECUÇÃO DO PROGRAMA

RUN o exemplo 1 depois de teclar o comando FLOW

```
<100> <110> AS PRIMEIRAS TRES LINHAS DESTE PROGRAMA
<120> SAO IMPRESSAS COM O RASTREAMENTO LIGADO
ESTA LINHA E IMPRESSA COM O RASTREAMENTO DESLIGADO
```

VARIAÇÕES DE USO

Não se tem conhecimento de nenhuma

VEJA TAMBÉM

FLOW, NOTRACE, TRACE OFF, TROFF

Comando/DOS **NOMON**

SINTAXE GERAL

NOMON [C] [,I] [,O]

NOMON C,I,O é um comando do DOS 3.3 para computadores que seguem a linha APPLE. Este comando quando executado desativa em parte ou totalmente o comando MON, retornando o sistema para o estado "normal" (isto é, ficando novamente invisíveis no monitor de vídeo as trocas de informações entre o computador e a drive).

NOMON finaliza a apresentação de comandos de disco e fluxo de dados iniciada em MON.

O comando NOMON desliga a mostra de comandos do disco e informações enviadas entre o computador e o disco. É um comando DOS, e requer PRINT e CHR$(4) quando em modo programado.

NOMON [C] [,I] [,O]

onde

C suprime a mostra dos comandos do disquete.

I suprime a mostra das informações de entrada que vão do disquete para o computador.

O suprime a mostra de saída das informações indo do computador para o disquete.

Os argumentos são: C para comando, I para INPUT e O para OUTPUT. Eles podem aparecer em qualquer ordem, ou em qualquer combinação, as vírgulas são opcionais.

EXEMPLO 1

```
100 REM RELACAO DE NOMES
110 D$ = CHR#(4)
120 PRINT D$; "MON C,I,O"
130 PRINT D$; "OPEN NOMES"
140 PRINT "ISTO VAI PARA O VIDEO"
150 PRINT D$; "WRITE NOMES"
160 PRINT "CLOVIS"
170 PRINT "ROSSANA"
180 PRINT "SUSY"
190 PRINT "SIMONE"
200 PRINT D$; "CLOSE NOMES"
210 PRINT D$; "NOMON C,I,O"
220 END
```

OBS:Pelo menos um dos argumentos deve estar presente ou NOMON é ignorado. O comando NOMON pode ser emitido de uma maneira tal que ele imprima quase invisivelmente, para isso use:

100 PRINT D$; "NOMON C,I,O": VTAB PEEK(37): CALL-868

onde D$ contém (CHR$(4)), VTAB PEEK(37) move o cursor para o começo da linha que contém NOMON C,I,O e CALL-868 limpa a linha.

VARIAÇÕES DE USO

O comando NOMON C,I,O pode ser utilizado tanto no modo direto quanto no modo programa.

VEJA TAMBÉM

MON

Instrução/Comando **NORMAL**

SINTAXE GERAL

NORMAL
Número de linha, comando NORMAL

NORMAL pode ser utilizado como comando no modo direto ou como instrução no modo programa, nos microcomputadores que seguem a linha APPLE na versão BASIC APPLESOFT para desligar os modos de vídeo FLASH e INVERSE.

NORMAL faz com que toda saída do microcomputador seja exibida no vídeo sob forma de caracteres brancos num fundo preto ou verde.

NORMAL deve ser utilizado depois das instruções INVERSE ou FLASH, ambas as quais criam no vídeo especiais efeitos visuais.

EXEMPLO 1

```
100 REM USANDO O COMANDO NORMAL
110  INVERSE
120  PRINT "ESTA E UMA IMPRESSAO NO MODO INVERSO"
130  NORMAL
140  PRINT "DE VOLTA AO NORMAL"
150  END
```

Para rodar esse programa, limpe a tela e tecle RUN.

EXECUÇÃO DO PROGRAMA

```
RUN
ESTA E UMA IMPRESSAO NO MODO INVERSO
DE VOLTA AO NORMAL
```

ORTOGRAFIA ALTERNATIVA

Não existe até o momento nenhuma função equivalente para os micros que seguem a linha TRS-80.

NORMAL não é disponível na versão Integer Basic da linha APPLE.

VARIAÇÕES DE USO

Não se tem conhecimento de nenhuma

VEJA TAMBÉM

INVERSE, FLASH

Operador **NOT**

SINTAXE GERAL

Número de linha IF NOT variável condição THEN...instrução.

NOT é utilizado em instruções IF-THEN como operador lógico que inverte o valor de uma variável ou expressão lógica.

NOT (assim como AND e OR) é usado para manipulações de bits e operações booleanas. Convertem seus argumentos em números inteiros (com sinal) em complemento de dois e 16 bits. Isto dá uma faixa de -32768 até +32767.

Após esta conversão é feita a operação lógica e apresentado um número na mesma faixa.

Estas operações são realizadas bit-a-bit.

EX:

NOT 0 = 1

A negação de 0 em 16 bits é 1111111111111111, que é -1

NOT -1 = 0

-1 é 1111111111111111 que negado resulta em 0.

IF NOT (X>10) THEN 150, por exemplo, significa "se o valor armazenado em X não for maior do que 10, GOTO linha 150

EXEMPLO 1

```
100 REM USANDO NOT LOGICO
110 X = 5
120 IF NOT (X>10) THEN 150
130 PRINT "NOT FALHOU NO TESTE DE OPERADOR LOGICO"
140 GOTO 160
150 PRINT "NOT PASSOU NO TESTE DE OPERADOR LOGICO"
160 END
```

EXECUÇÃO DO PROGRAMA

```
RUN
NOT PASSOU NO TESTE DE OPERADOR LOGICO
```

Alguns computadores utilizam NOT em comparações de strings.

IF NOT(X$=SIM OR X$=NAO) THEN 150 significa que se o string armazenado em X$ não for nem SIM nem NAO, o controle do programa passa para a linha 150.

EXEMPLO 2

```
100 USANDO NOT COMO STRING LOGICO
110 PRINT "TECLAR UM SIM OU UM NAO";
120 INPUT X$
130 IF NOT (X$="SIM" OR X$="NAO") THEN 160
140 PRINT "OBRIGADO"
150 GOTO 180
160 PRINT X$;" NAO E NEM SIM NEM NAO"
170 GOTO 110
180 END
```

EXECUÇÃO DO PROGRAMA

```
RUN
```

```
TECLAR UM SIM OU UM NAO? BEM
BEM NAO E NEM SIM NEM NAO
TECLAR UM SIM OU UM NAO? NAO
OBRIGADO
```

O operador NOT é utilizado por alguns computadores para formar o complemento binário (complemento de 1s) de um número (isto é, cada bit na representação binária do número é mudado).

Todos os 0(zeros) se convertem em 1(um) e todos os 1(um) se convertem em 0(zero).

```
EXEMPLO 3
100 REM USANDO NOT COMO COMPLEMENTO
110 PRINT "DE ENTRADA NUM NUMERO ENTRE -76532 E 76532";
120 INPUT X
130 Y = NOT(X)
140 PRINT "O COMPLEMENTO BINARIO DE";X;" E ";Y
150 GOTO 110
160 END
```

```
EXECUÇÃO DO PROGRAMA (utilizando-se 7)
RUN
DE ENTRADA NUM NUMERO ENTRE -76532 E 76532? 7
O COMPLEMENTO BINARIO DE 7 E -8
DE ENTRADA NUM NUMERO ENTRE -76532 E 76532?
```

SE O SEU COMPUTADOR NÃO TIVER ESSE OPERADOR

Instruções equivalentes podem ser elaboradas sem o uso de NOT.

NOT (X$=SIM OR X$=NAO) é a mesma coisa que X$<>SIM AND X$<>NAO

Para formar o complemento binário de um número utilize Y=-(X%1)

VARIAÇÕES DE USO

Não se tem conhecimento de nenhuma.

VEJA TAMBÉM

AND, OR, XOR, IF-THEN

Comando/Instrução **NOTRACE**

SINTAXE GERAL

Número de linha NOTRACE

NOTRACE pode ser utilizado como comando no modo direto ou como instrução no modo programa.

O comando NOTRACE é utilizado nos micros que seguem a linha APPLE II para cessar a função de restreamento iniciado com o comando ou instrução TRACE.

NOTRACE desativa a linha-número em que reside uma função TRACE nos micros da linha APPLE.

NOTRACE pode ser utilizado como instrução de programa para desligar o rastreamento em determinadas áreas de um programa

EXEMPLO 1

```
100 REM USANDO NOTRACE
110 TRACE
120 PRINT" CADA LINHA DEVE SER RASTREADA"
130 NOTRACE
140 PRINT "PELA INSTRUCAO TRACE"
150 PRINT "ATE ESTA SER DESLIGADA  PELA INSTRUCAO NOTRACE"
160 END
```

EXECUÇÃO DO PROGRAMA

```
RUN
#120 CADA LINHA DEVE SER RASTREADA
#130 PELA INSTRUCAO TRACE
ATE ESTA SER DESLIGADA PELA INSTRUCAO NOTRACE
```

ORTOGRAFIA ALTERNATIVA

Nos microcomputadores que seguem a linha TRS-80 usa-se TROFF para funções idênticas ao NOTRACE.

VARIAÇÕES DE USO

Não se tem conhecimento de nenhuma

VEJA TAMBÉM

TRACE, TRACE OFF, TROFF, NOFLOW

Função **NUM$**

SINTAXE GERAL

Número de linha LET variável = NUM$ número

NUM$ é uma função reconhecida por apenas algumas versões BASIC e funciona semelhante a STR$, que cria um string de números a partir de uma expressão numérica NUM$ (47.7), por exemplo, converte o valor 47.7 no string de caracteres "47.7". A diferença reside no fato de que operações aritméticas podem ser realizadas com a forma numérica ao passo que na forma de string podem ser aplicadas apenas funções de string.

EXEMPLO 1

```
100 REM USANDO NUM$
110 B = 789645
120 B$ = NUM$(B)
130 PRINT "SE O NUMERO ";B;" FOR CONVERTIDO AO STRING ";B$
140 PRINT "SIGNIFICA QUE A FUNCAO NUM$ FOI APROVADA"
150 END
```

EXECUÇÃO DO PROGRAMA

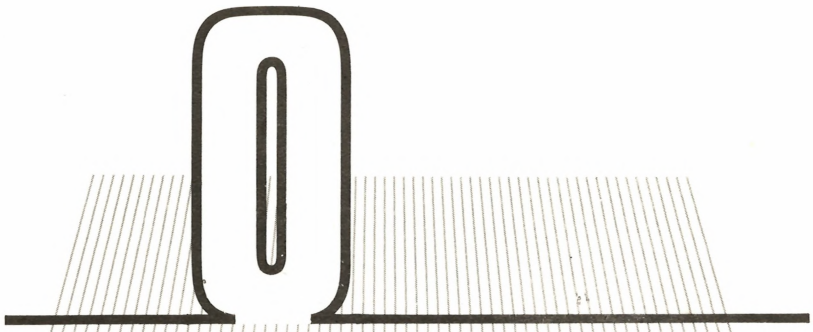
```
RUN
SE O NUMERO 789645 FOR CONVERTIDO AO STRING 789645
SIGNIFICA QUE A FUNCAO NUM$ FOI APROVADA
```

VARIAÇÕES DE USO

Não se tem conhecimento de nenhuma.

VEJA TAMBÉM

STR$, VAL, ASC, CHR$



ON ERROR GOTO • ON ERR GOTO
ON-GOSUB • ON-GOTO
ONNER GOTO • OPEN • OPTION
OR • OUT

Instrução **ON ERROR GOTO**
ON ERR GOTO

SINTAXE GERAL

Número de linha ON ERROR GOTO número de linha

ON ERR GOTO para microcomputadores que seguem a linha APPLE ou ON ERROR GOTO para os que seguem a linha TRS-80 têm funções idênticas a despeito da grafia diferente. Normalmente, se um erro acontece durante a execução de um programa, a execução do programa é interrompida.

Esta instrução pode ser incluída em um programa se um erro ocorrer quando da execução do programa.

Você pode criar uma rotina que evita a interrupção do programa.

OBS: A rotina que manipula o erro deverá terminar com a instrução RESUME.

A instrução ON ERROR GOTO é utilizada como desvio para uma sub-rotina de erro, ao se encontrar um erro no programa. A instrução ON ERROR GOTO deve aparecer no programa para prever a execução de um erro. Qualquer erro encontrado depois da instrução ON ERROR GOTO faz com que o computador execute o número de linha listado na instrução ON ERROR GOTO.

```
EXEMPLO 1
100 REM USANDO ON ERROR GOTO
110 ON ERROR GOTO 170
120 PRINT "DE ENTRADA NUM NUMERO E SERA CALCULADO O VALOR
INVERSO";
130 INPUT X
140 B = 1/X
150 PRINT "O INVERSO DE ";X; " E ";B
160 GOTO 120
170 PRINT "O INVERSO DE O NAO PODE SER CALCULADO TENTE DE
NOVO"
180 RESUME 120
190 END
```

```
EXECUÇÃO DO PROGRAMA (utilizando-se 7 e 0)
DE ENTRADA NUM NUMERO E SERA CALCULADO O VALOR INVERSO? 7
O INVERSO DE 7 E .142857
DE ENTRADA NUM NUMERO E SERA CALCULADO O VALOR INVERSO? 0
O INVERSO DE 0 NAO PODE SER CALCULADO TENTE DE NOVO
DE ENTRADA NUM NUMERO E SERA CALCULADO O VALOR INVERSO?
```

O erro,neste caso, foi a divisão por zero.

Caso ON ERROR GOTO 0 for executado durante uma sub-rotina ON ERROR GOTO, a mensagem de erro será impressa e a execução do programa irá parar. Teste essa característica adicionando ao exemplo a seguinte linha:

```
175 ON ERROR GOTO 0
```

A instrução RESUME é utilizada normalmente para desenvolver o comando a partir do programa principal para uma sub-rotina ON ERROR GOTO.

ORTOGRAFIA ALTERNATIVA
Algumas versões BASIC (tais como APPLESOFT) têm ON ERR GOTO (ao invés de ON ERROR GOTO)

VARIAÇÕES DE USO
Não se tem conhecimento de nenhuma.

VEJA TAMBÉM
ERROR, RESUME, ERR, ERL

Instrução **ON-GOSUB**

SINTAXE GERAL

ON K GOSUB XX,YY,ZZ,...WW

Número de linha ON variável GOSUB número de linhas

ON-GOSUB é uma sub-rotina múltipla de desvio que incorpora numa única instrução uma série de testes de IF-GOSUB; e aceita nas versões BASIC funcionando de maneira semelhante onde K representa uma variável numérica cujo valor será um número inteiro e positivo geralmente começando por (1).

Cada incremento de K encontrado causará uma passagem para a sub-rotina em sua correspondente linha número (XX,YY,ZZ,...WW) para execução da sub-rotina e retorno ao ponto de partida dentro do programa.

K deve ter um valor na faixa entre 0 e 255 ou a mensagem ERRO Quantidade Ilegal aparecerá no vídeo.

Se K for zero ou um valor maior que o número de linhas listado, a execução do programa continua a partir da linha seguinte ao ON GOSUB.

ON K GOSUB 160, 190, 220 por exemplo instrui o computador a se desviar para sub-rotinas, a começar nas linhas 160, 190, 220, se o valor de número inteiro de K for 1, 2 ou 3, respectivamente. Se INT K for menor do que 1 ou maior do que 3, todos os testes neste exemplo de ON-GOSUB irão falhar.

Em alguns computadores a execução cai então para a próxima linha de programa. Em outros, o programa entra em colapso, imprimindo-se uma mensagem de erro.

```
EXEMPLO 1
100 REM USANDO ON-GOSUB
110 PRINT "DE ENTRADA A UM NUMERO INTEIRO DE 1 A 3 ";
120 INPUT K
130 PRINT "A INSTRUCAO ON-GOSUB ";
140 ON K GOSUB 160, 190, 220
150 GOTO 110
160 REM SUB-ROTINA 1
170 PRINT "SE DESVIOU PARA A SUB-ROTINA 1"
180 RETURN
190 REM SUB-ROTINA 2
200 PRINT "SE DESVIOU PARA A SUB-ROTINA 2"
210 RETURN
220 REM SUB-ROTINA 3
230 PRINT "SE DESVIOU PARA A SUB-ROTINA 3"
240 RETURN
250 END
```

```
EXECUÇÃO DO PROGRAMA
RUN
DE ENTRADA A UM NUMERO INTEIRO DE 1 A 3
? 1
A INSTRUCAO ON-GOSUB SE DESVIOU PARA A SUB-ROTINA 1
```

```
DE ENTRADA A UM NUMERO INTEIRO DE 1 A 3
? 2
A INSTRUCAO ON-GOSUB SE DESVIOU PARA A SUB-ROTINA 2
DE ENTRADA A UM NUMERO INTEIRO DE 1 A 3
? 3
A INSTRUCAO ON-GOSUB SE DESVIOU PARA A SUB-ROTINA 3
DE ENTRADA A UM NUMERO INTEIRO DE 1 A 3
?
```

Utilize o mesmo exemplo e experimente dar entrada em valores decimais maiores do que 1 e menores que 4.

Experimente valores menores do que 1, mas maiores do que 4.

SE O SEU COMPUTADOR NÃO TIVER ESSA INSTRUÇÃO

Se o seu computador não passou o teste ON-GOSUB, substitua estas linhas:

```
135 IF K=1 GOSUB 160
140 IF K=2 GOSUB 190
145 IF K=3 GOSUB 220
```

(Tenha cuidado para que o valor de K não seja alterado na sub-rotina chamada.) Se essa sub-rotina funcionar, as funções INT intrínsicas podem ser duplicadas substituindo-se as seguintes linhas:

```
135 IF INT(K)=1 GOSUB 160
140 IF INT(K)=2 GOSUB 190
145 IF INT(K)=3 GOSUB 220
```

ORTOGRAFIA ALTERNATIVA

O TRS-80 Nível 1 aceita ON-GOS.

Esta instrução não é disponível na versão Integer Basic, para linha APPLE II porém deverá ser vista no manual deste microcomputador a instrução GOSUB, e a forma do GOSUB calculado para o Integer Basic.

VARIAÇÕES DE USO

Não se tem conhecimento de nenhuma.

VEJA TAMBÉM

ON-GOTO, ON-ERROR-GOTO, GOTO-OF, GOSUB

Instrução **ON-GOTO**

SINTAXE GERAL

ON K GOTO XX,YY,ZZ,...WW
ou
Número de linha ON variável GOTO número de linhas

ON K GOTO XX,YY,ZZ é uma instrução usada no modo programa.

ON K GOTO XX,YY,ZZ é aceito na maioria das versões BASIC e quando executado provoca múltiplas transferências que incorpora uma série de testes IF...THEN... numa única instrução.

K é uma variável numérica que representa um valor numérico na faixa entre 1 e 255 ou a mensagem de ERRO QUANTIDADE ILEGAL será emitida no vídeo.

Se K for zero ou um valor maior que o número de linhas listado, a execução do programa continua a partir da linha seguinte ao ON GOTO.

A cada incremento de K encontrado provocará uma passagem para seu número equivalente ou especificado em linha-número.

ON K GOTO 170,190,210, por exemplo instrui o computador para se desviar para as linhas 170,190 ou 220 se o valor de K for 1,2 ou 3 respectivamente.

O valor do número inteiro de K não pode ultrapassar o número de possíveis desvios na instrução. Se o valor de K for decimal, o computador encontra automaticamente seu valor de número inteiro e seleciona o número de linha de desvio apropriado.

EXEMPLO 1

```
100 REM ON(K) GOTO
110 PRINT "DE ENTRADA NO NUMERO 5,6 OU 7"
120 INPUT K
130 PRINT "A INSTRUCAO ON-GOTO ";
140 ON K GOTO 170,190,210
150 PRINT "FALHOU NO TESTE"
160 GOTO 230
170 PRINT "DESVIOU PARA A LINHA 170"
180 GOTO 110
190 PRINT "DESVIOU PARA A LINHA 190"
200 GOTO 110
210 PRINT "DESVIOU PARA A LINHA 210"
220 GOTO 110
230 END
```

EXECUÇÃO DO PROGRAMA (utilizando-se 5,6 e depois 7):

```
DE ENTRADA NO NUMERO 5,6 OU 7
? 5
A INSTRUCAO ON-GOTO DESVIOU PARA A LINHA 170
DE ENTRADA NO NUMERO 5,6 OU 7
? 6
A INSTRUCAO ON-GOTO DESVIOU PARA A LINHA 190
DE ENTRADA NO NUMERO 5,6 OU 7
? 7
A INSTRUCAO ON GOTO DESVIOU PARA A LINHA 210
```

DE ENTRADA NO NUMERO 5,6 OU 7
?

ORTOGRAFIA ALTERNATIVA

Alguns computadores permitem abreviaturas do GOTO na instrução ON-GOTO. O PDP-SE permite ON K GOT, ao passo que algumas versões BASIC permitem ON K G.

SE O SEU COMPUTADOR NÃO TIVER ESTA INSTRUÇÃO

Se o computador nao passou no teste ON-GOTO, substitua estas linhas:

```
135 IF K=5 THEN 170
140 IF K=6 THEN 190
145 IF K=7 THEN 210
```

Caso funcione essa substituição, as funções INT intrínsecas podem ser duplicadas substituindo-se as seguintes linhas:

```
135 IF INT(K)=5 THEN 170
140 IF INT(K)=6 THEN 190
145 IF INT(K)=7 THEN 210
```

VARIAÇÕES DE USO

Interpretadores diferentes podem ter um limite no número de opções de desvio (foram utilizadas 3 apenas a título de exemplo).

A instrução ON-GOTO é utilizada também com algumas outras palavras-chave para fins de desvio para outra parte do programa ao ocorrer determinada condição.

VEJA TAMBÉM

ON GOSUB, ON ERROR, GOTO-OF, GOTO

Comando/Instrução **ONNER GOTO**

SINTAXE GERAL

ONNER GOTO número de linha

Pula para a linha de número indicado quando acontecer erro num programa APPLESOFT.

Usando o comando ONNER GOTO da linguagem BASIC podem-se criar rotinas de tratamento de erros em BASIC que se referem às mensagens do DOS. Quando ocorre um erro de DOS, depois de um comando ONNER GOTO dentro de um programa, o código de erro é armazenado na posição de memória 222. Esta é a mesma posição em que o BASIC armazena o código das suas mensagens de erro.

Cada tipo de erro tem um código associado. Quando um erro ocorre dentro de um loop FOR-NEXT ou numa sub-rotina, os apontadores e as pilhas são alterados. A rotina de erro deve voltar para o início da declaração FOR ou GOSUB, reiniciando o loop ou a sub-rotina. Se a rotina de erro voltar para o NEXT ou o RETURN, outro erro poderá ocorrer.

ALGUMAS MENSAGENS DE ERROS COM SEUS CÓDIGOS PARA MICROCOMPUTADORES DA LINHA APPLE.

CODIGO = 1

Linguagem não-disponível. DOS (a versão Integer BASIC não está no disquete).

Quando o DOS não consegue encontrar uma linguagem de programação necessária para executar um comando.

CODIGO = 2 ou 3

Erro de faixa. DOS (parâmetro do comando muito grande).

Quando o valor de um parâmetro do comando for muito grande ou muito pequeno.

CODIGO = 4

Protegido cóntra escrita. DOS (o adesivo está colocado na fenda do disquete e protegendo este contra escrita).

Quando o DOS tenta armazenar uma informação em um disquete, mas o acionador não detecta o pequeno corte existente no envelope do disquete, isto é, existe um rótulo adesivo tapando o corte para protegê-lo contra escrita.

CODIGO = 5

Fim de dados. DOS (leitura alem do final de um arquivo-texto).

Ao tentar ler informações, além do fim de um arquivo de texto.

CODIGO = 6

Arquivo não encontrado. DOS (o nome do arquivo (programa ou texto) não consta do catálogo do disquete).

Quando um comando DOS especificar um arquivo que não existe no catálogo.

CODIGO = 7

Volume não-coincidente. DOS (parâmetro de volume ilegal).

Quando o parâmetro de volume (V) usado no comando do DOS não possui o mesmo número do volume assinalado para o disquete selecionado. O número do disquete pode ser identificado no

cabeçalho do catálogo mostrado pelo comando CATALOG.

CODIGO = 8

Erro de E/S (ERROR I/O) (porta do drive aberta ou disquete não inicializado).

Este tipo de erro ocorre quando:

1- a porta do acionador estiver aberta.

2- não existir disquete no acionador.

3- o disquete não estiver inicializado.

4- o disquete for inserido incorretamente.

5- não houver acionador correspondente ao parâmetro D.

6- não existir controlador no conector especificado pelo slot n.

7- um comando VERIFY descobre que o arquivo não está armazenado corretamente.

CODIGO = 9

Disco cheio. DOS (excesso de arquivos num disquete).

Quando o DOS tenta armazenar informações num disquete e não encontra espaço disponível. Podem ser utilizados, no máximo, 496 setores por disquete para armazenamento de informações. Se um arquivo individual excede 255 setores, o comando CATALOG mostrará o seu tamanho começando, novamente , pelo 000.

CODIGO = 10

Arquivo travado. DOS (tentativa de escrever por cima de um arquivo protegido).

Na tentativa de executar: SAVE, BSAVE, WRITE ou DELETE usando um nome de arquivo que foi travado (usando comando LOCK). Verifique o catálogo; os nomes de arquivos travados são precedidos por um asterisco (*).

CODIGO = 11

Erro de sintaxe. DOS (erro na sintaxe do comando ou vírgula incorreta).

Quando o DOS encontra um erro de sintaxe em seus comandos.

CODIGO = 12

Não há buffers disponíveis. DOS (excesso de arquivos-textos abertos).

Quando o DOS requer um outro "buffer" de arquivo para a entrada ou saída de dados quando todos os "buffers" disponíveis estiverem em uso. Na carga do DOS, o sistema reserva três "buffers" para entrada e saída de arquivo. O comando MAXFILES pode aumentar ou diminuir o número de "buffers". O comando OPEN reserva um "buffer" para o arquivo aberto e o comando CLOSE libera o respectivo "buffer".

O comando MAXFILES deve ser usado antes de carregar um programa na memória.

CODIGO = 13

Tipo de arquivo não-coincidente. DOS (o tipo de arquivo do disquete não coincide com o especificado no comando).

Quando o comando DOS tenta usar um arquivo cujo tipo não é apropriado para o comando em questão. Eis as combinações corretas:

1- LOAD a, SAVE a, RUN a:

"a" deve ser arquivo de programa BASIC.

2- OPEN a, READ a, WRITE a, APPEND a, POSITION a, EXEC a :

"a" deve ser um arquivo de texto.

3- BLOAD a, BRUN a, SAVE a :

"a" deve ser um arquivo binário.

CODIGO = 14

Programa muito comprido. DOS (memória insuficiente para

guardar o programa).

Quando um comando DOS tenta colocar um arquivo na memória do computador e descobre que a quantidade de memória disponível é insuficiente para conter todo o arquivo.

Pode haver casos em que um programa anterior tenha deixado o HIMEM com o valor muito baixo,ou um MAXFILES muito grande tenha deixado HIMEM baixo. Se fizermos com que o número de "buffers" seja igual a 3 usando o comando MAXFILES 3, então o HIMEM voltará ao valor original.

CODIGO = 15

Comando não-direto. DOS (o comando deve estar num programa).

Na tentativa de usar, diretamente, um comando para arquivo de texto (OPEN, APPEND, READ, WRITE, POSITION) fora das linhas de programa.

VEJA TAMBÉM

OPEN, APPEND, READ, WRITE, POSITION, CLOSE

Comando/DOS **OPEN**

SINTAXE GERAL

OPEN nomearq [,Ln] [,Dn] [,Sn] [,Vn]

OPEN nomearq é um comando do DOS 3.3 utilizado pelos microcomputadores que seguem a linha APPLE.

Prepara um arquivo de texto seqüencial ou aleatório para acesso. É um comando DOS e requer PRINT e CHR$(4) quando em modo programado. OPEN não pode ser usado em modo imediato.

OPEN significa abrir. Abrir um arquivo significa não apenas estabelecer a ligação entre o nome do arquivo físico e o número do arquivo lógico, no programa, mas permitir acesso do programa ao arquivo, isto é, ligá-lo ao programa.

OPEN permite E/S para um arquivo de disquete.

Um arquivo de disco deve ser aberto (OPEN) antes que se possa executar qualquer operação E/S de disquete nesse arquivo. OPEN aloca um buffer na memória de 595 bytes e determina o modo de acesso que será usado com o buffer. Os comandos READ nomearq e WRITE nomearq podem agora ser usados como arquivo para obter e guardar informações.

Se o arquivo especificado não está no disquete, será criado um. Se ele já está aberto, ele é fechado e em seguida reaberto.

Se o parâmetro L não é especificado, o arquivo é aberto como seqüencial.

Se o parâmetro L é especificado, o arquivo é aberto como de acesso aleatório (acesso direto).

Tamanho do registro é uma expressão de inteiros em bytes, na faixa de 1 a 32767.

OBSERVAÇÃO: Um arquivo pode ser aberto (OPEN) para entrada seqüencial ou acesso aleatório em mais de um número de arquivo por vez. No entanto, um arquivo só pode ser aberto para saída em apenas um número de arquivo por vez.

Se o disco no drive Dn do slot Sn não tiver o volume Vn, ocorre um erro.

Dn, Sn e Vn podem ser especificados em qualquer ordem, ou serem omitidos.

VARIAÇÕES DE USO

Não se tem conhecimento de nenhuma.

VEJA TAMBÉM

APPEND, CLOSE, WRITE, READ nomearq

Instrução **OPTION**

SINTAXE GERAL

Número de linha OPTION BASE =
ou
Número de linha OPTION BASE n

A instrução OPTION é utilizada em algumas versões BASIC com a instrução BASE para definir o elemento de matriz variável de BASE (mais baixo) como sendo qualquer valor de número inteiro de 0 a 10.

Exemplo:

```
100 OPTON BASE = 5
110 DIM X(10)
```

A instrução OPTION BASE define esse arranjo como tendo cinco elementos (X(6) até X(10)). Não sendo especificado o valor de OPTION BASE, o computador admite que o valor de BASE seja 0.

EXEMPLO 1

```
100 REM PROGRAMA DE TESTE OPTION
110 OPTION BASE = 4
120 DIM X(6)
130 FOR Y = 4 TO 6
140 X(Y) = Y
150 NEXT Y
160 OPTION BASE = 0
170 FOR Y = 0 TO 2
180 X(Y) = Y
190 NEXT Y
200 FOR Y = 0 TO 6
210 PRINT X(Y);
220 NEXT Y
230 PRINT "A INSTRUCAO OPTION FOI APROVADA"
240 END
```

EXECUÇÃO DO PROGRAMA

```
RUN
0 1 2 3 4 5 6 A INSTRUCAO OPTION FOI APROVADA
```

VARIAÇÕES DE USO

O BASIC Padrão da ANSI especifica apenas os valores de OPTION BASE 0 E 1, não sendo exigido sinal de = após a palavra BASE.

Nos computadores da linha LABO utilizamos a seguinte:

SINTAXE GERAL

OPTION BASE n

onde n e 1 ou 0

tem por finalidade declarar o valor mínimo para índices matriciais.

OBS.: A base implícita é 0. Se o comando OPTION BASE 1 for executado, o valor mais baixo que um índice matricial pode ter é um.

VEJA TAMBÉM

BASE

Instrução **OR**

SINTAXE GERAL

Número de linha IF condição OR condição...THEN...

OR é uma instrução que funciona como operador lógico utilizada dentro de uma instrução IF...THEN... para estabelecer uma comparação ou relação, equivalente a "ou". Os operadores lógicos são usados para manipulações de bits e operações booleanas.

As operações são realizadas bit a bit; OR funcionando como operador lógico fará com que o resultado de uma expressão lógica seja verdadeira se pelo menos uma das condições operadas for verdadeira. Para que o resultado seja falso todas as condições operadas têm que ser falsas. Existe, porém, uma diferença na forma de operar dos computadores que seguem a linha APPLE para os da linha TRS-80 e os da linha SINCLAIR.

Exemplo: X = Y ou Z

Nos computadores da linha APPLE, o resultado de tal operação fará com que X assuma o valor zero se ambos, Y e Z, forem zero, e 1(um) se pelo menos um dos operandos for diferente de zero. Já nos computadores da linha TRS-80 e SINCLAIR, o resultado em X será igual ao do maior dos dois operandos em questão (em termos absolutos). O valor de X somente será zero(0) se ambos, Y e Z, forem iguais a zero.

IF A = 3 OR B = 4 THEN 160

significa "se o valor da variável A for igual a 3 ou o valor da variável B for igual a 4, ou ambas, se satisfaz a condição IF-THEN, e a execução cai para a linha 160".

```
EXEMPLO 1
100 REM USANDO OR LOGICO
110 B = 4
120 A = 2
130 IF A = 3 OR B = 4 THEN 160
140 PRINT "OR NAO APROVOU COMO OPERADOR LOGICO"
150 GOTO 170
160 PRINT "OR APROVOU COMO OPERADOR LOGICO"
170 END

EXECUÇÃO DO PROGRAMA
RUN
OR APROVOU COMO OPERADOR LOGICO
```

Alguns computadores permitem que o operador OR seja utilizado para realização de testes conjugados com strings

IF A$ = "A" OR A$ = "APROVOU" THEN...

se interpreta "se a variável de string A$ contiver a letra A ou a palavra APROVOU, se satisfaz a condição IF-THEN.

```
EXEMPLO 2
200 REM USANDO OR LOGICO COM STRING
210 A$ = "APROVOU"
220 B$ = "NAO APROVOU"
230 IF A$ = "APROVOU" OR B$ = "NAO APROVOU" THEN 160
```

```
240 PRINT "OR NAO APROVOU COMO OPERADOR LOGICO COM STRING"
250 GOTO 170
260 PRINT "OR APROVOU COMO OPERADOR LOGICO COM STRING"
270 END

EXECUÇÃO DO PROGRAMA
RUN
OR APROVOU COMO OPERADOR LOGICO COM STRING
```

Alguns computadores utilizam o operador OR para determinar se as condições foram satisfeitas em qualquer um dos dois operadores lógicos. Sendo satisfeita a condição de um dos operadores, pelo menos, OR devolve um valor verdadeiro (A-1 na maioria dos computadores, devendo o leitor verificar o manual do seu sistema). Não sendo satisfeita nenhuma das duas condições, OR devolve o valor falso (0).

PRINT X = 7 OR Y > 8

significa que, se X tiver um valor de 7 ou o valor armazenado em Y for maior do que 8, ou ambas as coisas, o computador irá imprimir o número -1. Não sendo satisfeita nenhuma das duas condições, o computador irá imprimir 0.

```
EXEMPLO 3
300 REM USANDO OR LOGICO
310 PRINT "DE ENTRADA NUM NUMERO ENTRE 5 E 15";
320 INPUT X
330 Y = X <5 OR X>15
340 IF Y <> 5 THEN 370
350 PRINT X; " E UM NUMERO ENTRE 5 E 15"
360 GOTO 310
370 PRINT X; " E MAIOR QUE 5 E MAIOR QUE 15"
380 GOTO 310
390 END

EXECUÇÃO DO PROGRAMA
RUN
DE ENTRADA NUM NUMERO ENTRE 5 E 15? 7
7 E UM NUMERO ENTRE 5 E 15
DE ENTRADA NUM NUMERO ENTRE 5 E 15? 17
17 E MAIOR QUE 5 E MAIOR QUE 15
```

O operador OR é utilizado por alguns computadores para computar o OR binário lógico de dois números de acordo com as regras da álgebra booleana. OR compara as formas binárias de dois números, bit a bit. Se qualquer um dos dois bits OR-ados for um, o computador devolve um.

```
EXEMPLO
0 OR 0 = 0
0 OR 1 = 1
1 OR 0 = 1
1 OR 1 = 1
```

Assim sendo, se o computador executar a operação OR entre um número e outro, os bits de cada número são OR-ados logicamente com os bits correspondentes do outro número, produzindo-se um terceiro número.

EXEMPLO

DECIMAL		BINÁRIO
3		0011
	OR	
5		0101
------		------
= 7		0111

VARIAÇÕES DE USO

O operador OR é utilizado algumas vezes sob forma diferente, OR(P$,Q$), para modificar strings.

Alguns computadores americanos e japoneses aceitam OR(A$,B), onde A$ é string e B é constante hexadecimal.

SE O SEU COMPUTADOR NÃO TIVER ESTA INSTRUÇÃO

Se não estiver disponível no seu computador o operador OR lógico, o resultado pode ser simulado por subtração e multiplicação. Substitua a linha 130 do exemplo 1 por:

```
130 IF (A-2) * (B-6) = 0 THEN 160
```

VEJA TAMBÉM

AND, XOR, NOT, +, *

Instrução **OUT**

SINTAXE GERAL

Número de linha OUT PORTA,valor

OUT é uma instrução usada geralmente no modo programa que quando executada abre o canal de OUTPUT para o port(slot).

OUT tem necessidade de dois argumentos separados por vírgulas (sem parênteses). OUT não é uma função mas sim uma instrução completa que manda um byte para uma porta de saída.

Tanto o destino (porta) quanto o valor do byte a ser enviado devem ser números inteiros e positivos e estar na faixa de 0 e 255.

Exemplo:

```
100 OUT 255,4
```

quando executado o valor quatro é enviado à porta 255.

OBS: OUT é uma instrução que depende muito do hardware. A interface de uma porta, com aplicações específicas, deve ser projetada pelo usuário. Os valores de byte e de port devem ser números inteiros positivos ou variáveis entre 0 e 256.

Pressione o botão PLAY do gravador cassete e experimente este programa.

EXEMPLO 1 - configurado para micros da linha TRS-80

```
100 REM USANDO A INSTRUCAO OUT
110 PRINT "DE ENTRADA EM 4 PARA LIGAR O MOTOR DO GRAVADOR CASSETE"
120 INPUT G
130 OUT 255,G
140 PRINT "DE ENTRADA EM 0 PARA DESLIGAR O MOTOR"
150 INPUT G
160 OUT 255,G
170 END
```

EXECUÇÃO DO PROGRAMA

```
DE ENTRADA EM 4 PARA LIGAR O MOTOR DO GRAVADOR CASSETE
? 4
DE ENTRADA EM 0 PARA DESLIGAR O MOTOR
```

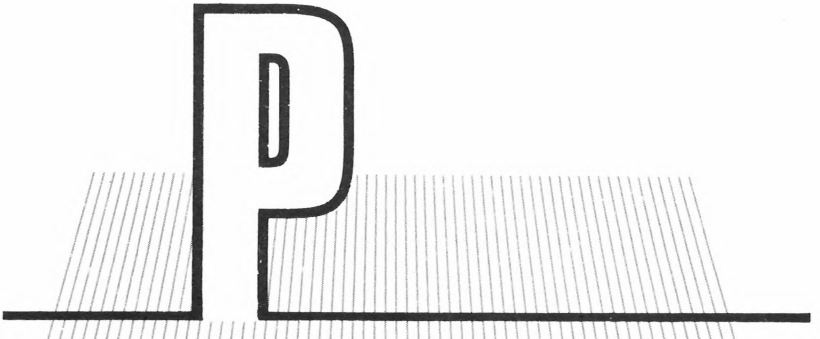
Se o motor do gravador cassete não se ligou, experimente este programa para verificar quais os números de porta e de byte que funcionam para o seu computador.

EXEMPLO 2

```
100 REM USANDO OUT
110 FOR P = 0 TO 255
120 FOR V = 0 TO 255
130 PRINT "PORTA n.";P
140 PRINT "VALOR BYTE n.";V
150 OUT P,V
160 NEXT V
170 NEXT P
180 END
```

VARIAÇÕES DE USO
Não se tem conhecimento de nenhuma.

VEJA TAMBÉM
INP, PEEK, POKE



PDL • PEEK • PI
PLOT • POINT • POKE • POP
POS • POSITION • PRECISION
PR # • PRINT • PRI • P. • ?
PRINT AT • PRINT USING

Função **PDL**

SINTAXE GERAL

PDL(exprnm)
ou
Número de linha variável = PDL (expressão numérica)

PDL(X) é uma função que deve ser utilizada no modo programa devendo ser precedida de uma instrução geralmente LET.

A função PDL(X) utilizada nos micros da linha APPLE II quando se refere ao controle dos jogos (X) normalmente será um valor entre 0 e 3 indicando qual dos controles de jogos deve ser lido.

Se for indicado um controlador menor que zero ou maior que 255, será gerada uma mensagem de erro "QUANTIDADE ILEGAL"

PDL retorna o valor corrente do controlador de jogo (paddle) especificado.

PDL é abreviatura de Paddle e se refere aos "paddles" de controle de jogos.

O valor retornado é um inteiro entre 0 e 255 baseado na rotação do padlle número expressão numérica, ou da resistência do dispositivo conectado ao soquete do controlador de jogo expressão numérica. Se o controlador for entre 4 e 255, o PDL retorna um valor imprevisível entre 0 e 255, podendo causar vários efeitos colaterais, tal como um sinal no alto-falante ou um súbito deslocamento no modo gráfico.

Se duas instruções PDL são executadas consecutivamente ou muito próximas, o segundo valor pode ser afetado pelo primeiro. Providencie para que várias instruções sejam executadas entre dois PDL (ou coloque um loop FOR-NEXT vazio).

EXEMPLO 1

```
100 REM USANDO PDL
110 J = PDL(0)
120 P = PDL(1)
130 PRINT "O VALOR DE PDL(0) E";J
140 PRINT "O VALOR DE PDL(1) E";P
150 PRINT "MUDE OS POSICIONAMENTOS DA UNIDADE DE CONTROLE E RODE DE NOVO"
160 END
```

EXECUÇÃO DO PROGRAMA

```
RUN
O VALOR DE PDL(0) E 13
O VALOR DE PDL(1) E 146
MUDE OS POSICIONAMENTOS DA UNIDADE DE CONTROLE E RODE DE NOVO
```

VARIAÇÕES DE USO

Esta função não é disponível nos micros que seguem a linha TRS-80 sem modificações.

VEJA TAMBÉM

GR, PLOT, KEY$

Função PEEK

SINTAXE GERAL

Número de linha LET variável = PEEK (endereço memória)
ou
Número de linha PRINT
ou
PEEK (endereço de memória)

PEEK (N) é uma função utilizada no modo programa devendo ser precedida por uma instrução geralmente PRINT ou LET.

A função PEEK(N) é utilizada para examinar o conteúdo de determinado endereço na memória do computador; este conteúdo refere-se ao valor decimal equivalente contido na memória e especificado por (N). Com a função PEEK verifica-se o conteúdo de qualquer posição de memória. O valor fornecido por PEEK é em decimal.

EXEMPLO:

100 LET X = PEEK(18136)

Quando executada esta linha a variável X irá conter o mesmo valor que estiver armazenado na memória cujo endereço é 18136. Para usar a função PEEK é necessário consultar as seguintes seções:

a) o apêndice do mapa de memória do seu computador (para saber onde buscar o valor);

b) tabela do código ASCII e códigos gráficos, para saber o que representam os valores retornados por PEEK.

A instrução PEEK devolve o conteúdo de um endereço de memória sob forma de um número entre 0 e 255 (a faixa de valores que pode ser armazenada numa célula de memória com 8 bits). PEEK pode ser utilizado com a instrução POKE para ler o que POKE introduz na memória. O número de endereço de valor mais elevado que pode ser desenvolvido pela instrução PEEK dependerá, evidentemente, do tamanho da memória do computador.

Examine o manual do seu computador antes de executar este exemplo, para determinar que os endereços de memória 18368 a 18380 são reservados para as operações normais do computador. Se os endereços 18368 a 18380 não forem reservados como memória livre no seu computador, selecione então um grupo de 13 endereços de memória consecutivos livres e altere as linhas 110 e 150 no exemplo.

```
EXEMPLO 1 (configuração para TRS-80)
100 USANDO A FUNCAO PEEK
110 FOR N = 18368 TO 18380
120 READ A
130 POKE N,A
140 NEXT N
150 FOR N = 18368 TO 18380
160 A = PEEK(N)
170 PRINT CHR$(A);
180 NEXT N
190 DATA 84,69,83,84,69,128,128,65,67,69,73,84,79
200 END
```

EXECUÇÃO DO PROGRAMA

TESTE ACEITO

EXEMPLO 2 (configuração para linha SINCLAIR)

```
200 REM USANDO FUNCAO PEEK
210 PRINT "ENDERECO ";TAB 8;"BYTE"
220 FOR X = 0 TO 20
230 PRINT X; TAB 8; PEEK X
240 NEXT X
```

Este exemplo 2 escreve os primeiros 21 bytes da ROM (e seus endereços).

A função PEEK é utilizada para ligar rotinas em linguagem de máquina com programas em BASIC.

VARIAÇÕES DE USO

Para usar PEEK em endereço acima de 32767, use a seguinte fórmula:

-1 * (65536 - endereço desejado)

Para utilizar PEEK no endereco 32900 use: PEEK (-32636)

Nos microcomputadores que seguem a linha TRS-80 no nível 1 esta função não é disponível.

VEJA TAMBÉM

POKE, USR(X), SYSTEM, FILL

Função **PI**

SINTAXE GERAL

Número de linha variável = expressão

PI é utilizado para representar o valor de π (3.14159265358979).

O perimetro em centímetros de um círculo com um centímetro de diâmetro. PI não tem argumento (apenas dez dígitos do mesmo são realmente acumulados no computador, e só oito aparecerão no vídeo).

PI é uma função devendo ser precedida de uma instrução PRINT ou LET.

Esta função é aceita por interpretadores que reconhecem as funções matemáticas.

Tente nos micros que seguem a linha SINCLAIR este exemplo:

```
EXEMPLO 1
100 PRINT PI
110 PRINT PI-3
120 PRINT PI-3.1
130 PRINT PI-3.14
140 PRINT PI-3.141
```

Executando o programa o microcomputador mostrará a precisão no armazenamento de II.

```
EXEMPLO 2
200 REM USANDO PI
210 R = 8
220 C = 2*PI*R
230 PRINT "A CIRCUNFERENCIA DE UM CIRCULO"
240 PRINT "COM RAIO DE 8 CM.";C;" CM."
250 END

EXECUÇÃO DO PROGRAMA
RUN
A CIRCUNFERENCIA DE UM CIRCULO
COM RAIO DE 8 CM. E 50.2654825 CM.
```

SE O SEU COMPUTADOR NÃO TIVER ESTA FUNÇÃO

Se o seu computador não tiver a função PI, substitua a mesma pelo valor de 3.14159265.

Instrução **PLOT**

SINTAXE GERAL

Número de linha PLOT coluna, linha
ou
Número de linha PLOT, X,Y

PLOT X,Y

Nos microcomputadores que seguem a linha APPLE a instrução PLOT X,Y executa um bloco gráfico em baixa resolução especificado pelos valores das variáveis X e Y ou por qualquer expressão usada para produzir um número dentro dos limites da tela.

Estando o computador no modo gráfico de baixa resolução PLOT X,Y coloca um ponto colorido na tela .

O número de coluna X varia de 0 a 39, sendo 0 a coluna mais à esquerda da tela e 39 a mais à direita. As linhas são representadas por Y, que varia de 0 a 47, sendo 0 a linha acima e 47 a de baixo. Se um ponto for colocado nas linhas de 40 a 47, cairá nas quatro linhas da janela de texto, a menos que seja utilizada uma instrução POKE -16302,0 para eliminar a janela de texto. Cada linha de texto requer 2 fileiras, o que possibilita dispor 4 linhas de texto por debaixo da apresentação visual dos gráficos.

A cor é atribuída de acordo com a instrução COLOR executada.

PLOT 5,20, por exemplo, instrui o computador a colorir um bloco para gráficos localizado na quinta coluna e na vigésima fileira (da matriz de gráficos). Para "desligar" o bloco de gráficos individuais, a cor 0 (preto) deve ser selecionada para cada bloco.

```
EXEMPLO 1
100 REM USANDO PLOT
110 GR
120 COLOR = 4
130 PLOT 0,0
140 PLOT 39,0
150 PLOT 39,39
160 PLOT 0,39
170 END
```

EXECUÇÃO DO PROGRAMA
RUN

Se o computador aceitou a instrução plot e seu vídeo for colorido, deve aparecer uma ponta verde em cada canto da tela.

Nos microcomputadores que seguem a linha SINCLAIR o vídeo apresenta 22 linhas e 32 colunas, o que dá 22 vezes 32 = 704 posições de caracteres cada qual contendo 4 pontos a serem plotados.

A instrução PLOT X,Y torna um ponto preto no vídeo com estas coordenadas, enquanto que a instrução UNPLOT volta a pô-lo em branco.

```
EXEMPLO 2 (linha SINCLAIR)
200 REM USANDO PLOT
210 PLOT INT (RND*64), INT (RND*44)
220 INPUT A$
```

```
230 GOTO 210
```

Ao ser executado este programa, ele desenha ponto numa posição a cada vez que for pressionada a tecla ENTER ou NEW LINE.

Porém, este outro exemplo fará um gráfico da função SIN (seno) para valores entre 0 e 2π (2 PI).

```
EXEMPLO 3
300 REM USANDO PLOT
310 FOR X = 0 TO 63
320 PLOT X,22 + SIN(N/32*PI)
330 NEXT X
```

VARIAÇÕES DE USO

Nos computadores que seguem a linha TRS-80, usa-se a instrução SET(X,Y)

VEJA TAMBÉM

CR, COLOR, TEXT, HLIN-AT, VLIN-AT, SET, RESET, POINT, DRAW, UNPLOT

Função **POINT**

SINTAXE GERAL

Número de linha IF POINT (X,Y) THEN...

POINT (X,Y) é uma função especial utilizada em conjunto com as instruções IF...THEN nos microcomputadores que seguem a linha TRS-80; seu propósito é testar se um determinado ponto da tela de vídeo está aceso (SET) ou apagado (RESET). Se o ponto está aceso (isto é, se ele foi SET), então a função POINT retorna um binário real (-1 em BASIC). Se o ponto está apagado, a função POINT retorna um binário falso (0 em BASIC).

EXEMPLO:

```
10 SET (50,28)
20 IF POINT (50,28) THEN PRINT "LIGADO"
30 PRINT "DESLIGADO"
```

Executando o programa imprimirá a mensagem "LIGADO", porque POINT retornará um binário real; então a execução procede para após THEN; se falhasse POINT retornaria um binário falso executando a linha 30. A função POINT é utilizada em conjunto com as instruções IF-THEN, pelo computador TRS-80, como característica especial para indicar se está ou não ligado um bloco específico de gráficos.

EXEMPLO 1

```
100 REM USANDO A FUNCAO POINT
110 CLS
120 FOR A = 10 TO 20 STEP 2
130 SET (A,7)
140 NEXT A
150 PRINT "POINT FOI ACEITO SE OS NUMEROS 11111111111
APARECEREM NO VIDEO"
160 FOR A = 10 TO 20
170 B = 1
180 IF POINT (A,7)=1 THEN B = 0
190 PRINT B;
200 NEXT A
210 GOTO 210
220 END
```

EXECUÇÃO DO PROGRAMA (Nível 1)

```
POINT FOI ACEITO SE OS NUMEROS 11111111111 APARECEREM NO
VIDEO
 1  1  1  1  1  1  1  1  1  1  1
```

Para obter os mesmos resultados para Nível II, mude a linha 180 para:

```
180 IF POINT(A,7) = -1 THEN A = 0
```

VARIAÇÕES DE USO

Nos microcomputadores que seguem a linha APPLE II pode ser usado SCRN (X,Y) para ter uma aproximação.

Para o nível 1 um valor devolve -1. Ambos esses níveis devolvem 0, caso o programa não esteja iluminado.

VEJA TAMBÉM

SET, RESET, CLS

Instrução/Comando **POKE**

SINTAXE GERAL

Número de linha endereço, valor

POKE X,Y pode ser utilizado como comando no modo direto ou como instrução no modo programa.

POKE X,Y relaciona-se com o equivalente em linguagem de máquina da variável Y no endereço de memória X.

POKE X,Y coloca no endereço de memória indicado por X o número inteiro ou conteúdo da variável inteira X.

Use com cuidado a instrução POKE. Algumas posições de memória contêm informação essencial para a operação ininterrupta do seu computador.

Mexer nestas posições pode destruir o seu programa ou travar o sistema operacional ou ainda alterar o BASIC.

O comando ou instrução POKE X,Y permite que seja colocado um valor (numérico ou alfanumérico) numa localização específica de memória, enquanto que uma função PEEK (X) precedida de uma instrução PRINT permite verificar o conteúdo de qualquer posição da memória. O valor fornecido por PEEK é em decimal.

EXEMPLO 1

```
100 REM USANDO A INSTRUCAO POKE
110 PRINT PEEK (18850)
120 POKE 18850,5
130 PRINT PEEK (18850)
```

EXECUÇÃO DO PROGRAMA

Na linha 110 apresenta-se na tela de vídeo o dado que está na memória cujo endereço é 18850.

Na linha 120 colocamos o número 5 nesta localização e na linha 130 verificamos se realmente o número 5 foi armazenado naquele endereço.

A instrução POKE é utilizada para armazenar valores de números inteiros de 0 até 255 (decimais) em localizações específicas da memória. POKE 15360.66, por exemplo, coloca o número ASCII 66 (que é a letra "B"), no endereço de memória 15360.

VARIAÇÕES DE USO

Este comando/instrução POKE X,Y não é disponível nas versões BASIC de nível I

OBS: Para usar PEEK e POKE em endereços acima de 32767, use a seguinte fórmula:

-1 * (65536 - endereço desejado)

VEJA TAMBÉM

PEEK, FILL

Instrução **POP**

SINTAXE GERAL

Número de linha POP

POP é uma instrução devendo ser utilizada no modo programa, e reconhecida na versão BASIC dos microcomputadores que seguem a linha APPLE II.

POP é utilizado em programas com sub-rotinas fazendo, quando executado, que o computador esqueça a posição de retorno para o mais recente GOSUB executado. A instrução POP quando executada muda o GOSUB mais recentemente executado em uma instrução GOTO.

A próxima instrução RETURN executada desviará para a instrução em seguida ao segundo mais recente GOSUB executado.

POP (estoura) o endereço de um número de linha GOSUB para fora do topo de uma pilha de memória que o armazena. Tão logo o programa encontre um RETURN, examina a pilha para verificar onde deve recomeçar a execução, mas se engana. Vamos experimentar isto de novo. Cada vez que se executa um GOSUB, o endereço em linguagem de máquina é armazenado numa seção especial de memória denominada de pilha de "empurrar para baixo". O último valor armazenado nessa pilha é o primeiro valor que será lido e utilizado.

Uma instrução RETURN lê esse endereço de cima "na pilha", para determinar para onde "devolver" o controle do programa depois de se ter completado o seu GOSUB.

No exemplo 1, o endereço, em linguagem de máquina, da linha 110, é armazenado no topo da pilha, ao passar o programa, sob efeito do GOSUB, para a linha 140. Executando-se o GOSUB na linha 140, seu endereço é empilhado no topo do endereço da linha 110.

A instrução POP na linha 170 empurra o endereço da linha 140 para fora do topo da pilha e o descarta. Ao se dirigir a instrução RETURN da linha 180 para a pilha para ver onde se deve recomeçar a execução do programa, encontra o endereço da linha 110 em vez do de linha 140. A execução recomeça no final da linha 110 e se desloca para a linha 120.

POP pode ser utilizado onde um resultado calculado ou uma condição de erro requerem a bifurcação para outro lugar do programa em vez de RETURN para o GOSUB mais recente.

EXEMPLO 1

```
100 REM USANDO POP
110 GOSUB 140
120 PRINT "POP FOI ACEITO"
130 GOTO 190
140 GOSUB 170
150 PRINT "POP NAO FOI ACEITO"
160 GOTO 190
170 POP
180 RETURN
190 END
```

EXECUÇÃO DO PROGRAMA
RUN
POP FOI ACEITO

OBS: **Se** o número total de instruções POP e RETURN executadas num programa exceder o número de instruções GOSUB, aparecerá uma mensagem de erro no vídeo.

SE O SEU COMPUTADOR NÃO TIVER ESTA INSTRUÇÃO

Se o seu computador falhou no teste POP, podem ser produzidos resultados semelhantes utilizando-se "ponteiros" (flags = "bandeiras")

EXEMPLO 2

```
200 REM USANDO POP
210
220 GOSUB 250
230 PRINT "VOLTE AO 230"
240 GOTO 310
250 GOSUB 290
260 IF R = 1 THEN 280
270 PRINT "NAO PARE AQUI"
280 RETURN
290 R = 1
300 RETURN
310 END
```

EXECUÇÃO DO PROGRAMA

```
RUN
VOLTE AO 230
```

VARIAÇÕES DE USO

Não se tem conhecimento de nenhuma.

VEJA TAMBÉM

GOSUB, RETURN, ON-GOSUB, GOSUB-OF

Instrução **POS**

SINTAXE GERAL

Número de linha variável POS (expressão numérica)

POS (0) é uma função aceita nos BASIC para versões de microcomputadores que seguem a linha APPLE ou TRS-80 de maneiras idênticas.

POS (0) devolve o valor da posição horizontal (0 para a extensão do campo TAB) do cursor no lugar onde se encontra.

POS (expressão numérica)

A expressão numérica e fictícia, podendo ter qualquer valor legal.

POS indica a posição do cursor na linha horizontal do vídeo variando de 0 a 39 nos micros que seguem a linha APPLE e de 0 a 63 nos micros que seguem a linha TRS-80. Sendo 0 para o carácter mais a esquerda.

POS(n,F$,G$) é função de string que encontra a posição inicial da primeira ocorrência do string G$ dentro do string F$. Não sendo encontrado $ dentro de F$, POS = 0.

POS (N) onde N é facultativo, sendo utilizado para iniciar a busca no n-ésimo carácter de F$. Se não for utilizado n, a busca começa no primeiro carácter de F$, o que fica mais para a esquerda.

```
EXEMPLO 1
100 REM USANDO FUNCAO POS
110 F$ = "MICRO"
120 G$ = "VIDEO"
140 IF R<>5 THEN 180
150 G$ = "VIDEO"
160 R = POS(F$,G$)
170 IF R<>0 THEN 180
180 PRINT "POS (F$,G$) FOI ACEITO"
190 GOTO 210
200 PRINT "POS (F$,G$) NAO FOI ACEITO"
210 END
```

```
EXECUÇÃO DO PROGRAMA
RUN
POS (F$,G$) FOI ACEITO
```

VARIAÇÕES DE USO

Alguns interpretadores (tais como BASIC da Microsoft) utilizam a função POS para informar a posição do cursor na linha atual de impressão. O valor de 'n' não tem importância - e apenas um número simulado.

Esta função não é disponível na versão Integer BASIC para os micros que seguem a linha APPLE II.

```
EXEMPLO 2
200 REM USANDO FUNCAO POS
210 PRINT "ESTAS" TAB(POS(0)+5)"PALAVRAS"
TAB(POS(0)+5)"ESTARAO";
```

```
220 PRINT TAB(POS(0)+5)"SEMPRE" TAB(POS(0)+5)"SEPARADAS"
```

EXECUÇÃO DO PROGRAMA

```
RUN
ESTAS    PALAVRAS    ESTARAO    SEMPRE      SEPARADAS
```

VEJA TAMBÉM

INSTR, INDEX

Comando **POSITION**

SINTAXE GERAL

POSITION nomearq [,Rn]

POSITION nomearq [,Rn] é um comando do DOS utilizado em arquivos seqüenciais e aceito em microcomputadores que seguem a linha APPLE.

POSITION nomearq [,Rn]

Este comando, quando executado, coloca o ponteiro de posição no começo do enésimo campo especificado em [,Rn] após o campo corrente.

Ao mover o apontador de arquivo do disco, move-se um determinado número de posições para a frente.

Este comando permite escrever (WRITE) e ler (READ) informações a partir de qualquer campo dentro de um arquivo seqüencial. O parâmetro R especifica quantos campos o apontador de arquivo se move para a frente a partir de sua posição presente. O número a seguir ao R deve ser uma constante inteira num valor entre 0 e 32767. Se ausente, o parâmetro adotado é 0, ou seja, o apontador não se move. Se o arquivo é aberto por POSITION, os campos são contados a partir do seu início. Esse arquivo deve ser seqüencial.

POSITION indica que o ponteiro "posição dentro do arquivo" seja movido sempre para a frente, pulando os campos.

Se o arquivo não possuir nenhum campo correspondente à posição especificada, será emitida a mensagem FIM DE ARQUIVO parando a máquina.

Um campo consiste numa série de caracteres terminados por RETURN. O POSITION vai ao longo do arquivo, carácter por carácter, contando tais caracteres; o número de carriages returns encontrado é o número de campos pulados.

O comando OPEN, automaticamente, posiciona o ponteiro "posição dentro do arquivo" no início do primeiro campo. Portanto, se o comando POSITION for usado imediatamente depois de um OPEN, a posição relativa de campo corresponderá, também, à posição absoluta de campo.

Como qualquer outro comando DOS, o comando POSITION cancela READ ou WRITE.

Portanto, POSITION deve ser usado antes do READ ou WRITE, associado.

Este é um comando DOS, e requer PRINT e CHR$(4) em modo programado.

POSITION não pode ser usado em modo imediato.

VARIAÇÕES DE USO

Não se tem conhecimento de nenhuma.

VEJA TAMBÉM

APPEND, OPEN, WRITE, CLOSE

Instrução **PRECISION**

SINTAXE GERAL

Número de linha PRECISION N

PRECISION é uma instrução utilizada no modo programa e reconhecida em apenas algumas versões BASIC de alguns computadores americanos e europeus.

A instrução PRECISION, conforme utilizada, se destina a especificar o número máximo de dígitos a serem impressos para a direita do ponto decimal, mediante uma instrução PRINT. 200 PRECISION 2, por exemplo, poderia ser utilizada num programa em que os valores impressos devam representar cruzeiros e centavos. Se o valor real for mais extenso do que o número de dígitos especificado, o número será arredondado para o número de casas decimais desejado, não sendo exibidos os dígitos mais para a direita.

EXEMPLO 1

```
100 REM USANDO PRECISION
110 PRECISION 4
120 N = 0.1544589
130 PRINT N
140 PRINT "PRECISION FOI ACEITO SE TIVER SIDO IMPRESSO 0.1545"
150 END
```

EXECUÇÃO DO PROGRAMA

```
RUN
0.1545
PRECISION FOI ACEITO SE TIVER SIDO IMPRESSO 0.1545
```

SE O SEU COMPUTADOR NÃO TIVER ESSA INSTRUÇÃO

Não estando disponível no seu computador a instrução PRECISION, experimente a instrução DIGITS na linha 110.

O número máximo de dígitos após o ponto decimal pode também ser controlado apagando-se a linha 110, e substituindo-se a linha 130 por:

```
130 PRINT INT (N*10000 + .5)/10000
```

VARIAÇÕES DE USO

Não se tem conhecimento de nenhuma

VEJA TAMBÉM

DIGITS, PRINT USING, INT

Comando/DOS **PR #**

SINTAXE GERAL

PR# slot

PR# é um comando do DOS 3.3 utilizado nos microcomputadores que seguem a linha APPLE. Inexiste um equivalente na linha TRS-80 sem uma interface de expansão e o hardware periférico associado.

Seleciona o slot que irá receber as próximas saídas.

PR# X transfere a saída para o conector número X, onde X deve ser um inteiro entre 0 e 7.

EXEMPLO 1

PR#6 manda as saídas subseqüentes do microcomputador da linha APPLE, para o periférico conectado no slot 6 (geralmente a drive), em vez da tela do vídeo.

O comando PR#0 devolve as saídas à tela do vídeo.

EXEMPLO 2

PR#1 manda as saídas subseqüentes para o periférico controlado pelo slot 1 (usualmente a impressora).

PR# 0 especifica a tela de vídeo como saída.

Sempre que o DOS estiver presente na memória dos micros da linha APPLE II , PR# será considerado um comando DOS, e vai requerer PRINT e CHR$(4) em modo programado.

Nos microcomputadores que seguem a linha APPLE, o comando determina que o OUTPUT do computador siga a posição de saída (X) indicada pelo usuário.

Em alguns casos é possível direcionar o OUTPUT para o gravador cassete usando o comando PRINT #-1.

VARIAÇÕES DE USO

Não se tem conhecimento de nenhuma.

VEJA TAMBÉM

IN#s ou LPRINT (no seu manual TRS-80)

Comando/Instrução **PRINT**
PRI • P. • ?

SINTAXE GERAL

PRINT lista de itens
ou
PRINT expressão, variável, valor (numérico ou alfanumérico)
ou
Número da Linha PRINT expressão, variável, valor (numérico ou alfanumérico)

PRINT pode ser utilizado como comando no modo direto ou instrução no modo programa.

PRINT coloca uma informação especificada pelo usuário em um determinado ponto na tela ou em outro dispositivo de saída.

PRINT imprime um item ou uma lista de itens na tela do vídeo; estes itens podem ser constantes ou string(mensagens entre aspas), variáveis string, constantes numéricas (números), variáveis ou expressões envolvendo todos os itens precedentes. Os itens a serem impressos devem ser separados por vírgulas ou por ponto-e-vírgula.

PRINT determina ao computador para apresentar na tela do vídeo ou em uma impressora, resultados de expressões já efetuados os cálculos, variáveis no seu valor atual, valores numéricos ou uma mensagem alfanumérica desde que fornecida entre aspas.

Nos microcomputadores que seguem as linhas APPLE, TRS-80 e SINCLAIR pode-se utilizar a função TAB(X), juntamente com o PRINT para fazer a tabulação, onde X deve ser um valor inteiro variando de 0 a 255. Nos computadores que seguem a linha APPLE, porém, a tabulação pode ser efetuada com a utilização das funções HTAB X e VTAB Y onde X pode ser um número inteiro e positivo variando de 1 a 40 controlando a impressão na coluna especificada na linha horizontal do vídeo, e, Y representa também um número inteiro e positivo variando de 1 a 24 controlando a impressão nas linhas apresentadas na tela.

A instrução PRINT @ X, dos computadores que seguem a linha TRS-80 e a instrução PRINT AT X, Y dos micros que seguem a linha SINCLAIR são semelhantes às duas instruções HTAB e VTAB dos da linha APPLE.

Como existem discrepâncias nas dimensões de tela das três linhas de equipamentos apresentamos as dimensões de cada um:

TRS-80: 16 linhas por 64 colunas (ou 32 colunas)
APPLE: 24 linhas por 40 colunas (ou 80 colunas)
SINCLAIR: 22 linhas por 32 colunas

EXEMPLO 1

```
100 REM USANDO A INSTRUCAO PRINT
110 PRINT 50 + 30 - 60
120 PRINT X
130 PRINT 51
140 PRINT "ESTES SAO OS RESULTADOS"
150 END
```

```
EXECUÇÃO DO PROGRAMA
RUN
 20
 0
 51
ESTES SAO OS RESULTADOS
```

OBS.: 20 é o valor da expressão na linha 110
zero é o valor atual da variável X por não ter sido ainda atribuído valor algum.
51 valor numérico simples na linha 130
o texto apresentado é somente o que estava entre aspas na linha 140.

PRINT tem uma ampla variedade de uso:
Ex PRINT 2 + 2 ordena que o computador resolva a soma e imprima na tela do vídeo a resposta.
PRINT permite também que se faça no computador as funções de uma calculadora podendo não só somar mas também subtrair, multiplicar (usando o asterisco *) em vez do sinal de vezes e dividir (utilizando / em vez de -). Pode também elevar um número à potência de outro utilizando o operador ** ou ↑ ou ↑ ou !.
O computador resolverá também combinações das operações.

EXEMPLO
PRINT 40 - 3 ↑ 2 * 2 + 6 / 2

Dá-nos a resposta 25.

$$40 - \underline{3 \uparrow 2} * 2 + 6 / 2$$

$$40 - \underline{9 * 2} + 6 / 2$$

$$40 - 18 + \underline{6 / 2}$$

$$\underline{40 - 18} + 3$$

$$\underline{22 + 3}$$

$$25$$

OBS.: O computador na execução percorrerá todas as linhas analisando a expressão da esquerda para a direita dando prioridade às potências (↑) e então todas as multiplicações e divisões (* e /) e por fim as adições e subtrações (+ e -)
NOTA: podemos formalizar isto dando a cada operação uma prioridade, um número entre 1 e 16. As operações com maior prioridade são avaliadas em primeiro lugar, e as operações com igual prioridade são avaliadas da esquerda para a direita.
↑ exponenciação tem prioridade 10.
* e / multiplicação e divisão têm prioridade 8.
+ e - adição e subtração têm prioridade 6.
Quando - é utilizado para ter acesso aos números negativos,

como quando se escreve -1, então tem prioridade de 9. (É uma operação unitária, oposta da subtração 3-1: uma operação unitária tem um operando, enquanto uma operação binária tem dois). Esta ordem é absolutamente rígida, mas pode-se ultrapassar utilizando parênteses: qualquer expressão aritmética ou numérica entre parênteses é avaliada em primeiro lugar e é depois tratada como um só número.

Quando as vírgulas são utilizadas numa instrução PRINT o cursor avança automaticamente para fazer com que os itens individuais sejam impressos em zonas horizontais predeterminadas com aproximadamente 16 caracteres por linha. As larguras das zonas de impressão variam de computador para computador.

```
EXEMPLO 2
200 REM USANDO , NA INSTRUCAO PRINT
210 PRINT "ZONA 1","ZONA 2"
220 PRINT "123456789012345","123456789012345"
230 END
```

```
EXECUÇÃO DO PROGRAMA
RUN
ZONA 1           ZONA 2
123456789012345 123456789012345
```

OBS.:quando uma vírgula for encontrada, o cursor se moverá para a próxima zona de impressão.

O ; (ponto-e-vírgula) é muito usado na instrução PRINT para rejuntar (concatenar) numa só linha elementos de palavras ou frases. Um ponto-e-vírgula anula o retorno do carro, assim sendo, o próximo PRINT inicia onde o último parou.

```
EXEMPLO 3
300 REM USANDO ; NUMA INSTRUCAO PRINT
310 PRINT "MICRO";
320 PRINT "COMPUTADOR"
330 END
```

```
EXECUÇÃO DO PROGRAMA
RUN
MICROCOMPUTADOR
```

OBS.: O ponto-e-vírgula funciona de maneira semelhante à vírgula, mas imprime os valores de saída concentrados uns ao lado dos outros, ao invés de ficarem em zonas predeterminadas.

NOTA: Se não colocarmos nenhuma pontuação após a instrução ou comando PRINT, o cursor passará para o início da próxima linha. Pode-se imprimir ou pular linhas em branco na tela; para isso basta usar a instrução PRINT sem nada escrever adiante.

```
EXEMPLO 4
400 REM USANDO PRINT SEM PONTUACAO E PRINT VAZIO
410 PRINT "NOME:"
420 PRINT " CLOVIS PEREIRA"
430 PRINT
440 PRINT "PROFISSAO:"
450 PRINT " PROFESSOR"
460 END
```

EXECUÇÃO DO PROGRAMA
RUN
NOME:
CLOVIS PEREIRA

PROFISSAO:
PROFESSOR

IMPRESSÃO COM FORMATOS:
Informações Gerais
A instrução TAB(n) é utilizada em conjunto com a instrução PRINT de maneira semelhante à aplicação da tecla de tabulação numa máquina de escrever.

SINTAXE
Número da linha PRINT TAB(coluna)

TAB(coluna) move a posição PRINT para a coluna especificada. Fica na mesma linha ou, se isso implicasse retrocesso de espaço, movê-la-ia para a seguinte.
Para mais informações a respeito, veja-se TAB

A função AT é utilizada com PRINT no BASIC Nível 1 dos computadores da linha TRS-80 e em computadores que seguem a linha SINCLAR como o TK-83, TK85, NE-80, NE-8000 e CP-200.

SINTAXE
Numero da linha PRINT AT linha da tela, coluna

AT move a posição PRINT (a posição onde o próximo item vai ser impresso) para a linha e a coluna especificadas. As linhas na tela do vídeo estão numeradas de 0 (parte de cima) a 21 (parte de baixo) e as colunas de 0 (à esquerda) até 31 (à direita).

OBS.: O número de linhas e colunas da tela de vídeo varia de computador para computador.

O operador @ é utilizado no Nível II e III do TRS-80 e computadores que seguem esta linha como CP-500, DGT-1000, e outros

SINTAXE
Número da linha PRINT @, posição, lista de itens

O PRINT @ especifica exatamente onde a impressão deve ser iniciada.
Para mais informações veja também PRINT AT e PRINT @ (arroba).

O PRINT USING é utilizado em alguns computadores como característica PRINT especial; esta instrução possibilita a especificação de um formato para a impressão dos valores numéricos e alfanuméricos. Ele pode ter muitas aplicações, tais como: impressão de cabeçalhos, títulos de relatórios, relatórios de contabilidade, cheques, extratos de contas ou onde um formato de impressão específico for necessário.
Para mais informações consulte a seção PRINT USING.

LPRINT é utilizado por algumas versões BASIC da Microsoft para enviar uma instrução PRINT a uma impressora de linha ao invés de encaminhá-la para a tela. LPRINT pode ser utilizado quer como comando direto quer como instrução de programa.

SINTAXE:
LPRINT variável, constante, ou expressão

LPRINT também pode ser usado com todas as opções disponíveis para PRINT, exceto PRINT @. Na impressora LPRINT USING imprime a informação na impressora utilizando um formato específico. LPRINT TAB moverá o carro da impressora para a direita atendendo à tabulação que for especificada.

EXEMPLO:
LPRINT TAB(4) "NOME" TAB(35) "ENDERECO"
Para mais informações consulte a seção LPRINT.

PRINT # é utilizado em alguns computadores, para armazenar (gravar) todos os valores de variáveis especificados em fita cassete. Para que esta instrução seja executada o gravador deve estar adequadamente conectado e colocado no modo de Gravação.

SINTAXE:
PRINT # -1, lista de itens

A instrução PRINT # deve sempre indicar o número do dispositivo, isto porque alguns computadores podem trabalhar com mais de um gravador. Para uso normal, com apenas um gravador conectado, o número do dispositivo deve ser -1.

EXEMPLO: 5
500 REM USANDO A INSTRUCAO PRINT #
510 PRINT "OS DADOS DEVEM ESTAR SENDO GRAVADOS NA FITA CASSETE"
520 PRINT # -1, "FICHA: ",1,2
530 PRINT "PRINT #-1 COMPLETOU A TRANSFERENCIA DE DADOS PARA FITA CASSETE"
540 END

EXECUÇÃO DO PROGRAMA
RUN
OS DADOS DEVEM ESTAR SENDO GRAVADOS NA FITA CASSETE
PRINT # - 1 COMPLETOU A TRANSFERENCIA DE DADOS PARA FITA CASSETE

OBS.: Os valores podem ser copiados posteriormente da fita, utilizando-se a instrução INPUT #-1. A instrução #-1 deve ser idêntica a instrução INPUT #-1, em relação ao número e tipo de itens na lista PRINT #-1.
VEJA TAMBÉM INPUT #.

NOTA: para se certificar de que os dados foram gravados, quando da execução do programa EXEMPLO 5, rebobine (Rewind) a fita, coloque o gravador no modo "Play" e execute este programa.

600 REM DADOS DE ENTRADA DA FITA CASSETE
610 PRINT "O COMPUTADOR DEVE ESTAR LENDO DADOS DA FITA

```
CASSETE"
     620 INPUT # -1,A$,A,B
     630 PRINT "FORAM LIDOS DA FITA CASSETE OS SEGUINTES DADOS:"
     640 PRINT A$,A,B
     650 END

     EXECUÇÃO DO PROGRAMA
     RUN
     O COMPUTADOR DEVE ESTAR LENDO DADOS DA FITA CASSETE
     FORAM LIDOS DA FITA CASSETE OS SEGUINTES DADOS:
     FICHA          1              2
```

Para mais informações veja a seção específica do PRINT # e do INPUT #

ORTOGRAFIAS ALTERNATIVAS

Alguns computadores europeus e americanos aceitam PRI ou P. como formas abreviadas da palavra chave PRINT.

Sendo a instrução PRINT uma das mais utilizadas em muitas versões BASIC, ela é substituída por um único símbolo.

o North Star BASIC por exemplo utiliza !, o DEC BASIC-PLUS utiliaza &, o ABC-80, da Suécia, utiliza ; e o MAXI-BASIC do Digital Group utiliza #. Entretanto a maioria dos computadores que aceita o BASIC da Microsoft a palavra chave PRINT é freqüentemente substituída por um único símbolo, o ponto de interrogação (?).

VEJA TAMBÉM

PRINT AT, PRINT USING, LPRINT, CUR, LIN, ;(ponto e vírgula) ,(vírgula), %, ?

Instrução **PRINT AT**

SINTAXE GERAL

Número de linha PRINT AT X,Y

Nos microcomputadores que seguem a linha SINCLAIR a instrução PRINT com a ajuda da função AT pode determinar exatamente uma posição na tela do vídeo onde se deseje colocar uma informação.

PRINT AT X,Y onde X,Y representam o número do endereço ou as coordenadas que acompanham PRINT AT X,Y endereça as posições da tela como se esta fosse uma matriz bidimensional.

AT linha, coluna move a posição PRINT (o lugar onde o próximo item deve ser impresso), para a linha e colunas especificadas. As linhas são numeradas de 0 a 21 enquanto que as colunas são numeradas de 0 a 31.

Pode-se AT para posicionar o PRINT até mesmo onde já existe algo escrito.

Neste caso o conteúdo existente será apagado.

```
EXEMPLO 1
100 REM USANDO A INSTRUCAO PRINT AT
110 PRINT AT 8,10 "SULLIVAN"
120 PRINT AT 8,10 "SULLIVAN"
130 GOTO 110
```

PRINT AT também é aceito e utilizado pelo BASIC Nível 1 dos computadores que seguem a linha TRS-80, para indicar a posição inicial de uma instrução PRINT. O valor AT pode ser um número, uma variável numérica ou uma operação matemática. Devem ser inseridos entre o valor AT e o string uma vírgula ou ponto e vírgula. Existem 1024 localizacoes na tela, sendo 16 fileiras com 64 endereços horizontais em cada.

```
EXEMPLO 2
200 PRINT AT 256, "SULLIVAN"
210 PRINT AT (256); "SULLIVAN"
```

Ambas as linhas imprimem a palavra SULLIVAN na localização 256, sendo facultativos os parênteses.

```
EXEMPLO 3
300 REM USANDO PRINT AT
305 CLS
310  PRINT  AT  256, "3. SE ESTA LINHA FOR IMPRESSA DEPOIS DA
LINHA 2"
320 PRINT AT 128,"2. A INSTRUCAO PRINT AT FOI ACEITA"
330 GOTO 330

EXECUÇÃO DO PROGRAMA
RUN
2. A INSTRUCAO PRINT AT FOI ACEITA
3. SE ESTA LINHA FOR IMPRESSA DEPOIS DA LINHA 2.
```

Para interrupção do programa pressione a tecla BREAK.

ORTOGRAFIAS ALTERNATIVAS

O BASIC Microsoft substitui PRINT @ por PRINT AT, e requer uma vírgula depois da localização.

No BASIC APPLESOFT usa-se HTAB X VTAB Y.

VARIAÇÕES DE USO

Não se tem conhecimento de nenhuma.

VEJA TAMBÉM

PRINT AT, TAB, HTAB X e VTAB X

Instrução **PRINT USING**

SINTAXE GERAL

Número de linha PRINT USING lista de itens

PRINT USING é utilizado pela maioria dos microcomputadores que seguem a linha TRS-80 e em versões BASIC com CPM.

PRINT USING é a instrução PRINT mais poderosa e mais complexa disponível na linguagem BASIC.

Esta instrução permite especificar um formato para impressão alfanumérico e valores numéricos. Pode ser usada em muitas aplicações, tais como: Impressão de títulos de relatórios, relatórios de contabilidade (extrato de contas), cheques ou sempre que um formato de impressão for requerido.

A instrução PRINT USING usa o seguinte formato:

PRINT USING alfanumérico; valor.

O alfanumérico e o valor podem ser expressões variáveis ou constantes. Esta instrução imprimirá a expressão contida no alfanumérico, inserindo o valor numérico mostrado à direita do ponto-e-vírgula, como determinado pelos especificadores de campo.

Os seguintes especificadores de campo podem ser usados no alfanumérico:

O símbolo especifica a posição de cada dígito localizado no valor numérico. O número de símbolos diferentes que você usa estabiliza o campo numérico. Se o campo numérico é maior que o número de dígitos do valor numérico, então as posições não usadas do campo à esquerda do número serão mostradas como espaços e aquelas à direita do ponto decimal serão mostradas como zeros.

O símbolo # reserva uma posição para cada dígito num número ou numa variável numérica, para esquerda e para direita de um ponto decimal. Inserem-se automaticamente zeros, se não houver nada para a direita, fazendo com que este dispositivo seja valioso para a impressão de dados financeiros. O símbolo # imprime sempre o ponto decimal no mesmo lugar, o que facilita examinar fileiras de números.

O ponto decimal(.) pode ser colocado em qualquer lugar no campo numérico estabilizado pelo sinal #. O arredondamento tem lugar quando os dígitos à direita do ponto decimal são suprimidos.

, A vírgula quando colocada em qualquer posição entre o primeiro dígito e o ponto decimal mostrará uma vírgula à esquerda de cada terceiro dígito, como requerido. A vírgula estabiliza uma posição adicional do campo. A posição da vírgula na instrução PRINT USING não afeta a posição da vírgula impressa.

** Dois asteriscos colocados no começo do campo farão com que todas as posições não usadas à esquerda do decimal sejam preenchidas com asteriscos. Os dois asteriscos estabilizarão mais duas posições no campo. O asterisco (*) pode ser impresso em todos os espaços não utilizados à esquerda do ponto decimal de

determinado número. A finalidade primordial é evitar que alguém aumente o valor constante de um cheque impresso pelo computador.

$$ Dois cifrões colocados no começo do campo atuarão como um cifrão oscilante. Estes ocuparão a primeira posição precedente ao número. O sinal $ pode ser impresso antes do número listado na instrução PRINT USING, inserindo o sinal de dólar duplo ($$) diante do sinal #.

^^^^ Faz com que o número seja impresso no formato exponencial (E ou D).

Quatro sinais exponenciais podem ser utilizados após o sinal # para imprimir números expressos em notação exponencial ou científica. Nos computadores que seguem a linha do TRS-80 usa-se !!!! em vez de ^^^^.

+ Quando um símbolo + é colocado no começo ou final do campo, ele será impresso tão especificado quanto um + para números positivos ou quanto um - para números negativos.

Sinal de + colocado para a esquerda dos sinais # faz com que o sinal + seja impresso antes de cada número positivo e sinal - antes de cada número negativo. Sendo colocado sinal de + para a direita dos sinais #, o computador imprime sinal de - para a direita de todos os números negativos, inserindo-se um espaço para a direita de todos os números positivos.

EXEMPLO:

```
PRINT USING "+###";231
PRINT USING "###+";231
```

imprimira os numeros:

+231 e 231-

- Quando um símbolo é colocado no final do campo, causará o aparecimento de um sinal negativo após todos os números negativos e como um espaço para números positivos.

% Para especificar quando um campo alfanumérico de mais de um carácter é usado. A extensão do campo alfanumérico será 2, mais o número de espaços entre os sinais de porcentagem.

A maioria dos computadores permite que os operadores números e strings de PRINT USING sejam especificados como variáveis.

! Faz com que o computador use o primeiro carácter alfanumérico de valor corrente. O sinal ! imprime apenas o carácter mais para a esquerda de um string ou variável de string, listada numa instrução PRINT USING.

EXEMPLO:

```
PRINT USING "!";"SULLIVAN"
```

imprime a letra S.

Qualquer outro carácter que se inclua no alfanumérico USING será mostrado como um alfanumérico literal.

O seguinte programa ajudará a demonstrar estes especificadores de formato:

```
100 INPUT A$,A
110 PRINT USING A$;A
120 GOTO 100
```

RUN este programa e tente variados especificadores alfanuméricos para A$ e valores variados para A.

Por exemplo:

```
RUN
? ##.##,12.12
12.12
? ###.##,12.12
12.12
? ##.##,121.21
% 121.21
```

O sinal % é automaticamente mostrado se o campo não for grande o suficiente para conter o número de dígitos achados no valor numérico. O número total à esquerda do decimal será mostrado, precedido por este símbolo:

```
? ##.##,12.127
12.13
```

Note que o número foi arredondado para duas casas decimais.

```
? +##.##,12.12
+12.12
? "A RESPOSTA E +##.##",-12.12
A RESPOSTA E -12.12
?##,##+,12.12
12.12+
?"A RESPOSTA E ##.##+.",-12.12
A RESPOSTA E 12.12-.
?"A RESPOSTA E ##.##-.",12.12
A RESPOSTA E 12.12.
?##.##-,-12.12
12.12-
?"**** NO TOTAL.",12.12
**12 NO TOTAL.
?****.##,1212.12
1212.12
?$$###.##,12.12
$12.12
?"##,####",12121.2
12,121
?"####,# NO TOTAL.",12121.2
12,121 NO TOTAL
?"### NO TOTAL.",1212
%1212 NO TOTAL
```

Outra forma de usar a instrução PRINT USING é com os especificadores de campo alfanumérico "!" e % espaços %.

Você não pode usar o carácter de declaração % (número inteiro) desta forma.

VARIAÇÕES DE USO

Utilizando-se STR$(X) e VAL(string) nos microcomputadores que seguem a linha APPLE, consegue-se simular o uso de PRINT USING, nestes micros.

VEJA TAMBÉM

VAL, STR$

RAD • RADIAN
RANDOM • RAN • RAND
READ • RECALL • REM
RENAME • RENUMBER • RESET
RESTORE • REST. • RES •
RESUME • RETURN • RET • R.
RIGHT$ • RND •
RANDON • RANDOMIZE
ROT = • RUN • RU • R.

Instrução/Comando **RAD • RADIAN**

SINTAXE GERAL

Número de linha RAD

RAD pode ser utilizada como comando no modo direto ou como instrução no modo programa

RAD é utilizada e reconhecida em algumas versões BASIC, para fazer com que se realizem cálculos trigonométricos em RADians ao invés de graus. A maioria dos computadores está no modo radianos ao serem ligados, mas alguns possuem também a capacidade de calcularem funções trigonométricas em graus. Se uma instrução DEG tiver sido utilizada num programa, a instrução RAD será necessária para fazer voltar o computador ao seu modo "normal".
Um radiano equivale a aproximadamente 57 graus.

EXEMPLO 1

```
100 REM USANDO RAD
110 PRINT "DE ENTRADA NUMA MEDIDA DE ANGULO (EM GRAUS)";
120 INPUT X
130 Y = RAD(X)
140 PRINT "UMA MEDIDA DE ";X;" GRAUS E IGUAL A ";Y;" RADIANOS"
150 END
```

EXECUÇÃO DO PROGRAMA

```
RUN
DE ENTRADA NUMA MEDIDA DE ANGULO (EM GRAUS)?37
UMA MEDIDA DE 37 GRAUS E IGUAL A .6457721 RADIANOS
```

ORTOGRAFIA ALTERNATIVA

Alguns computadores (tais como o Sharp TRS-80 de Bolso) utilizam RADIAN com a instrução que coloca o computador no modo de radianos para fins de cálculos trigonométricos.

VARIAÇÕES DE USO

Alguns BASIC (tais como MAX BASIC) utilizam RAD (n) como função para converter um valor (n) de graus para radianos.

SE O SEU COMPUTADOR NÃO TIVER ESSA INSTRUÇÃO

Se o seu computador não tiver a instrução RAD, esta poderá ser simulada multiplicando-se os valores em graus por .0174533. Para utilizar esse fator de conversão no exemplo 1, substitua a linha 130 por:

```
130 Y = X*0.0174533
```

VEJA TAMBÉM

DEG, GRAD, ACS, ASN, ATN, COS, SIN, TAN

Instrução **RANDOM**
RANDOMIZE • RAN • RAND

SINTAXE GERAL

Número de linha RANDOM
Número de linha RANDOMIZE

RANDOM ou RANDOMIZE é uma instrução usada no modo programa, que renumera randomicamente (aleatoriamente) um grupo de números, porém reconhecida em algumas versões BASIC; verifique o manual do seu computador.

RANDOM é de fato uma instrução completa, em vez de uma função. Ela é uma instrução que gera números aleatórios, se um programa usar a função RND(N).

Deve-se colocar RANDOM no começo do programa, para garantir que se tenha uma seqüência imprevisível de números pseudo-aleatórios a cada vez que o programa for executado.

A instrução RANDOMIZE é utilizada para "remexer" ou "realimentar" um conjunto de números (guardados no computador) em seqüência aleatória. Os respectivos números são gerados na medida das necessidades conforme tenham de ser selecionados pela função RND.

Colocar RANDOMIZE num programa diante da função RND faz com que se gere um novo conjunto de números aleatórios para a função RND cada vez que o programa seja rodado.

EXEMPLO 1

```
100 REM USANDO RANDOM
110 RANDOM
120 FOR R = 1 TO 100
130 PRINT RND(R)
140 NEXT R
150 PRINT "RANDOM NECESSITA SER EXECUTADO APENAS UMA VEZ"
160 END
```

Cada vez que se roda o programa deve ser impresso um novo conjunto de números aleatórios.

Algumas versões de linguagens BASIC (tais como BASIC-80) esperam que o valor "semente" seja incluído na instrução RANDOMIZE 49817. Não sendo incluída a "semente", o programa para e o usuário é instigado a dar entrada num valor de semente, a partir do teclado.

ORTOGRAFIA ALTERNATIVAS

Diversos computadores utilizam a palavra chave RANDOM (tais como no TRS-80) ao invés de RANDOMIZE. Algumas versões BASIC aceitam RAN como forma abreviada.

Na versão BASIC para SINCLAIR usa-se RAND.

SE O SEU COMPUTADOR NÃO TIVER ESTA INSTRUÇÃO

Se o seu computador não tiver RANDOMIZE talvez a utilização de um número negativo em conjunto com RND irá "ressemear" ou

passar a gerar uma nova seqüência de números. Ver (RND). Se assim não for, pode-se utilizar um simples procedimento com cada programa para fazer com que apareçam novas seqüências de números para cada rodada.

Substitua a linha 110 do exemplo 1 e adicione as seguintes linhas:

```
110 PRINT "DE ENTRADA NUM NUMERO INTEIRO PARA RND";
102 INPUT Y
104 FOR Z = 1 TO Y
106 R = RND(R)        ou RND(0)
108 NEXT Z
```

e rode o programa diversas vezes, utilizando números diferentes. A razão por que isto funciona é que RND gera a mesma seqüência de números cada vez que se liga o computador (ou cada vez que se tecla RUN, para certos computadores). Com efeito, RND "lê" a partir do topo de uma relação de números aleatórios cada vez que se utiliza o programa. O nosso procedimento induz o programa a"descartar" o topo da lista e começar a utilizar a entrada (S+1)-ésimo. Isto permite diferentes pontos iniciais para cada valor dado.

VARIAÇÕES DE USO
Não se tem conhecimento de nenhuma.

VEJA TAMBÉM
RND

Instrução **READ**

SINTAXE GERAL

READ X
Número de linha READ var,var,var

READ é uma instrução devendo ser utilizada no modo programa.
READ X esta instrução é utilizada associando-se valores contidos a outra instrução chamada DATA. O par de instruções READ...DATA é aceito pela maioria dos microcomputadores exceto nos que seguem a linha SINCLAIR.

A instrução READ X fará com que a variável indicada (X) aceite os valores relacionados na instrução DATA.

READ lê seqüencialmente os dados da instrução DATA, começando com o primeiro item da instrução DATA e terminando com o último item.

Os dados da relação na instrução DATA devem ser coerentes com os tipos de variáveis que a instrução DATA lê. Toda vez que é executada uma instrução READ são lidos dados de uma linha com a instrução DATA, todos os valores relacionados na instrução DATA devem ser separados por vírgula, porém não pode haver vírgula após a palavra-chave (instrução) DATA.

O ponteiro se desloca então para o próximo dado na(s) linha(s) de DATA e espera a próxima instrução READ. Depois que o último dado tiver sido lido de todas as instruções DATA, o ponteiro de dados deve ser reposto no início da relação de DATA antes que possam ser executadas novas instruções READ (Ver RESTORE).

Se isto não for feito haverá uma mensagem de erro: FIM DE DADOS.

```
EXEMPLO 1
100 REM USANDO A INSTRUCAO READ
110 READ X
120 PRINT "A INSTRUCAO READ FOI ACEITA NA LINHA";X
130 DATA 110
140 END

EXECUÇÃO DO PROGRAMA
A INSTRUCAO READ FOI ACEITA NA LINHA 110
```

Uma vez que os computadores permitem colocar mais de uma variável numa instrução READ, cada variável deve estar separada por uma vírgula da próxima, sendo que o número de READs não deve ultrapassar o número de itens de dados, e os dados lidos devem ser coerentes com as variáveis.

```
EXEMPLO 2
200 REM USANDO A INSTRUCAO READ MULTIPLA
210 READ A$,N,B$
220 PRINT A$,N,B$
230 DATA TENHO,40,ANOS
240 PRINT "A INSTRUCAO READ FUNCIONOU SE IMPRIMIR N="N
250 END
```

```
EXECUÇÃO DO PROGRAMA
TENHO      40     ANOS
A INSTRUCAO READ FUNCIONOU SE IMPRIMIR N=40
```

A maioria dos computadores permitem a leitura de strings a partir de instruções DATA. Toda vez que um string for lido de uma instrução DATA, deverá haver uma variável de string correspondente numa instrução READ.

Muitos computadores permitem ler dados tanto numéricos como de string mediante a mesma instrução READ, estando ambos contidos na mesma linha de DATA.

```
EXEMPLO 3
300 REM USANDO A INSTRUCAO READ MULTIPLA
310 READ F$,N$,S
320 PRINT "FUNCIONARIO", "NOME", "SALARIO"
330 PRINT F$,N$,S
340 IF N$= "FIM" THEN PRINT "FIM DA LISTA":END
350 DATA N.1,ANTONIO,500000,N.2,CLOVIS,900000
360 DATA N.3,ROSSANA,800000,N.4,FIM,0,0
370 GOTO 110
```

```
EXECUÇÃO DO PROGRAMA
FUNCIONARIO     NOME        SALARIO
N.1             ANTONIO     500000
FUNCIONARIO     NOME        SALARIO
N.2             CLOVIS      900000
FUNCIONARIO     NOME        SALARIO
N.3             ROSSANA     800000
FUNCIONARIO     NOME        SALARIO
N.4             FIM         0
FIM DA LISTA
```

ORTOGRAFIAS ALTERNATIVAS

Alguns computadores permitem abreviaturas de READ, aceitando REA ao invés de READ.

VARIAÇÕES DE USO

READ só pode ser executado no modo imediato se o programa na memória tiver os valores em DATA correspondentes.

A única outra maneira de armazenar e chamar os dados é dando entrada através do teclado ou a partir de armazenamento fora de linha em fita magnética e disquete. Se o DOS está presente, o READ em modo imediato é tratado como um comando do DOS e uma mensagem de erro (COMANDO NÃO DIRETO) será apresentada.

Na versão INTEGER BASIC para os computadores da linha APPLE a instrução READ não está disponível.

Nos BASIC de NÍVEL 1 não há leitura de strings.

VEJA TAMBÉM

DATA, RESTORE, , (vírgula)

Comando/Instrução **RECALL**

SINTAXE GERAL

Número de linha RECALL varnm

RECALL é utilizado na versão BASIC APPLESOFT dos microcomputadores que seguem a linha APPLE II tanto como instrução como comando direto a fim de carregar uma matriz de valores numéricos a partir de fita cassete.

Em programas nos micros da linha APPLE o comando RECALL X fará com que o computador execute INPUTS DE DADOS (DATA) estocados em fita de um gravador cassete. Os dados serão sob a forma de variáveis numéricas simples ou subscritas, real ou integral, ou inteira.

Sendo executado um comando RECALL varnm o interpretador BASIC APPLESOFT espera indefinidamente até que uma variável seja achada na fita; nenhuma outra instrução poderá ser executada neste interim. O comando RECALL não controla o movimento da fita nem informará quando deverá acionar o botão PLAY do gravador.

O microcomputador também sinaliza com um bip quando começa a carregar e dá outro bip quando pára.

A variável deve ser dimensionada (DIM) antes do comando RECALL ou aparecerá a mensagem de erro (FIM DE DADOS).

Quando utiliza-se RECALL não é necessário usar os mesmos nomes de variáveis usadas na instrução STORE que armazenou o programa.

A instrução/comando STORE armazena valores de arranjos e a instrução/comando RECALL os lê de volta.

Se o arranjo chamado tiver mais elementos que o gravado, valores obtidos normalmente serão prejudicados.

Há entretanto duas opções: chama-se para um arranjo com o mesmo número de dimensões do gravado, onde cada dimensão exceto a última tenha o mesmo tamanho da dimensão correspondente ao arranjo gravado. A última dimensão pode ser mais larga no arranjo chamado.

Pode-se também chamar para um arranjo com mais dimensões que o gravado, se as dimensões que estão no arranjo coincidem com aquelas do arranjo chamado ou as excedem no caso de última dimensão do arranjo gravado.

EXEMPLO 1

```
100 REM USANDO RECALL
110 DIM X(20) , Y(20)
120 FOR R = 1 TO 20
130 X(R) = R
140 NEXT R
150 STORE X
160 PRINT "ACIONE RECORD E PLAY - PRESSIONE RETURN"
170 INPUT X$
180 RECALL Y
190 FOR R = 1 TO 20
200 PRINT Y(R),
210 NEXT R
220 END
```

EXECUÇÃO DO PROGRAMA

```
ACIONE RECORD E PLAY - PRESSIONE RETURN
?
1               2               3
4               5               6
7               8               9
10              11              12
13              14              15
16              17              18
19              20
```

VARIAÇÃO DE USO

Não se tem conhecimento de nenhuma.

ORTOGRAFIA ALTERNATIVA

Nos computadores que seguem a linha TRS-80 usa-se INPUT 1 e as funções são aproximadamente as mesmas do RECALL X.

Este comando/instrução não é disponível na versão Integer BASIC.

VEJA TAMBÉM

STORE, CLOAD, DIM

Instrução **REM**

SINTAXE GERAL

Número de linha REM comentário.

REM é uma instrução e deve ser utilizada no modo programa.

A instrução REM serve para especificar uma linha de comentários; entende-se por comentários qualquer seqüência que a linha do programa acolher.

O microcomputador ignora tudo que vem depois da instrução REM. Esta instrução é usada para documentar um programa. As instruções REM são reproduzidas nas listagens dos programas, porém são ignoradas durante a sua execução.

A instrução REM tem funções idênticas na maioria dos sistemas, permitindo que os usuários incluam especificações ao longo de um programa apenas para informação de quem for executá-lo, sem interferir com as funções do próprio programa.

A instrução REM pode ser colocada numa linha sozinha ou em uma linha de instruções múltiplas.

A instrução REM não pode ser colocada antes de outras instruções numa linha múltipla, pois todo o texto que segue ao REM será tratado como um comentário.

Caso as REM (comentários) exigirem mais de uma linha cada uma das respectivas linhas deve começar com REM.

EXEMPLO 1

```
100 USANDO REM
110 REM ESTE PROGRAMA CALCULA O QUADRADO DE UM NUMERO
120 INPUT X
130 PRINT "O QUADRADO DE";X
140 PRINT "E IGUAL A:";X*X
150 END
```

EXECUÇÃO DO PROGRAMA

```
? 7
O QUADRADO DE 7
E IGUAL A: 49
```

Alguns computadores permitem a instrução REM ou REMARK, outros aceitam apenas uma variante.

VARIAÇÕES DE USO

Não se tem conhecimento de nenhuma.

Comando/DOS **RENAME**

SINTAXE GERAL

RENAME nomearq1, nomearq2 [,Dn] [,Sn] [,Vn]

RENAME é um comando do DOS 3.3 utilizado pelos microcomputadores que seguem a linha APPLE II.

O comando RENAME muda o nome do arquivo nomearq1, para o nome indicado pelo nomearq2.

Pode-se trocar o nome de qualquer arquivo, armazenado em um disquete, com o comando RENAME.

RENAME funciona para qualquer arquivo de disquete, independente do tipo de BASIC que está sendo usado.

EXEMPLO

RENAME FATURA,FATURAMENTO

nomearq1 é o nome de um arquivo existente e nomearq2 é o novo nome que deve ser único.

Se nomearq1 não existe, aparecerá uma mensagem de erro. O arquivo deve estar desprotegido.

RENAME pode trocar o nome do arquivo por outro que já exista dentro do disquete, e pode fazê-lo muitas vezes. Por isso deve-se ter cuidado para não escolher para um arquivo um nome que já exista no disquete. Se existir, ficarão dois arquivos com o mesmo nome. Isso pode causar muita confusão.

Não se pode mudar o nome de um arquivo protegido. Pode-se especificar o drive, slot e volume, em qualquer ordem, se desejado.

[,Dn] [,Sn] [,Vn]

Se não for utilizado na sintaxe, o microcomputador assumirá aquilo que já tiver sido acionado, ou ativado anteriormente.

RENAME é um comando DOS, requer PRINT e CHR$(4) no modo programado.

VEJA TAMBÉM

OPEN nomearq

Comando **RENUMBER**

SINTAXE GERAL

RENUMBER

RENUMBER renumera as linhas de um programa.
Os números de linha utilizados em GOTO, GOSUB, IF-THEN, ON-GOTO e ON-GOSUB são alternados de acordo, a fim de se manter o mesmo esquema de ramificação.
Se não for determinado o número no comando RENUMBER, o computador renumera automaticamente cada linha do programa, espaçando as linhas de 10 em 10.

```
EXEMPLO 1
100 REM USANDO O COMANDO RENUMBER
110 K = 1
120 PRINT "SE CADA LINHA DO PROGRAMA ";
130 GOTO 170
140 PRINT "O COMANDO RENUMBER";
150 K = 2
160 GOTO 180
170 PRINT "FOR RENUMERADA"
180 ON K GOTO 140,190
190 PRINT "FOI ACEITO NO SEU MICROCOMPUTADOR"
200 END
```

RUN (rode) para ter certeza de que funciona.
Teclar o comando RENUMBER e RUN de novo.

```
EXECUÇÃO DO PROGRAMA
SE CADA LINHA DO PROGRAMA
FOR RENUMERADA
O COMANDO RENUMBER FOI ACEITO NO SEU MICROCOMPUTADOR
```

Após a execução do comando RENUMBER, as linhas de programa estarão renumeradas; dê um LIST para confirmar.

```
10 REM USANDO RENUMBER
20 K = 1
30 PRINT "SE CADA LINHA DO PROGRAMA ";
40 GOTO 80
50 PRINT "O COMANDO RENUMBER ";
60 K = 2
70 GOTO 90
80 PRINT "FOR RENUMERADA"
90 ON K GOTO 50,100
100 PRINT "FOI ACEITO NO SEU MICROCOMPUTADOR"
110 END
```

Algumas versões BASIC apresentam um comando de renumeração de linhas com maiores facilidades.
RENUMBER 10,10
renumera cada linha do programa a começar com a linha mais baixa, com incremento de 10.

A renumeração das linhas pode ser feita de diversas maneiras; com intervalos maiores do que 10, e iniciando em determinada linha.

RENUMBER 80,20

Podemos também fixar somente o valor inicial onde desejamos que o nosso programa seja renumerado, e deixar o intervalo obedecer a especificação implícita do sistema.

RENUMBER 30

Estabelece o valor inicial na linha 30, sendo o intervalo de 10, como especifica o sistema.

VARIAÇÕES DE USO

O computador LABO usa RENUM para renumerar as linhas do programa.

SINTAXE GERAL

RENUM [[novo número] [[,número antigo] [incremento]]]

Novo número é o primeiro número de linha a ser usado na nova seqüência. Por omissão é 10.

Número antigo é a linha do programa atual onde deve começar a renumeração. Por omissão é a primeira linha do programa.

Incremento é o incremento a ser usado na nova seqüência. Por omissão é 10.

RENUM muda também todas as referências a números de linha que seguirem os comandos GOTO, GOSUB, THEN, ON...GOTO, ON...GOSUB e ERL para refletir os novos números de linha

EXEMPLOS:

RENUM

Renumera o programa inteiro. Primeira linha será 10, com incremento de 10 em 10.

RENUM 500,,50

Renumera o programa inteiro. O primeiro novo número de linha será 500, sendo o incremento 50.

RENUM 800,700,30

Renumera as linhas de 700 em diante para que comecem com o número de linha 800 e com incremento de 30.

RENUMBER COMANDO/DOS

O programa RENUMBER é um programa na versão BASIC APPLESOFT de microcomputadores que seguem a linha APPLE para renumerar as linhas de um programa básico ou fundir dois programas.

Use o programa RENUMBER para renumerar todas ou algumas das declarações do seu programa APPLESOFT base, para fundir as declarações de dois de seus programas, ou inserir uma sub-rotina dentro do seu programa de uma sub-rotina de arquivo.

O programa RENUMBER está no disco master system.

Sintaxe do comando RENUMBER

o primeiro carácter da linha de um comando RENUMBER é um (&). A linha de comando mais curta é: (&) RETURN.

Quando você dá esse comando, renumera o programa que está na memória, começando com o primeiro programa e terminando com o último programa de declaração: chamado de 10 o primeiro programa as declarações do incremento são de 10 em 10. Use a vírgula para separar os comandos quando houver mais que um comando por linha.

EXEMPLO

&:S 50, E 100, F 500

Diz a RENUMBER para processar linhas de 50 até 100 de um

programa e dá a linha da primeira declaração o número 500.

VARIAÇÕES DE USO
Não se tem conhecimento de nenhuma.

VEJA TAMBÉM
AUTO, GOTO, GOSUB, IF-THEN, ON-GOSUB, ON GOTO, LIST

Instrução/Comando **RESET**

SINTAXE GERAL

RESET (X,Y)

A instrução RESET é utilizada pelos microcomputadores que seguem a linha TRS-80 para apagar um ponto gráfico plotado (aceso) na linha X coluna Y.

RESET pode ser utilizado tanto como instrução no modo programa, como comando no modo direto.

Para acender o ponto luminoso SET(X,Y).

EX:

SET (64,23)

Este comando acende um ponto no meio da tela do vídeo na coordenada horizontal representada por X e na coordenada vertical representada por Y.

RESET (64,23)

simplesmente apaga o ponto que foi aceso como SET(64,23)

O bloco a ser "DESLIGADO" dentro da matriz é especificado pelas coordenadas X, Y, entre parênteses, que segue a instrução RESET.

A tela do vídeo para fins gráficos ficará dividida em 128 pontos horizontais representados por (X) e serão numerados de 0 a 127 da esquerda para a direita.

Os pontos verticais são numerados de 0 a 47 de cima para baixo e são representados por (Y).

EXEMPLO 1

```
100 REM USANDO RESET
110 CLS
120 X = 10
130 FOR Y = 1 TO 100
140 SET (Y,X)
150 NEXT Y
160 PRINT
170 PRINT "RESET FOI APROVADO SE A LINHA ACESA DESAPARECEU"
180 FOR Y = 1 TO 100
190 RESET (Y,X)
200 NEXT Y
210 END
```

EXECUÇÃO DO PROGRAMA

```
RUN
RESET FOI APROVADO SE A LINHA ACESA DESAPARECEU
```

ORTOGRAFIA ALTERNATIVA

R é utilizado em BASIC nível 1 em algumas versões BASIC como breviatura de RESET. Nos microcomputadores que seguem a linha PPLE usa-se PLOT(X,Y) e nos que seguem a linha SINCLAIR NPLOT(X,Y). Nos micros que seguem a linha APPLE, o PLOT serve anto para acender um ponto apagado, como para apagar um ponto nteriormente aceso.

Nos microcomputadores da linha TRS-80 COLOR é necessário sar mais um parâmetro (K) para representar a cor SET(X,Y,K)

devendo este representar um número inteiro de 0 a 8.

VARIAÇÓES DE USO
Não se tem conhecimento de nenhuma.

VEJA TAMBÉM
SET, CLS, POINT, CLRDOT

Instrução **RESTORE • REST. • RES**

SINTAXE GERAL

Número de linha RESTORE

RESTORE é uma instrução que funciona em conjunto com o par de instruções de entrada READ...DATA...

RESTORE deve ser utilizada no modo programa.

Esta instrução funciona de maneira idêntica nas versões BASIC em que é aceita.

A instrução RESTORE faz com que o apontador de dados (pointer) de uma relação de valores de instruções DATA vá para o início das linhas de dados (DATA) do programa.

A instrução RESTORE faz com que o próximo dado lido pela instrução READ seguinte seja o primeiro valor da primeira instrução DATA do programa.

Isto permite ao computador utilizar dados armazenados em instruções DATA mais de uma vez.

EXEMPLO 1

```
100 REM USANDO A INSTRUCAO RESTORE
110 READ A
120 IF A = 3 THEN 140
130 GOTO 110
140 RESTORE
150 READ A
160 IF A = 1 THEN 190
170 PRINT "RESTORE NAO APROVOU"
180 GOTO 210
190 PRINT "RESTORE APROVOU"
200 DATA 1, 2, 3
210 END
```

EXECUÇÃO DO PROGRAMA

```
RUN
RESTORE APROVOU
```

ORTOGRAFIA ALTERNATIVA

Alguns microcomputadores, que seguem o TRS-80 nível 1, permitem abreviar RESTORE a REST. Outros, micros americanos e europeus, aceitam RES como forma abreviada de RESTORE.

EXEMPLO 2

```
200 REM USANDO RESTORE
210 DATA 5,10
220 DATA 20,30
230 DATA 40,50
240 READ A,B,C
250 RESTORE
260 READ E,F
270 RESTORE
280 READ G
```

```
290 PRINT A,B,C,E,F,G
300 END
```

EXECUÇÃO DO PROGRAMA

```
RUN
5,10,20,5,10,5
```

Existem certos microcomputadores, de linhas americanas, que podem restaurar o apontador de dados numéricos e dados de string separadamente. RESTORE$, ao ser inserido num programa, restaura unicamente os dados de string. RESTORE$ é utilizado quando devem ser utilizados novos valores numéricos.

```
EXEMPLO 3
200 REM USANDO RESTORE$
210 READ A$
220 READ R
230 RESTORE$
240 READ P$
250 READ R
260 IF R = 2 THEN 290
270 PRINT "NAO APROVOU"
280 GOTO 310
290 PRINT "RESTORE$ APROVOU"
300 DATA EXEMPLO 1, 2
310 END
```

EXECUÇÃO DO PROGRAMA

```
RUN
RESTORE$ APROVOU
```

VEJA TAMBÉM

DATA, READ

Instrução **RESUME**

SINTAXE GERAL

Número de linha RESUME

RESUME é uma instrução utilizada somente no modo programa.
O computador não permite a execução da instrução RESUME se não estiver precedida por uma instrução ON ERROR GOTO.
RESUME é aceita na versão BASIC dos microcomputadores que seguem a linha APPLE, mas funciona de maneira idêntica nas versões BASIC em que é aceita.
RESUME é utilizada como instrução no final de uma rotina (ONERR GOTO) ou (ON ERROR GOTO) para contornar erros fazendo com que a execução do programa recomece a partir do ponto onde ocorre o erro do usuário. Todas as variáveis anteriores continuam na memória, exceto aquelas que possam ter interferido com a execução do erro.
Veja ON-ERROR-GOTO para um programa de exemplo que utiliza RESUME (número de linha).
Esta instrução não é disponível na versão Integer BASIC.

RESUME NEXT é utilizada para bifurcar para a linha que segue o erro e CONTinua a execução do programa.

```
EXEMPLO 1
100 REM USANDO A INSTRUCAO RESUME
110 ONERR GOTO 170
120 PRINT "DE ENTRADA NUM NUMERO POSITIVO";
130 INPUT X
140 B = LOG(X)
150 PRINT "O LOG DE ";X;" E ";B
160 GOTO 120
170 PRINT "NAO E PERMITIDO NUMERO NEGATIVO"
180 X = X*-1
190 RESUME 0
200 END

EXECUÇÃO DO PROGRAMA (utilizando-se -7)
RUN
DE ENTRADA NUM NUMERO POSITIVO? -7
NAO E PERMITIDO NUMERO NEGATIVO
O LOG DE 7 E 1.94591015
DE ENTRADA NUM NUMERO POSITIVO?5
O LOG DE 5 E 1.60943791

VARIAÇÕES DE USO
Não se tem conhecimento de nenhuma.

VEJA TAMBÉM
ON-ERROR-GOTO, ERL
```

Instrução **RETURN • RET • R.**

SINTAXE GERAL

Número da linha RETURN

A instrução RETURN é utilizada em conjunto com a instrução GOSUB.

Suas funções são idênticas na maioria das versões BASIC.

RETURN como última instrução de uma sub-rotina completa a execução da instrução GOSUB, instruindo o microcomputador para retornar ao programa principal continuando a execução a partir da próxima instrução após o GOSUB.

Como última instrução numa sub-rotina, manda o computador devolver o controle para a linha que contém a instrução GOSUB, e continuar a execução do programa a partir desse ponto.

O computador não permitirá a execução da instrução RETURN se não estiver precedida de uma instrução GOSUB.

EXEMPLO 1

```
100 REM USANDO A INSTRUCAO RETURN
110 PRINT "EXECUTANDO PROGRAMA PRINCIPAL"
120 GOSUB 1000
130 PRINT "RETORNEI AO PROGRAMA PRINCIPAL"
140 PRINT "RETURN FOI APROVADO"
150 END

1000 PRINT "EXECUTANDO A SUB-ROTINA"
1010 PRINT "RETORNANDO AO PROGRAMA PRINCIPAL"
1020 RETURN
```

EXECUÇÃO DO PROGRAMA

```
RUN
EXECUTANDO PROGRAMA PRINCIPAL
EXECUTANDO A SUB-ROTINA
RETORNANDO AO PROGRAMA PRINCIPAL
RETORNEI AO PROGRAMA PRINCIPAL
RETURN FOI APROVADO
```

ORTOGRAFIAS ALTERNATIVAS

Utilizam-se diversas abreviaturas, inclusive RET. (TRS-80 Nível 1), RET (Dec. PDP-SE) e R. (Acorn ATOM).

VARIAÇÕES DE USO

Nos micros que seguem a linha APPLE II a instrução POP faz o programa esquecer o endereço do retorno do GOSUB mais recente. Uma instrução RETURN após uma instrução POP faz voltar para o endereço seguinte ao segundo mais recente GOSUB.

VEJA TAMBÉM

GOSUB, ON-GOSUB, IF-GOSUB, GOSUB-OF

Função **RIGHT$**

SINTAXE GERAL

Número de linha PRINT RIGHT$(X$,N)
ou
Número de linha PRINT RIGHT$("string",N)

RIGHT$ é uma função usada normalmente no modo programa e geralmente precedida de uma instrução LET ou PRINT.

Tem funções idênticas na maioria das versões BASIC em que é aceita.

A função RIGHT$(X$,N) fará com que o microcomputador leia a string designada (X$) a partir da extrema direita até o carácter que se encontre em "N" lugares dimensionados a partir da direita.

Quaisquer ações (PRINT ou LET) serão realizadas até o último carácter N da string.

A função RIGHT$(string) é utilizada para isolar um número específico(n) de caracteres de string, a contar do carácter mais para a direita.

PRINT RIGHT$("SULLIVAN",4), por exemplo, imprime as letras IVAN, que são os quatro caracteres da direita em SULLIVAN, que é string.

O string deve estar entre aspas ou deve ser atribuído a uma variável de string (variável alfanumérica).

O valor N deve ser um inteiro entre 0 e 255 e X$ (variável alfanumérica ou valor alfanumérico) não pode ter mais de 255 caracteres.

Se N for maior ou igual à quantidade de caracteres de X$ o string inteiro é retornado.

Esta função não é disponível nas versões Integer BASIC em microcomputadores que seguem a linha APPLE.

O número (n) de caracteres pode ser expresso como variável, número ou operação aritmética, sendo separado o string do número por uma vírgula.

Sendo decimal o valor de (n), o computador encontra automaticamente o valor de número inteiro

EXEMPLO 1

```
100 REM USANDO A FUNCAO RIGHT$
110 X$="REAPROVADA"
120 Y$=RIGHT$(X$,8)
130 PRINT "A FUNCAO RIGHT$ FOI ";Y$
140 END
```

EXECUÇÃO DO PROGRAMA

```
A FUNCAO RIGHT$ FOI APROVADA
```

ORTOGRAFIA ALTERNATIVA

A função RIGHT é utilizada em alguns microcomputadores que utilizam versão MAX BASIC.

VARIAÇÕES DE USO

Alguns BASIC utilizam RIGHT$(X$,N) para isolar o substring de X$, que começa na posição N. Inclui-se tudo que estiver à

direita desse ponto. RIGHT$ ("ROSSANA",3), por exemplo, devolve ANA, uma vez que A está na terceira posição.

VEJA TAMBÉM

PRINT, LEFT$, MID$, CHR$, SPACE$, STR$, STRING$, INKEY$, INSTR, SEG$

Função RND • RAND
RANDON • RANDOMIZE

SINTAXE GERAL

Número de linha PRINT RND(X)
ou
Número de linha LET Y = RND(X)

RND (X) é uma função podendo ser utilizada tanto no modo programa quanto no modo imediato devendo, entretanto, ser precedido de uma instrução geralmente LET ou PRINT.

RND (X) é utilizado em quase todos os microcomputadores para gerar números randômicos (aleatórios). Na maioria dos microcomputadores e nos que seguem a linha APPLE RND gera um número real maior ou igual a 0 (zero) e menor que 1 (versão BASIC da Microsoft).

Dará um valor entre 0 e 1 (.00000001 até .99999999) apenas, não importando o valor dado a (X). Esta função é usada quando se deseja atribuir ao computador a escolha arbitrária de um valor qualquer que interfere com a execução de uma instrução preestabelecida.

Por exemplo, a geração de índices, números ou gráficos abstratos na tela.

A função RND(0) reporta-se ao número mais recentemente gerado entre 0 e 1.

Nos microcomputadores que seguem a linha TRS-80 RND(X) gera números aleatórios inteiros e positivos que variam de 1 a X.

Nestes microcomputadores a instrução RANDON assegura que o número gerado ao se executar a função RND seja realmente aleatória. Nos microcomputadores que seguem a linha SINCLAIR a instrução RND é que assegura a geração aleatória para RND(X).

A função RND(0) nos microcomputadores que seguem a linha TRS-80 gera um número aleatório entre 0 e 1 assim como no BASIC Microsoft. Neste último, a mesma seqüência de número é gerada toda vez que se volta a executar o programa através de RUN, desde que não seja encontrada antes a instrução RANDOMIZE. Nos micros da linha TRS-80 havendo ou não a instrução RANDON, seqüências diferentes serão geradas.

RND(X) onde X <0 não é aceito.

RND(X) onde X >0 gera números aleatórios inteiros e positivos entre 1 e X.

Já nos microcomputadores que aceitam a versão BASIC da Microsoft, RND(X) onde X >0 indica que o próximo número randômico (aleatório) da seqüência inicializada pela instrução RANDOMIZE deve ser gerado, sempre entre 0 e 1.

Poucos cumprem a regra ANSI segundo a qual RND deve ser utilizado sozinho, sem (X). As normas ANSI requerem também que a seqüência de números criada se repita cada vez que se utiliza RND (para fins de depuração), a não ser que a palavra BASIC RANDOMIZE seja incluída no programa, para "redistribuir os números".

Certos microcomputadores que seguem as linhas APPLE e TRS-80 Nível 1 têm um BASIC de números inteiros limitado. Utilizam

RND(X) para gerar números inteiros entre certos valores mínimos e máximos da máquina utilizada, tipicamente -32768 e +32767.

De maneira habitual, cada vez que se tecle RND, o computador irá devolver um número diferente. Emprega-se comumente o valor de X para especificar como irá funcionar a função RND.

Se for dado a X um valor negativo, por exemplo, isto poderá "realimentar o gerador de números aleatórios. Em outras palavras, estará mandando que uma nova seqüência de números seja criada cada vez que se altere o número negativo. Sendo repetido o mesmo número negativo, a mesma seqüência de números será repetida. Obrigar essa repetição é prática valiosa para detecção de programa.

Outros computadores utilizam X negativo para fazer com que o valor se repita. Nesse caso, X deve ser positivo para que RND opere "normalmente".

Se o valor de X for igual ou superior a 1, alguns microcomputadores (tais como o Sinclair, ZX80 e o TRS-80) devolvem um número inteiro positivo entre 1 e o valor de X.

Seu manual de referência de computador será a melhor fonte de informações a respeito da maneira pela qual seu computador em particular pode utilizar RND.

```
EXEMPLO 1
para micros que seguem versão BASIC da Microsoft
100 REM USANDO A FUNCAO RND
110 FOR A = 1 TO 5
120 PRINT RND (A)
130 NEXT A
140 END

EXECUÇÃO DO PROGRAMA
RUN
0.131137465
0.80924873
0.846447204
0.841536558
0.591965711
```

RND(0) é utilizado por alguns computadores, a fim de especificar a mesma operação de RND.

```
EXEMPLO 2
200 REM PROGRAMA TESTE RND (0)
210 FOR A = 1 TO 5
220 PRINT RND(0)
230 NEXT A
240 END

EXECUÇÃO DO PROGRAMA
RUN
0.591965711         0.591965711
0.591965711         0.591965711
0.591965711
```

Embora RND(X) gere números aleatórios, em determinados microcomputadores RND(0) repete o último número pelo gerador de números aleatórios.

```
EXEMPLO 3
300 REM PROGRAMA DE TESTE RND(0) COMO REPETIDOR
310 PRINT "RND(1)"
320 FOR A = 1 TO 4
330 PRINT RND(1)
340 NEXT A
350 PRINT "RND(0)"
360 FOR B = 1 TO 5
370 PRINT RND (0)
380 NEXT B
190 END

EXECUÇÃO DO PROGRAMA
RUN
RND(1)
0.26800113
0.419217095
0.878831482
0.368373372
0.123235316

RND(0)
0.123235316
0.123235316
0.123235316
0.123235316
0.123235316
```

Alguns computadores criam um número aleatório entre 1 e o valor de X, se X for maior do que 1 (nos micros que seguem a linha TRS-80).

RND(X) converte o valor de X automaticamente num número inteiro.

```
EXEMPLO 4
400 REM PROGRAMA DE TESTE RND
410 A = 100
420 FOR B = 1 TO 5
430 PRINT RND(A)
440 NEXT B
150 END

EXECUÇÃO DO PROGRAMA
RUN
15
81
91
97
53
```

UM TRUQUE

Se o seu computador for daqueles que geram números aleatórios 0 e 1, e precisar um número inteiro aleatório entre 1 e 10, experimente este truque:

```
PRINT INT(RND(1)*10)+1
```

Um número aleatório entre 1 e 10 pode ser impresso mediante

este truque! PRINT INT(10!RND+1)

A forma geral para se gerarem números inteiros aleatórios entre A e B é :

INT(RND*(B-A+1)+A)

Nos microcomputadores que seguem a linha SINCLAIR,RND e RAND: estão ambas na mesma tecla, mas enquanto RND é uma função, RAND é uma instrução.

A instrução RAND é utilizada para controlar o grau de casualidade de RND.

RND não é verdadeiramente aleatório, segue sim uma seqüência fixa de 65536, números que estão misturados de modo a parecerem casuais.

Pode utilizar-se RAND para inicializar o RND num lugar definido nesta seqüência escrevendo RAND e depois um número entre 1 e 65535 e depois NEWLINE (ou ENTER).

EXEMPLO

```
10 RAND 1
20 PRINT RND
```

tente substituir também a linha 20 por

```
20 PRINT INT (RND*10)+1
```

Comando **ROT=**

SINTAXE GERAL

ROT = exprnm

ROT = exprnm pode ser utilizado como comando no modo imediato ou como instrução no modo programa nos microcomputadores que seguem a linha APPLE II na versão BASIÇ APPLESOFT.

ROT = exprnm altera a orientação do desenho da forma gráfica de alta resolução executada por DRAW ou XDRAW.

O comando ou instrução ROT estabelece rotação do padrão para DRAW e XDRAW. Roda a forma em relação ao centro da tela (nos eixos X e Y). Posiciona uma rotação angular para a forma a ser desenhada por DRAW e XDRAW.

EXEMPLOS:

ROT = 0 desenha a forma na orientação que foi definida.

ROT = 16 coloca o ângulo de rotação de forma em 90 graus no sentido horário.

ROT = 32 coloca o ângulo de rotação de forma em 180 graus no sentido horário.

O valor de ROT varia de 0 a 255, apesar de haver apenas 64 possibilidades de rotação, de 0 a 63.

Uma instrução SCALE = exprnm determina o tamanho da forma gráfica de alta resolução desenhada por DRAW ou XDRAW.

Quando SCALE é colocado em 1, existem quatro formas possíveis de ROT, são elas:

0 = 0 graus
16 = 90 graus
32 = 180 graus
48 = 270 graus

Quando SCALE é igual a 2, ficam existindo oito valores possíveis para ROT =, quando SCALE = 3, existem 16 valores, e assim sucessivamente, até o máximo de 64 valores diferentes.

Todas as 64 possibilidades de rotação ficam disponíveis ao se usar SCALE = 5 ou maior.

Um valor não reconhecível de ROT = causa na forma a ser desenhada a orientação do próximo menor (usualmente) valor reconhecido de rotação.

exprnm deve ter um valor entre 0 e 255 ou uma mensagem de erro aparecerá.

VARIAÇÕES DE USO

ROT = não será reconhecido como comando ou instrução, a menos que o caracter "=" seja o primeiro caracter a seguir diferente de branco (espaço).

EXEMPLO

ROT = exprnm

Este comando ou instrução não é disponível na versão INTEGER BASIC para microcomputadores da linha APPLE II.

VEJA TAMBÉM

SCALE

Comando/Instrução **RUN • RU • R.**

SINTAXE GERAL

RUN

RUN é um comando de manutenção geralmente utilizado no modo direto, podendo também ser usado como instrução no modo programa.

O comando RUN zera as variáveis e executa o(s) programa(s) armazenado(s) na memória, a começar com o número de linha mais baixo. Em muitos computadores, um número de linha pode ser incluído após o comando RUN para especificar uma linha inicial outra que não a primeira (como, por exemplo, RUN 130).

EXEMPLO 1

```
100 REM USANDO RUN
110 PRINT "ESTA IMPRESSAO TEVE INICIO NA LINHA 110"
120 GOTO 140
130 PRINT "ESTA IMPRESSAO TEVE INICIO NA LINHA 130"
140 END
```

EXECUÇÃO DO PROGRAMA

Depois de se dar entrada no comando RUN, o microcomputador deve imprimir a mensagem:

ESTA IMPRESSAO TEVE INICIO NA LINHA 110.

Acrescentando o número 130 ao comando RUN, RUN130 ou RUN 130, o computador deverá começar a execução na linha 130, imprimindo a seguinte mensagem:

ESTA IMPRESSAO TEVE INICIO NA LINHA 130

Quando estiver usando variáveis numéricas e se deseja recomeçar a execução do programa sem zerar todas as variáveis e strings deve-se usar o comando GOTO.

Embora a maioria dos microcomputadores utilizem RUN estritamente como comando a nível de denominação, alguns aceitam RUN como instrução de programa. Os computadores que utilizam BASIC Microsoft de disquete aceitam RUN como forma da instrução CHAIN. (Ver CHAIN). Por exemplo, 5780 RUN "PROG:'" carrega do disquete um programa denominado PROG, e passsa a RUN (rodar) o mesmo.

ORTOGRAFIAS ALTERNATIVAS

Alguns computadores utilizam a abreviatura RU. Outros aceitam R. e os micros que seguem a linha SINCLAIR possuem o comando na própria tecla.

VEJA TAMBÉM

CHAIN

SAVE • SCALE • SCRN
SET • S. • SGN • SHLOAD
SIN • SIND • SING • SLOW
SPC • SPEED • SQR • SQRT
STEP • STE • ST • S.
STOP • STO • STORE
STRING$ • STRING • STR
STR $ • SYSTEM • SYS

Comando **SAVE**

SINTAXE GERAL EM CASSETE
SAVE

SINTAXE GERAL EM DISQUETE
SAVE nomearq

SAVE é um comando geralmente utilizado no modo direto.
SAVE é utilizado principalmente nos microcomputadores que seguem a linha APPLE II e alguns outros microcomputadores americanos e europeus para gravar o programa contido na memória RAM em fita cassete ou disquete.

EXEMPLO 1
```
100 REM USANDO O COMANDO SAVE
110 PRINT "ESTE PROGRAMA CONFIRMA O COMANDO SAVE"
120 END
```

Ligue o gravador de cassete no modo de gravar e tecle o comando SAVE.
Os microcomputadores que seguem a linha APPLE apitam quando inicia a gravação do programa e também quando esta termina. O último apito dá indicação para a parada manual do gravador.
Depois que o programa estiver gravado em fita cassete, tecle NEW a fim de apagar o programa da memória. Carregue o programa da fita magnética de volta para o microcomputador (utilize a instrução LOAD). Liste o programa para se certificar de que o programa armazenado na memória do microcomputador é idêntico daquele que foi lançado inicialmente no microcomputador (veja a instrução LIST).

Amostra do Programa

```
ESTE PROGRAMA TESTA A CARACTERISTICA SAVE
```

VARIAÇÕES DE USO
Em microcomputadores que seguem a linha TRS-80 como CP-500 DGT 100 DGT 1000 e outros o comando equivalente é o CSAVE.
Na versão Integer BASIC, SAVE só pode ser usado no modo direto.
Alguns computadores com capacidade de armazenamento em disquete utilizam a instrução SAVE para copiar programas da memória do computador para a memória em disquete. É preciso ter um nome de arquivo.
Exemplo:

SAVE EXEMPLO 1

SINTAXE EM DISQUETE
SAVE nomearq(,Dn)(,Sn)(,Vn)
Este é um comando do DOS sistema operacional em disquete.
Não existindo arquivo com o nomearq no disquete será criado um com este nome e armazenado no disquete na linguagem

programada. Existindo um arquivo com o mesmo nome e na mesma linguagem do programa, o conteúdo daquele arquivo será zerado e o programa corrente será armazenado no seu lugar.

Se o programa a ser armazenado já existir com o mesmo nome porém em linguagem diferente ou com tipo diferente, será impressa no vídeo a mensagem de erro "FILE TYPE MISMATCH".

Dn,Sn e Vn representam drive número, slot número e volume número; podem ser especificados em qualquer ordem ou mesmo omitidos; neste caso Dn e Sn irão se referir ao drive e slot ativado por último e V0 é usado por ausência de Vn.

Este comando quando utilizado no modo programa deve ser precedido de PRINT CHR$(4).

VEJA TAMBÉM
LOAD, CLOAD, LIST

Comando/Instrução **SCALE**

SINTAXE GERAL

SCALE = exprnm

SCALE = exprnm pode ser utilizado como comando no modo direto ou como instrução no modo programa. É reconhecido pelos microcomputadores que seguem a linha APPLE na versão BASIC APPLESOFT para estabelecer o tamanho da forma gráfica de alta resolução desenhada pelas instruções DRAW ou XDRAW.

SCALE = exprnm X estabelece escala (1 a 255) e dá o tamanho da forma gráfica de alta resolução desenhada por DRAW ou XDRAW.

Pode-se sempre usar SCALE antes de desenhar uma forma pela primeira vez num programa.

O tamanho da forma da tabela é multiplicado pelo valor de exprnm.

EX: SCALE = 1

Manda manter a escala de um ponto para cada vetor de desenho.

SCALE = 2

Desenha dois pontos para cada vetor de desenho.

A máxima escala que se pode determinar é SCALE = 0 que coloca 256 pontos para cada vetor do desenho.

O valor da exprnm deve ser um inteiro entre 0 e 255, ou uma mensagem de erro aparecerá.

VARIAÇÕES DE USO

SCALE é uma palavra reservada do APPLESOFT, não disponível na versão Integer BASIC.

SCALE não é reconhecida como palavra reservada a menos que o caracter "=" seja o primeiro a seguir diferente de branco (espaço).

VEJA TAMBÉM

ROT

Função **SCRN**

SINTAXE GERAL

SCRN (col,lin)

SCRN (col,lin) é uma função usada normalmente no modo programa devendo ser precedida por uma instrução geralmente LET, PRINT ou IF...THEN...

Esta função é aceita nos microcomputadores que seguem a linha APPLE.

Retorna o código da cor do ponto gráfico de baixa resolução com as coordenadas especificadas,

SCRN é utilizado como característica especial para indicar a cor do ponto gráfico na tela. O computador tem capacidade de exibir 16 cores (numeradas de 0 até 15). A listagem completa das cores consta da instrução COLOR.

O ponto de gráficos é especificado pelas coordenadas X, Y, entre parenteses, após a instrução SCRIN.

A declaração:

X = SCRN(10,22)

atribui à variável X o número da cor na coordenada entre parênteses (nesse caso, linha 11 e coluna 23). A cor devolvida para a variável está entre 0 e 15.

Esta declaração pode ser muito útil ao se escrever programas avançados em baixa resolução.

Esses valores podem apresentar-se de 0 até 39 para linha e de 0 até 47, para coluna.

Se col está na faixa de 0-39, SCRN retorna o código da cor do ponto gráfico(col,lin). Se col está na faixa 0-47 e lin na faixa 0-31, SCRN devolve o ponto, o número da cor do ponto gráfico (col-40,lin+16).

Se col está na faixa 40-47 e lin na faixa de 32-47, SCRN devolve um número sem relação com coisa alguma na tela.

Se SCRN é usado durante o modo gráfico de alta resolução, o número devolvido está relacionado com área de memória gráfica de baixa resolução em vez de alta resolução.

```
EXEMPLO 1
100 REM USANDO A FUNCAO SCRN
110 GR
120 COLOR = 13
130 PLOT 25, 15
140 IF SCRN(25,15)THEN 170
150 PRINT "A FUNCAO SCRN NAO FOI ACEITA"
160 GOTO 180
170 PRINT "A FUNCAO SCRIN FOI ACEITA"
180 END

EXECUÇÃO DO PROGRAMA
RUN
A FUNCÃO SCRN FOI ACEITA

EXEMPLO 2
200 REM USANDO A FUNCAO SCRN
```

```
210 GR
220 COLOR 14
230 PLOT 12,12
240 PRINT SCRN(12,12)
```

Executando o programa o computador da linha APPLE respondera 14.

VARIAÇÕES DE USO

SCRN(col,lin) é usado durante o modo gráfico de alta resolução; o número devolvido está relacionado com a área de memória gráfica de baixa resolução em vez de alta resolução.

SCRN só será reconhecida como função se o primeiro carácter diferente do espaço a sua frente for parênteses.

VEJA TAMBÉM

COLOR, PLOT, GR, POINT

Instrução **SET • S.**

SINTAXE GERAL

Número de linha SET (X,Y)
ou para micros coloridos
Número de linha SET (X,Y,K)

SET (X,Y) pode ser usado como comando no modo direto ou como instrução no modo programa para ligar um ponto gráfico na tela de micros da linha TRS-80. Nos da linha APPLE e SINCLAIR a instrução equivalente é o PLOT (X,Y).

A instrução SET é utilizada pelos micros que seguem a linha TRS-80 para acender um ponto luminoso em um ponto predeterminado da tela.

SET (X,Y) liga o bloco de gráficos especificados pelas coordenadas X e Y.

A tela de vídeo é dividida em uma grade em que a coordenada X é numerada da esquerda para a direita de 0 a 127 e a coordenada Y é numerada de cima para baixo de 0 a 47.

Portanto, o ponto (0,0) está no extremo superior esquerdo do vídeo, enquanto que o ponto (127,47) está no canto extremo inferior direito.

Os argumentos X e Y podem ser constantes numéricas, variáveis ou expressões numéricas.

Eles não precisam ser números inteiros porque SET (X,Y) usa a porção INTeira de X e Y.

SET (X,Y) é válido para:

0 <= X < 128
0 <= Y < 48

Para desligar o bloco de gráficos, veja a instrução RESET.

EXEMPLO 1

```
100 REM USANDO A INSTRUCAO SET
110 SET (RND(128)-1, RND(48)-1)
120 PRINT "SET FOI ACEITO"
130 PRINT "SE TIVER APARECIDO UM PONTO LUMINOSO NA TELA"
180 END
```

EXECUÇÃO DO PROGRAMA

```
RUN
SET FOI ACEITO
SE TIVER APARECIDO UM PONTO LUMINOSO NA TELA
```

Depois modifique a linha 110 para:

```
110 INPUT X,Y : SET (X,Y)
```

e execute o programa para ver onde os pontos acesos estão localizados.

```
RUN
? 60
?? 37
SET FOI ACEITO
SE TIVER APARECIDO UM PONTO LUMINOSO NA TELA
```

ORTOGRAFIA ALTERNATIVA

Alguns microcomputadores que seguem a linha TRS-80 nível 1 aceitam S. a título de abreviatura.

VEJA TAMBÉM

RESET

Função **SGN**

SINTAXE GERAL

Número de linha PRINT SGN (X)

SGN (X) é uma função normalmente usada no modo direto ou no modo programa sendo precedida por uma instrução geralmente PRINT ou LET.

SGN (X) nos indica se X é um número ou expressão numérica,cujo resultado seja positivo, negativo ou igual a zero. Para isto a função SGN (X) retornará:

+1 se X >0
-1 se X <0
 0 se X =0

Por exemplo, PRINT SGN (18), SGN (-40), SGN (0) vai imprimir:

```
          1        -1          0
```

EXEMPLO 1

```
100 REM USANDO A FUNCAO SGN
110 X = 40
120 Y = SGN(X)
130 IF Y = 1 THEN 160
140 PRINT "SGN NAO FOI ACEITO"
150 GOTO 170
160 PRINT "SGN FOI ACEITO"
170 END
```

EXECUÇÃO DO PROGRAMA

```
RUN
SGN FOI ACEITO
```

SE O SEU COMPUTADOR NÃO TIVER ESSA FUNÇÃO

Se o seu computador falhou no teste SGN, use a seguinte sub-rotina:

```
999 GOTO 170
1000 REM SUB-ROTINA DA FUNCAO SGN INPUT X, OUTPUT Y
1010 Y = 0
1020 IF X = 0 THEN 1060
1030 Y = 1
1040 IF X>0 THEN 1060
1050 Y = -1
1060 RETURN
```

Mude a linha do exemplo 1 para

```
120 GOSUB 1000
```

VARIAÇÕES DE USO

Não se tem conhecimento de nenhuma.

VEJA TAMBÉM

ABS

Comando **SHLOAD**

SINTAXE GERAL

SHLOAD

SHLOAD é um comando reconhecido pelos microcomputadores que seguem a linha APPLE na versão do BASIC APPLESOFT que quando executado carrega uma forma gráfica de alta resolução de uma fita cassete.

Carrega um padrão de tabela de forma da fita do gravador.

É um comando do APPLESOFT que opera exclusivamente com fitas; carrega uma forma gráfica de alta resolução da fita cassete. Não disponível em Integer BASIC.

Antes de gravar a tabela de formas na fita, deve-se calcular seu comprimento (em bytes, hexadecimal).

Após guardar tabelas de forma em fita, o APPLESOFT pode ler de volta para a memória. Primeiro volte a fita até o início. Depois digite

SHLOAD

Acione o botão play do gravador. O alto-falante gera bips por duas vezes e então devolve o controle do computador para o teclado.

O SHLOAD coloca automaticamente o HIMEM: no seu valor corrente menos o comprimento da tabela lida em bytes.

O endereço inicial da forma é carregado nos endereços de memória 22 a 23.

VARIAÇÕES DE USO

Veja o estudo sobre o HIMEM no manual do seu computador da linha APPLE para maiores detalhes.

Este comando SHLOAD não é disponível na versão INTEGER BASIC.

VEJA TAMBÉM

LOAD

Função **SIN • SIND • SING**

SINTAXE GERAL

Número de linha, PRINT SIN(A)
ou
Número de linha LET Y = SIN(A)

SIN(A) é uma função usada normalmente no modo direto ou no modo programa e geralmente precedida de uma instrução LET ou PRINT.

SIN(A) seno de A, onde A é expresso em radianos.

SIN(A) esta função atribui ou imprime o seno de um ângulo A fornecido em radianos, não em graus.

Um radiano é igual a aproximadamente: 57.29578 graus.

O seno (SIN) é definido como sendo a relação entre o comprimento do lado oposto ao ângulo e o comprimento da hipotenusa.

Esta fórmula se aplica tão-somente aos ângulos retos.

O contrário de SIN é ARCSIN. ARCSIN encontra o valor do ângulo uma vez que seu SIN, ou relação entre lados, seja conhecida.

```
EXEMPLO 1
100 REM USANDO SIN
110 PRINT "DE ENTRADA NUM ANGULO (EXPRESSO EM RADIANOS)";
120 INPUT A
130 Y = SIN (A)
140 PRINT "O SENO DE UM ANGULO DE ";A;" RADIANOS E ";Y
150 END
```

```
EXECUÇÃO DO PROGRAMA
RUN
DE ENTRADA NUM ANGULO(EXPRESSO EM RADIANS)? 2
O SENO DE UM ANGULO DE 2 RADIANOS E .909297
```

Para obter o seno de A quando A é dado em graus, use:
SIN(X*.1745329)

Para se converterem ângulos de radianos em graus, multiplique os radianos por 57.29578.

Alguns computadores permitem dar entrada no ângulo quer em graus, quer em grados (100 grados = 90 graus). Esses computadores utilizam a função SIND para graus e SING para grados.

```
EXEMPLO 2
200 REM USANDO A FUNCAO SIN.
210 REM USANDO SIN PARA OBTER UM DESENHO
220 FOR I = 0 TO 6.4 STEP 0.3
230 PRINT TAB(A+18); "PROGRAMA"
240 NEXT I
250 END
```

EXECUÇÃO DO PROGRAMA

```
RUN
                    PROGRAMA
                        PROGRAMA
                            PROGRAMA
                                PROGRAMA
                                  PROGRAMA
                                    PROGRAMA
                                    PROGRAMA
                                  PROGRAMA
                                 PROGRAMA
                               PROGRAMA
                             PROGRAMA
                           PROGRAMA
                         PROGRAMA
                       PROGRAMA
                      PROGRAMA
                      PROGRAMA
                       PROGRAMA
                         PROGRAMA
                            PROGRAMA
                                PROGRAMA
                                    PROGRAMA
                                        PROGRAMA
```

SE O SEU COMPUTADOR NÃO TIVER ESSA FUNÇÃO

Se o seu interpretador não tiver a capacidade SIN, poderá ser utilizada em seu lugar a seguinte sub-rotina, para computar o seno em radianos.

```
999 GOTO 150
1000 REM * SUBROTINA SENO * INPUT R EM RADIANS SAIDA R
1010 REM UTILIZA TAMBEM INTERNAMENTE C E Z
1020 R=R*57.29578
1030 IF R=0 THEN 1250
1040 Z=ABS(R)/R
1050 C=R
1060 R=Z*R
1070 IF R<360 THEN 1090
1080 R=R-INT(R/360)*360
1090 IF R<=90 THEN 1200
1100 R=R/90
1110 Y=INT(R)
1120 R=(R-Y)*90
1130 ON Y GOTO 1140,1160,1190
1140 R=90-R
1150 GOTO 1200
1160 R=-R
1180 GOTO 1200
1190 R=R-90
1200 R=Z*R/57.29578
1210 IF ABS(R)<2.48616E-4 THEN 1250
1220 Y=R*R
1230 Y=((((Y/72-1)*Y/42+1)*Y/20-1)*Y/6+1)*R
1240 GOTO 1260
1250 Y=R
1260 R=C/57.29578
1270 RETURN
```

Tenha cuidado também de efetuar as seguintes alterações no exemplo 1:

```
125 R = A
130 GOSUB 1000
```

Para achar o SENO de um ângulo (expresso em graus) elimine a linha 1020 ou mude a linha 130 para:

```
130 GOSUB 1030
```

VARIAÇÕES DE USO

Pouquíssimos interpretadores BASIC convertem tudo automaticamente para graus de ângulo.

Esta função não é disponível para a versão Integer BASIC nos micros que seguem a linha APPLE, sendo reconhecida apenas na versão do BASIC APPLESOFT.

VEJA TAMBÉM

TAN, COS, ATN, ACS, ASN

Comando **SLOW**

SINTAXE GERAL

SLOW

SLOW é um comando usado no modo direto, sendo utilizado pelos microcomputadores que seguem a linha SINCLAIR.

Os microcomputadores que seguem a linha SINCLAIR podem trabalhar em duas velocidades SLOW(lento) ou FAST(rápido).

Quando se liga pela primeira vez o computador, este trabalha no modo SLOW podendo-se programar e dar informações simultaneamente no vídeo.

Esse modo é ideal para desenhos animados.

Estando o microcomputador ativado no comando FAST para se voltar ao normal (computação e imagem), escreva

SLOW

É apenas uma questão de gosto o fato de se querer mudar o modo de computação e imagem.

Podem-se utilizar as instruções SLOW e FAST nos programas sem qualquer problema.

EX:

```
100 SLOW
110 FOR N = 1 TO 64
120 PRINT "A"
130 IF N = 32 THEN FAST
140 NEXT N
150 GOTO 100
```

VARIAÇÕES DE USO

Não se tem conhecimento de nenhuma.

VEJA TAMBÉM

FAST

Função **SPC**

SINTAXE GERAL

SPC (exprnm)

SPC é uma função aceita pelos microcomputadores da linha APPLE, usada normalmente no modo programa devendo ser precedida por uma instrução geralmente PRINT.

SPC (X) permite imprimir espaços em branco. Assim qualquer carácter passado pelo cursor é zerado.

Só pode ser usado com o comando PRINT; coloca X espaços entre o último item impresso e o próximo. X deve estar entre parênteses.

SPC (0) não dá nenhum efeito (0 espaços).

A expressão é convertida em um inteiro entre 0 e 255.

Todavia é possível realizar a impressão de qualquer número de espaços, usando-se mais instruções SPC.

EXEMPLO:

PRINT SPC(255), SPC(100)

permite imprimir 355 espaços em branco.

SPC pode concatenar os itens precedidos e seguidos por justaposição ou intervenção do ponto-e-vírgula (;).

A função SPC move o cursor para a direita a partir da coluna em que ele estiver no momento em que SPC for encontrado.

Sempre que se inclui a função SPC numa declaração PRINT, faz-se com que o próximo carácter a ser colocado no vídeo apareça deslocado do número de posições especificado.

```
EXEMPLO 1
100 REM USANDO SPC
110 HOME
120 REM PULA ESPACOS E DA MENSAGEM
130 PRINT "PERTO" SPC(10) "LONGE"
140 END
```

```
EXECUÇÃO DO PROGRAMA
RUN
PERTO           LONGE
```

SPC não é disponível em Integer BASIC.

VARIAÇÕES DE USO

Não se tem conhecimento de nenhuma.

VEJA TAMBÉM

;(ponto-e-vírgula) ,(vírgula), AT

Comando **SPEED**

SINTAXE GERAL

SPEED = exprnm
ou
variável numérica

SPPED pode ser utilizado como comando no modo direto ou como instrução no modo programa.

SPEED é um comando ou instrução utilizada pelos microcomputadores que seguem a linha APPLE na versão APPLESOFT não sendo disponível para a versão Integer BASIC.

SPEED = X ajusta a velocidade de saída dos caracteres (0 a 255).

Ajusta a velocidade de emissão de caracteres para a tela ou um outro plano de entrada/saída (I/O).

A velocidade varia de 0 (mais lenta) até 255 (mais rápida).

A velocidade com que os caracteres são mostrados no vídeo é variável. Pode-se diminuir sua velocidade normal através da declaração SPEED.

EXEMPLO 1

```
100 REM USANDO SPEED
110 INPUT "VELOCIDADE"; V
120 SPEED = V
130 FOR CT = 1 TO 3
140 PRINT "VEL"
150 NEXT
160 PRINT "FIM VEL"
170 SPEED = 255
180 END
```

O valor da expressão na linha 120 ajusta a velocidade de impressão; 0 é a menor e 255 é a maior.

VARIAÇÕES DE USO

Pode-se também usar um FOR...NEXT (vazio) para reduzir a velocidade de saída de caracteres no vídeo.

VEJA TAMBÉM

FOR...NEXT, PAUSE

Função SQR • SQRT

SINTAXE GERAL

Número de linha PRINT SQR(X)

SQR é uma função usada geralmente no modo direto ou no modo programa devendo ser precedida de uma instrução LET ou PRINT. A função SQR(X) calcula a raiz quadrada de qualquer número positivo (X) ou expressão positiva representada pelo argumento (X).

EXEMPLO 1

```
100 REM USANDO SQR
110 PRINT "A RAIZ QUADRADA DE 64 E ";
120 PRINT SQR(64)
130 PRINT "SQR FOI ACEITO SE O RESULTADO FOR 8"
140 END
```

EXECUÇÃO DO PROGRAMA

```
RUN
A RAIZ QUADRADA DE 64 E 8
SQR FOI ACEITO SE O RESULTADO FOR 8
```

ORTOGRAFIA ALTERNATIVA

SQRT é utilizado por alguns microcomputadores para indicar a função de raiz quadrada

SE O SEU COMPUTADOR NÃO TIVER ESSA FUNÇÃO:

Se o computador falhou no exemplo 1, use a seguinte sub-rotina:

```
999 GOTO 140
1000 REM *SUBROTINA DE RAIZ QUADRADA* INPUT R, SAIDA Y
1010 REM UTILIZA TAMBEM INTERNAMENTE VARIAVEIS R E Y
1020 IF R=0 THEN 1140
1030 IF R>0 THEN 1060
1040 PRINT "RAIZ QUADRADA DE NUMERO NEGATIVO?"
1050 STOP
1060 Y=R/4
1070 Z=0
1080 W=(R/Y-Y)/2
1090 IF W=0 THEN 1150
1100 IF W=Z THEN 1150
1110 Y=Y+W
1120 Z=W
1130 GOTO 1080
1140 Y=0
1150 RETURN
```

Para utilizar essa sub-rotina no exemplo 1, efetue as seguintes alterações:

```
115 R = 64
120 GOSUB 1000
130 PRINT Y
```

VARIAÇÕES DE USO

Não se tem conhecimento de nenhuma.

Função **STEP • STE • ST • S.**

SINTAXE GERAL

Número de linha FOR varnm = valor inicial TO valor final STEP incremento.

STEP é uma função usada geralmente no modo programa devendo ser precedida de uma instrução FOR...TO...

A função STEP é utilizada para especificar a extensão entre passos numa instrução FOR-NEXT.

O valor do incremento do STEP pode ser representado por um número inteiro ou fracionário, positivo ou negativo e também por uma variável ou expressão numérica.

Há casos em que o STEP é opcional:

Não sendo especificado o valor para o incremento do STEP admite-se automaticamente pelo microcomputador o número +1 e o STEP não precisará ser digitado no programa.

Na sintaxe geral, se o valor inicial atribuído for maior que o valor final terá que se digitar a função STEP e o seu incremento terá que ser negativo.

EXEMPLO:

```
10 FOR X = 1 TO 10
20 PRINT X
30 NEXT X
40 END
```

EXEMPLO 1

```
100 REM USANDO STEP
110 PRINT "SENDO O VALOR DE STEP 3,J =";
120 FOR J = 1 TO 20 STEP 3
130 PRINT J;
140 NEXT J
150 END
```

EXECUÇÃO DO PROGRAMA

```
RUN
SENDO O VALOR DE STEP 3,J = 14710131619
```

EXEMPLO 2

```
200 REM USANDO "STEP NEGATIVO"
210 PRINT "SENDO O VALOR DE STEP -1,J = ";
220 FOR J = 20 TO 1 STEP -1
230 PRINT J;
240 NEXT J
250 END
```

EXECUÇÃO DO PROGRAMA

```
RUN
SENDO O VALOR DE STEP -1,J = 20191817161514131211109876543 21
```

O exemplo 3 verifica a capacidade do interpretador para

manejar valores de STEP decimais não-inteiros.

```
EXEMPLO 3
300 REM UTILIZANDO "STEP NAO INTEIRO"
310 PRINT "SENDO O VALOR DE STEP .7,J = ";
320 FOR J = 1 TO 10 STEP .7
330 PRINT J;
340 NEXT J
350 END

EXECUÇÃO DO PROGRAMA
RUN
SENDO O VALOR DE STEP .7,J =
11.72.43.13.84.55.25.96.67.388.79.4
```

Determinados interpretadores aceitam uma variável como valor de STEP, assim como, por exemplo, FOR J = 1 TO 30 STEP A faz com que o valor de J seja incrementado pelo valor da variável A cada vez que se execute a correspondente instrução NEXT.

```
EXEMPLO 4
400 REM USANDO STEP COMO VARIAVEL
410 PRINT "DE ENTRADA NUM VALOR DE STEP (ENTRE 1 E 5)"
420 INPUT N
430 PRINT "O VALOR DE J = ";
440 FOR J = 1 TO 10 STEP N
450 PRINT J;
460 NEXT J
470 END

EXECUÇÃO DO PROGRAMA
RUN
DE ENTRADA NUM VALOR DE STEP (ENTRE 1 E 5)
? 5
O VALOR DE J = 16
```

ORTOGRAFIAS ALTERNATIVAS

Alguns microcomputadores utilizam STE, outros ST e os da linha TRS-80 nível 1 aceitam S. no lugar de STEP.

SE O SEU COMPUTADOR NÃO TIVER ESSA FUNÇÃO

Não sendo STEP intrínseco no seu computador ou não sendo suficientemente poderosa a referida função, poderá facilmente ser simulada por instruções FOR-NEXT ascendentes. Omita STEP N na linha 440 no último exemplo, e acrescente as linhas seguintes:

```
435 C = 1
450 PRINT C;
455 C = C + N
457 IF C = 5 GOTO 470
```

Inserir essas linhas logo antes da correspondente instrução NEXT permite incrementar J por qualquer número inteiro ou fração decimal que se queira.

VEJA TAMBÉM

FOR, NEXT

Instrução **STOP • STO • ST • S.**

SINTAXE GERAL

Número de linha STOP

STOP é uma instrução reconhecida pela maioria das linhas de microcomputadores e quando executada interrompe a execução do programa, retornando o controle do microcomputador ao usuário.

A instrução STOP pode ser colocada em qualquer linha do programa para parar sua execução.

CONT faz uma pausa, isto é, uma parada lógica e não uma parada definitiva como a instrução END.

Alguns microcomputadores interrompem a execução do programa na linha que contém a instrução STOP ao passo que outros saltam para a linha que contém a instrução END.

Muitos microcomputadores dotados de interpretadores (não, porém, via de regra, os providos de compilador) imprimem o número de linha em que parou o programa e permitem o prosseguimento da execução do programa mediante o comando CONT.

STOP ao interromper a execução do programa simula a tecla BREAK.

Esta instrução pode ser utilizada quando se deseja depurar o programa. Durante o BREAK pode-se examinar o programa em execução e mudar variáveis.

EXEMPLO 1

```
100 REM USANDO STOP
110 PRINT "STOP PROVOCARA UMA PAUSA"
120 STOP
130 PRINT "A INSTRUCAO STOP NAO FOI ACEITA"
140 END
```

EXECUÇÃO DO PROGRAMA

```
RUN
STOP PROVOCARA UMA PAUSA
BREAK NA 120
```

ORTOGRAFIAS ALTERNATIVAS

STO é utilizado em vez de STOP em alguns microcomputadores americanos e europeus, ao passo que o microcomputador TRS-80 Nível 1 utiliza ST. e S.

VARIAÇÕES DE USO

Tratar de utilizar tanto STOP como END no mesmo programa poderá acarretar extrema frustração, a não ser que se conheçam as capacidades da máquina. Alguns micros exigem intervenção física (pressionar um botão) antes de RUN (rodar), após ter sido acionada a instrução STOP no programa.

Outros (principalmente computadores de grande porte) aceitam um número ilimitado de STOPs, mas apenas um END. Ainda outros permitem um número ilimitado de ENDs, não, porém, STOPs. A

maioria dos micros permite misturar STOPs e ENDs sem acarretar dificuldades.

Contanto que se proceda com cuidado, o problema STOP/END pode quase sempre ser resolvido, convertendo-se facilmente os programas.

STOP não é aceito na versão Integer BASIC dos microcomputadores que seguem a linha APPLE II.

VEJA TAMBÉM
CONT, END, GO

Instrução/Comando **STORE**

SINTAXE GERAL

Número de linha, STORE variável numérica

STORE pode ser usado como comando no modo direto ou como instrução no modo programa em microcomputadores que seguem a linha APPLE.

STORE varnm é utilizado com o propósito de salvar uma matriz de valores numéricos em uma fita cassete.

O comando ou instrução STORE varnm não controla o movimento da fita cassete e nem adverte quando se deve apertar a tecla de gravação do gravador. Deve-se ter o cassete rodando e pronto para gravar quando o STORE for executado.

Uma matriz extensa pode ser colocada na fita magnética sob controle do programa, sendo em seguida RECALL-ada (chamada de volta) pelo mesmo programa ou por outro.

Um programa na versão BASIC APPLESOFT deve mostrar indicações contidas através de instruções PRINT. O computador sinalizará quando iniciar o armazenamento de valores e sinalizará de novo quando terminar.

EXEMPLO 2

```
100 REM PROGRAMA USANDO STORE
110 DIM X(25),Y(25)
120 FOR R = 1 TO 25
130 X(R) = R
140 NEXT R
150 STORE X
160 PRINT "REBOBINE FITA E COLOQUE EM PLAY, PRESSIONE
RETURN"
170 INPUT X$
180 RECALL Y
190 FOR R = 1 TO 25
200 PRINT Y(R),
210 NEXT R
220 END
```

EXECUÇÃO DO PROGRAMA

```
REBOBINE A FITA E COLOQUE EM PLAY, PRESSIONE RETURN

?
1           2           3
4           5           6
7           8           9
10          11          12
13          14          15
16          17          18
19          20          21
22          23          24
25
```

VARIAÇÕES DE USO

Só se pode usar STORE para guardar valores numéricos;

valores ou variáveis alfanuméricas devem ser convertidas para valores inteiros usando a função ASC antes do armazenamento.

Esta instrução não é disponível na versão Integer BASIC para a linha APPLE.

VEJA TAMBÉM
RECALL, CSAVE, DIM

Função **STRING$ • STRING • STR**

SINTAXE GERAL

Número de linha PRINT STRING$ (n "carácter" ou número código ASCII)

STRING$ (n, "carácter") é uma função utilizada no modo direto ou no modo programa, sendo precedida por um comando ou instrução PRINT.

STRING$ (n "carácter" ou número código ASCII) é utilizado principalmente nos microcomputadores que seguem a linha TRS-80 para retornar um valor alfanumérico. Esta função não é disponível para os microcomputadores da linha APPLE II.

A função STRING$ (n,código ASCII ou "carácter") é utilizada em conjunto com a instrução PRINT para imprimir um carácter ASCII(n) número de vezes.

STRING$ (n "carácter") O argumento n pode ser qualquer expressão numérica com o valor de zero a 255.

PRINT STRING$(10,66), por exemplo, imprime o carácter ASCII B (código ASCII 66) dez vezes.

PRINT STRING$ (20,"*")
resulta em **********

STRING$ (n,"carácter") o carácter pode ser representado por qualquer número de 0 a 255; neste caso, ele será manipulado como um código gráfico, de controle ou ASCII.

Esta função é bastante útil quando da elaboração de tabelas, gráficos etc.

```
EXEMPLO 1
100 REM USANDO A FUNCAO STRING$
110 PRINT STRING$(9,"*")
120 PRINT "FUNCAO STRING$ ";
130 PRINT STRING$(9,"*")
140 END
```

```
EXECUÇÃO DO PROGRAMA
RUN
*********FUNCAO STRING$**********
```

VARIAÇÕES DE USO

Alguns computadores (tais como o TRS-80) permitem caracteres de string (entre aspas) ou variáveis de string na função STRING$.

Por exemplo, 10 PRINT STRING$(15,"-")
imprime o carácter "-" quinze vezes.

```
EXEMPLO 2
200 B$ = "X"
210 PRINT STRING$ (5,B$)
```

```
EXECUÇÃO DO PROGRAMA
RUN
XXXXX
```

EXEMPLO 3

```
300 REM USANDO STRING$
310 PRINT "DE ENTRADA EM QUALQUER LETRA, NUMERO OU SIMBOLO";
320 INPUT L$
330 PRINT STRING$ (10,"#");
340 PRINT STRING$ (10,L$)
350 END
```

EXECUÇÃO DO PROGRAMA

```
RUN
DE ENTRADA EM QUALQUER LETRA, NUMERO OU SIMBOLO? Y
##########YYYYYYYYYY
```

ORTOGRAFIAS ALTERNATIVAS

Alguns computadores aceitam STRING ou STRING$.

SE O SEU COMPUTADOR NÃO TIVER ESTA FUNÇÃO

Se o seu computador não aceitar a função STRING$, esta poderá ser simulada procurando na tabela ASCII o carácter que está em conformidade com o código ASCII listado na função STRING$. Coloque numa instrução PRINT o número de vezes especificado pelo primeiro número da função.

VEJA TAMBÉM

PRINT, ASC, CHR$, LEN, MID$, LEFT$, RIGHT$, STR$, VAL

Função **STR $**

SINTAXE GERAL

Número de linha PRINT STR$(exprnm)

STR$(X) é uma função usada normalmente no modo programa, devendo ser precedida por uma instrução geralmente LET ou PRINT.

Esta função é reconhecida pela maioria das versões BASIC.

A função STR(X) converte uma variável numérica, expressão ou constante numérica em STRING.

A variável numérica, expressão ou constante numérica deve permanecer entre parênteses.

Por exemplo:

```
100 A$ = STR$(91)
110 PRINT A$
```

Imprime o número 91 sob forma de string. O computador deixa automaticamente o lugar antes do número para colocação do respectivo sinal. Caso o número seja positivo, esse espaço é deixado em branco.

A conversão de um número a um string mediante a função STR$ permite a sua manipulação pelo uso de modificadores de string (tais como LEFT$, RIGHT$, MID$, ASC etc.)

EXEMPLO 1

```
100 REM USANDO A FUNCAO STR$
110 A = -85
120 B = -19
130 A$ = STR$(A)
140 B$ = STR$(B)
150 PRINT A$
160 PRINT B$
170 C$ = MID$(A$,2)
180 D$ = MID$(B$,2)
190 E$ = C$+D$
200 PRINT E$
210 END
```

EXECUÇÃO DO PROGRAMA

```
RUN
-85
-19
8519
```

NOTA:

No exemplo acima você pode efetuar operações aritméticas com as variáveis A e B, mas apenas manipulações de strings com as variáveis A$,B$,C$, e E$.

VARIAÇÕES DE USO

STR$(exprnm)

O valor da expressão numérica é convertido em um valor alfanumérico.

O valor alfanumérico é o mesmo que se obteria com PRINT exprnm.

EXEMPLO: STR$(2/3) = ".666666667"

Se a expressão numérica exceder o limite designado para números reais, aparecerá na tela a mensagem de erro: OVERFLOW ERROR.

Esta função não é disponível na versão INTEGER BASIC da linha APPLE.

VEJA TAMBÉM

ASC, CHR$, LEN, LEFT$, MID$, RIGHT$, STRING$, VAL, NUM$

Comando/Instrução **SYSTEM • SYS**

SINTAXE GERAL

SYSTEM

SYSTEM pode ser utilizado como comando no modo direto ou como instrução no modo programa.

SYSTEM carrega do cassete, programa em linguagem de máquina.

SYSTEM é utilizado por determinados microcomputadores principalmente os que seguem a linha TRS-80 para permitir que se carreguem dados de linguagem de máquina (código objeto) de fita cassete ou disquete para o microcomputador.

Executando no microcomputador a linha que contém a instrução SYSTEM, ou se teclando SYSTEM no terminal, o computador muda para o modo de monitoragem e imprime um asterisco seguido de um ponto de interrogação (*?) ou algum outro símbolo. Esse sinal indica que o computador está pronto para aceitar o arquivo objeto a partir de disco ou fita magnética.

Coloque no gravador cassete uma fita com código objeto, e ponha o gravador no modo PLAY. Tecle o nome do arquivo objeto e RETURN. O motor do gravador cassete é controlado pelo computador, que o liga e desliga antes e depois do ciclo de carregamento. O cassete deverá "tocar" os dados de volta para o computador. Uma vez carregados os dados no computador, é exibido outro *?.

EXEMPLO

Você pode também criar seu próprio arquivo-objeto utilizando o EDITOR ASSEMBLER do seu microcomputador, e se ele segue a linha TRS-80 faça o seguinte:

Prepare o gravador. Para carregar este arquivo digite SYSTEM e tecle ENTER, então o símbolo *? aparecerá na tela.

Ao digitar agora o nome-de-arquivo, não sendo necessário as aspas, a fita começará a ser lida. Durante a leitura da fita aparecerá no canto superior direito do vídeo dois asteriscos, um fixo outro piscando. Quando o arquivo já tiver sido carregado aparecerá outro *? no vídeo.

Digite uma barra / seguida de um endereço (em forma decimal) da maneira que você desejar que a execução se inicie ou simplesmente pressione / ENTER sem nenhum endereço. Neste caso a execução começará no endereço especificado pelo arquivo-objeto.

ORTOGRAFIAS ALTERNATIVAS

Alguns microcomputadores americanos utilizam SYS como abreviatura de SYSTEM.

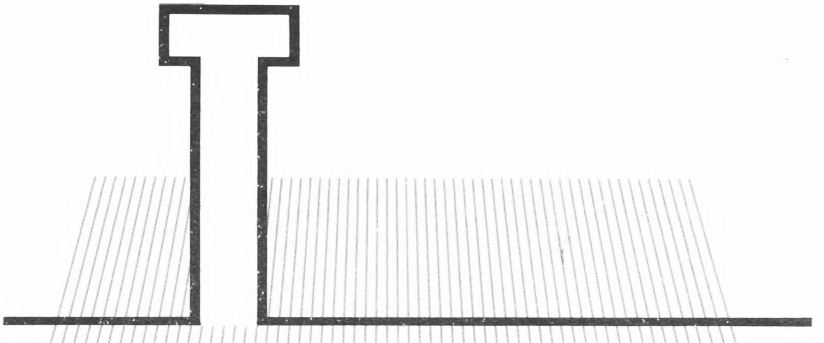
VARIAÇÕES DE USO

Nos microcomputadores da linha APPLE utiliza-se CALL-151 no lugar de SYSTEM.

O comando SYSTEM se assemelha à tecla (escape) em muitos teclados. Ambos colocam o comando no modo System.

VEJA TAMBÉM

PEEK, POKE, MON



TAB • T. • TAN • TAND • TANG
TAPPEND • TAPE • TEXT
THEN • THE • T. • TIME
TIME$ • TLOAD • TOP
TRACE • TRACE OFF
TRACE ON • TROFF • TRON

Função **TAB • T.**

SINTAXE GERAL

PRINT TAB(exprnm); ou TAB coi
Número de linha, instrução PRINT função TAB(N)

TAB(N) é uma função usada normalmente no modo programa e geralmente precedida de uma instrução PRINT.

Para TAB(expnm) ou TAB(N) N representa as colunas que são numeradas de 1 a 255.

Se exprnm ou N é maior que o dispositivo de saída (40 caracteres linha APPLE ou 63 linha TRS-80) para o vídeo ele moverá o cursor uma linha para baixo e continuará contando a partir da margem esquerda.

A função TAB(N) é utilizada em conjunto com instruções PRINT de maneira semelhante à tecla TAB numa máquina de escrever, para mover o cursor para a direita de acordo como valor N.

TAB(N) coloca caracteres em branco durante o movimento do cursor enquanto se movimenta para a direita zerando assim o que havia previamente na tela.

Sendo seguida a instrução PRINT por TAB(N), o computador insere um número de espaços (contido entre parênteses) à frente da instrução a ser impressa. O valor de N deve sempre ser positivo, e deve ser entre 1 e 255.

TAB(N) não pode ser usado para mover o cursor para a esquerda. Se N for zero TAB move para a coluna 256 e se o valor de N ou o da expnm estiver fora da faixa especificada (1-255) o TAB será ignorado e causará uma mensagem de erro. Veja HTAB e TAB(Integer Basic) em computadores da linha APPLE.

Sendo utilizada mais de uma instrução TAB(N) numa única linha, os números devem ficar progressivamente maiores, deixando espaço entre um e outro para o material que vai ser impresso. Deixando-se entre TABs espaço suficiente, será ultrapassado o espaço disponível, assim como ocorreria com a máquina de escrever.

O valor pode ser expresso sob forma de número, PRINT TAB (10); uma variável, PRINT TAB (N); ou uma expressão, PRINT TAB (7Y+X).

A função TAB(N) deve ser seguida de ponto-e-vírgula ou uma vírgula, mas isto dependendo do respectivo interpretador.

```
EXEMPLO 1
100 REM USANDO A FUNCAO TAB
110 X = 3
120 PRINT TAB(X); "SULLIVAN"
130 PRINT TAB(X*X); "SULLIVAN"
140 PRINT TAB TAB(X*5+5); "TAB (20)"
150 END

EXECUÇÃO DO PROGRAMA
RUN
  SULLIVAN
        SULLIVAN
```

O valor máximo que seu computador é capaz de TAB se determina rapidamente por se acrescentarem ao exemplo 1 as seguintes linhas:

```
150 PRINT "TECLE UM VALOR DE TAB";
160 INPUT X
170 PRINT TAB(X); "TAB";X
180 GOTO 150
```

O valor de TAB a que se deu entrada na linha 160 fará com que a linha 170 imprima o valor de TAB após o mesmo número de espaços.

ORTOGRAFIA ALTERNATIVA

Alguns computadores americanos e europeus utilizam versões BASIC que aceitam T. como abreviação de TAB.

VARIAÇÕES DE USO

Não se tem conhecimento de nenhuma.

SE O SEU COMPUTADOR NÃO TIVER ESTA FUNÇÃO

Não existe nenhum substituto totalmente satisfatório de TAB, havendo, contudo, diversas maneiras de obter saídas impressas que poderão ser aceitáveis. Admitindo-se uma série original PRINT:

```
200 PRINT TAB(5); "O";TAB(15);"SULLIVAN";TAB(30)"ENSINA"
210 PRINT TAB(7);"RAPIDO"
RUN
    O           SULLIVAN        ENSINA
     RAPIDO
```

Os valores de TAB são números simples e poderiam ser substituídos por:

```
200 PRINT"  O   SULLIVAN   ENSINA   RAPIDO"
```

ou, de forma menos acurada

```
200 PRINT "O","SULLIVAN","ENSINA","RAPIDO"
RUN
O           SULLIVAN
ENSINA      RAPIDO
```

ou

por uma combinação de espaços inseridos e espaçamento automático de zonas.

Um terceiro método, geralmente satisfatório, de se chegar a uma saída impressa utilizável envolve combinar a capacidade de supressão de retrocesso do ponto-e-vírgula (;), o zoneamento automático pela vírgula (,), e os espaços inseridos. Com determinados interpretadores (compiladores), no entanto, isto poderá dar lugar a uma situação bastante mal ajeitada.

VEJA TAMBÉM

PRINT, PRINT USING, PRINT AT, ,(vírgula), SPACES$, HTAB, TAB

Função **TAN • TAND • TANG**

SINTAXE GERAL

Número de linha LET Y = TAN (N)

TAN (N) é uma função reconhecida em muitas versões BASIC e quando executada retorna à tangente do número ou expressão numérica N fornecida em radianos.

Em todas as funções, o argumento (N) deve estar incluído entre parênteses e pode ser tanto uma variável numérica ou um número.

EXEMPLO:

Se N = 1.57, então TAN (1.57) e TAN (N) resulta numa mesma resposta do microcomputador.

TAN é uma função usada geralmente no modo programa.

O contrário de TAN é ARCTAN (ATN).

EXEMPLO 1

```
100 REM USANDO A FUNCAO TAN
110 PRINT "DE ENTRADA NUM ANGULO EM RADIANOS";
120 INPUT X
130 T = TAN (X)
140 PRINT "A TANGENTE DE UM ANGULO DE " ;X;" RADIANOS E ";T
150 END
```

EXECUÇÃO DO PROGRAMA

```
RUN
DE ENTRADA NUM ANGULO EM RADIANOS? 2
A TANGENTE DE UM ANGULO DE 2 RADIANOS E -2.18504
```

Para converter valores de graus para radianos, multiplique o ângulo em graus por .0174533. Por exemplo: R = TAN(A*.0174533). Para converter valores de radianos para graus, multiplique o ângulo em radianos 57.29578.

Alguns computadores aceitam a medida do ângulo em graus ou grados (100 grados = 90 graus). Esses computadores utilizam a função TAND para graus e TANG para grados.

VARIAÇÕES DE USO

Alguns porém raros interpretadores BASIC convertem automaticamente os valores para graus.

VEJA TAMBÉM

SIN, COS, ATN, ASN, ACS

Comando **TAPPEND • TAPE**

SINTAXE GERAL

TAPPEND

TAPPEND é um comando reconhecido por poucos interpretadores BASIC.

Este comando quando executado combina um programa obtido de uma fita cassete com outro que já se encontra armazenado na memória RAM.

TAPPEND significa Tappe APPEND (acrescentar/conteúdo de /fita).

Os números de linha das instruções de programa trazidas para a memória devem ser superiores ao último número de linha já constante da memória.

```
EXEMPLO 2
500 PRINT "EU SOU O"
600 PRINT "PROGRAMA 2"
700 END
```

Tecle então END ou SCRATCH para apagar o programa e tecle em seguida o exemplo 1:

```
EXEMPLO 1
100 REM "VERIFICANDO O EXEMPLO 1"
110 PRINT "EU DERIVO"
120 PRINT "DO PROGRAMA 1"
130 PRINT "        MAS..."
```

Tecle em seguida TAPPEND PROGRAMA 2 e RUN

```
EXECUÇÃO DO PROGRAMA
EU DERIVO
DO PROGRAMA 1
    MAS...
EU SOU O
PROGRAMA 2
```

ORTOGRAFIAS ALTERNATIVAS

Microcomputadores da linha TRS-80 utilizam: TAPE(S=fonte D=destino) para transferir programas em linguagem de máquina, d fita para o disquete, de disquete para fita e de fita para memória RAM (T=fita, D=disquete, R=RAM)

EXEMPLO:

```
TAPE (S=D, D=T)
```

VEJA TAMBÉM

APPEND, CLOAD, CSAVE, TLOAD, TSAVE

Comando/Instrução **TEXT**

SINTAXE GERAL

Número de linha TEXT

TEXT é uma instrução utilizada pelos microcomputadores que seguem a linha APPLE II para retornar ao vídeo (tela) o modo TEXTO a partir de qualquer um dos modos gráficos.

TEXT pode ser utilizado como comando no modo direto, porém e mais utilizado como instrução no modo programa mudando a execução do programa do modo gráfico para o modo TEXTO (narrativo) normal.

Se na janela de texto foi colocado um gráfico ou algo diferente da tela cheia, a instrução TEXT quando executada apaga a tela cheia.

OBS: TEXT não apaga a tela, ou mais precisamente, não limpa a página de memória de baixa resolução. Quando o modo TEXTO está sendo utilizado na tela com gráfico em baixa resolução, TEXT quando executado deixará, no modo gráfico de baixa resolução, as 20 linhas de cima da tela **cheias de caracteres estranhos.**

TEXT é utilizado geralmente no modo programa.

EXEMPLO 1

```
100 REM USANDO A INSTRUCAO TEXT
110 TEXT
120 PRINT "A INSTRUCAO TEXT FOI ACEITA"
130 END
```

EXECUÇÃO DO PROGRAMA

```
RUN
A INSTRUCAO TEXT FOI ACEITA
```

VARIAÇÕES DE USO

TEXT é utilizado em algumas versões BASIC para especificar determinadas variáveis designadas sob forma de variáveis de string. TEXT X, Y, Z, por exemplo, define como variáveis de string as variáveis X, Y, Z.

VEJA TAMBÉM

GR, DEFSTR

Instrução **THEN • THE • T.**

SINTAXE GERAL

Número de linha, IF condição THEN instrução ou número de linha.

THEN quando executado tem como função inicial a sentença-ação de uma instrução do tipo IF/THEN.

Em alguns microcomputadores THEN é opcional, exceto quando se requer a eliminação de uma ambigüidade, como em:

IF A < 0 100

A instrução THEN deve ser usada em instruções IF/THEN/ELSE.

THEN e utilizado em conjunto com a instrução IF para indicar a próxima operação que o microcomputador deve realizar, uma vez satisfeita a condição da instrução IF.

Para mais informações, veja IF-THEN

EXEMPLO 1

```
100 REM USANDO A INSTRUCAO THEN
110 A = 10
120 IF A = 10 THEN 130
130 PRINT "THEN FOI ACEITO"
140 GOTO 160
150 PRINT "THEN NAO FOI ACEITO"
160 END
```

EXECUÇÃO DO PROGRAMA

```
RUN
THEN FOI ACEITO
```

ORTOGRAFIAS ALTERNATIVAS

THE é aceito no lugar de THEN em algumas versões BASIC e também micros TRS-80 Nível 1.

VARIAÇÕES DE USO

IF...THEN...ELSE

THEN testa a expressão e se for verdadeira executa instruções após o THEN. Se for falsa executará as instruções após o ELSE.

Em algumas versões do BASIC - não nas dos micros da linha SINCLAIR - IF pode tomar a forma IF condição THEN número da linha, mas significa o mesmo que:

IF condição THEN GOTO número da linha.

VEJA TAMBÉM

IF-THEN

Comando/Instrução **TIME**

SINTAXE GERAL

Número de linha variável = TIME

TIME é utilizado como característica especial em determinados microcomputadores para indicar o tempo decorrido em segundos ou frações de segundo a partir de conhecido ponto de referência no tempo.

A maioria das máquinas de "tempo compartilhado" conta o tempo a partir da meia-noite (vinte e quatro horas) até a meia-noite seguinte, ao passo que as máquinas autônomas contam a partir do momento em que o computador é ligado até aquele em que é desligado.

PRINT TIME, por exemplo, poderá imprimir um número semelhante a 024210 para indicar o tempo total de funcionamento do microcomputador em determinadas unidades críticas.

A unidade de medição de tempo do computador varia de maquina para máquina. Alguns micros, por exemplo, incrementam o valor do TIME à razão de 60 vezes por segundo, as unidades que utilizam a versão MAX BASIC incrementam 1.000 vezes por segundo, e na versão BASIC-PLUS-2 incrementa à razão de uma unidade por segundo.

Alguns microcomputadores, relatam o TIME decorrido sob forma de número de seis dígitos, sendo que o valor não pode ser alterado nem recolocado em zero, a não ser por se desligar o computador.

```
EXEMPLO 1
100 REM USANDO TIME
110 X = TIME
120 PRINT "O TEMPO CORRE"
130 FOR R = 1 TO 2000
140 NEXT R
150 Y = TIME
160 IF Y>X THEN 100
170 PRINT "A INSTRUCAO TIME NAO FOI ACEITA"
180 GOTO 200
190 PRINT "TIME FOI ACEITO:TEMPO DECORRIDO = ";Y-X
200 END

EXECUÇÃO DO PROGRAMA
RUN
O TEMPO CORRE
TIME FOI ACEITO:TEMPO DECORRIDO = 270
```

ORTOGRAFIA ALTERNATIVA

TIM é utilizado por alguns microcomputadores americanos ao passo que outros utilizam TI.

VARIAÇÕES DE USO

Alguns microcomputadores que seguem a linha TRS-80 aceitam TIME hh:mm:ss para atualizar ou obter as horas do relógio interno.

Exemplo TIME 11:20:30

Interpretadores BASIC-PLUS-2 utilizam as seguintes variações de TIME:TIME(0) indica o tempo decorrido total, em segundos, desde meia-noite.

100 PRINT TIME(0), por exemplo, poderá imprimir um valor semelhante a 25128, indicando que se passaram 25.128 segundos desde meia-noite.

TIME(1%) indica o tempo total de programa decorrido em décimos de segundos.

100 PRINT TIME(1%), por exemplo, poderá imprimir um valor semelhante a 85, como sinal de que o programa andou durante 8.5 segundos antes de imprimir TIME(1%).

TIME(2%) indica o tempo decorrido total em minutos em que o terminal esteve ligado ao sistema de compartilhamento de tempo.

10 PRINT TIME(2%), por exemplo, poderá imprimir um valor semelhante a 130, indicando que decorreram 130 minutos desde que o terminal foi ligado ao sistema de compartilhamento de tempo.

O BASIC de microcomputadores Hewlett Packard 2000F, de compartilhamento de tempo, utiliza TIME sob forma de comando, para imprimir o tempo decorrido desde que o terminal foi ligado no sistema, bem como o tempo cumulativo da respectiva conta.

VEJA TAMBÉM
TIME$, CLK$, TIME

Função TIME$

SINTAXE GERAL

Número de linha, PRINT TIME$

TIME$ é uma função aceita por alguns microcomputadores que quando utilizada deve vir precedida de PRINT.

TIME$ é utilizado por alguns microcomputadores americanos para indicar a hora.

Em alguns micros indica a hora em: horas (0-24), minutos e segundos, sob forma de número de seis dígitos (hhmmss).

O valor de TIME$ é "acertado" por se atribuir um número de seis dígitos (entre aspas) a TIME$.

TIME$ = "163500", por exemplo, estabelece como as horas a cifra 163500 (que é a mesma coisa que 4.35 da tarde). A contagem avançada por incrementos de um segundo cada a partir do momento em que o computador é ligado (TIME$ é inicializado em 000000) é a partir do momento em que se lhe atribui um valor novo.

Alguns microcomputadores que seguem a linha TRS-80 modelo 1 e III utilizam TIME$ para fornecer a data atual e a hora. Estes microcomputadores têm um relógio interno e, para utilizá-lo, deve-se ajustá-lo na data e hora corretas, executando o exemplo abaixo num microcomputador CP-500:

```
EXEMPLO 1
100 REM ACERTANDO DATA E HORA
110 DEFINT A-Z
120 DIM TM(5)
130 CL = 16924
140 PRINT "ENTRE COM OS VALORES:MM,DD,AA,HR,MN,SS"
150 INPUT TM(0),TM(1),TM(2),TM(3),TM(4),TM(5)
160 FOR I = 0 TO 5
170 POKE CL-I,TM(I)
180 NEXT I
190 PRINT "RELOGIO AJUSTADO"
200 END

EXECUÇÃO DO PROGRAMA
RUN
ENTRE COM 6 VALORES:MM,DD,AA,HR,MN,SS
? 11,26,84,10,35,25
RELÓGIO AJUSTADO
```

Após ajustar o relógio pode-se utilizá-lo a qualquer momento, por exemplo:

OBS: O relógio estará desligado durante operações com gravador cassete e em outras ocasiões. Portanto, ele precisará ser corrigido periodicamente.

```
EXEMPLO 2
200 REM USANDO TIME$
210 PRINT "A HORA ATUAL E";TIME$
220 PRINT "A FUNCAO TIME$ FOI ACEITA"
230 PRINT "SE FOR IMPRESSO UM NUMERO DE SEIS DIGITOS"
```

```
240 END
```

EXECUÇÃO DO PROGRAMA

```
RUN
A HORA ATUAL E 10:38:39
A FUNCAO TIME$ FOI ACEITA
SE FOR IMPRESSO UM NÚMERO DE SEIS DIGITOS
```

ORTOGRAFIA ALTERNATIVA

TI$ é utilizado por microcomputadores de linha americana Commodore PET como abreviatura de TIME$.

VARIAÇÕES DE USO

Microcomputadores que utilizam a versao BASIC-PLUS-2 utilizam TIME$(0%) para indicar as horas, por horas e minutos, deixando-se fora os segundos.

PRINT TIME$(%), por exemplo, irá imprimir a hora com o formato 14:31. O computador insere automaticamente o "dois pontos" entre as horas e minutos.

Na versão BASIC-PLUS-2, por exemplo, utiliza-se também TIME$(n) para indicar a hora em (n) minutos antes da meia-noite. PRINT TIME$(61), por exemplo, imprime 22:29.

VEJA TAMBÉM

TIME, CLK$

Comando **TLOAD**

SINTAXE GERAL

TLOAD

TLOAD é um comando utilizado no modo direto e reconhecido por poucos microcomputadores, e quando executado carrega o programa de uma fita cassete para a memória do microcomputador.

Se o seu microcomputador aceita TLOAD dê entrada a este programa no microcomputador e armazene em fita cassete. (Veja TSAVE, onde aparecem os detalhes).

EXEMPLO

```
100 REM USANDO O COMANDO TLOAD
110 PRINT "ESTE PROGRAMA TESTA A CARACTERISTICA TLOAD"
120 END
```

Uma vez o programa gravado em fita cassete, apague a memória do computador mediante a instrução NEW (ou SCRATCH, ou qualquer outra instrução que seja apropriada).

Rebobine a fita, coloque o gravador no modo PLAY, e tecle o comando TLOAD.

Ao parar o gravador, LIST o programa para verificar se o programa carregado está em conformidade com o programa acima. Se tudo estiver bem, execute o programa digitando RUN e RETURN (CR - ENTER ou equivalente).

EXECUÇÃO DO PROGRAMA

```
ESTE PROGRAMA TESTA A CARACTERISTICA TLOAD
```

VARIAÇÕES DE USO

Não se tem conhecimento de nenhuma.

VEJA TAMBÉM

CLOAD, LOAD, TSAVE, LIST, NEW, SCRATCH

Função **TOP**

SINTAXE GERAL

Número de linha, condição IF função TOP

TOP é uma função reconhecida por poucos interpretadores BASIC, devendo ser utilizada dentro de uma instrução IF...THEN...ou precedida por uma instrução LET ou PRINT.

A função TOP é geralmente usada no modo programa.

A função TOP em alguns microcomputadores americanos identifica o endereço do primeiro byte de memória ainda não utilizada.

Se o seu microcomputador aceita a função TOP; PRINT TOP, por exemplo, irá imprimir o endereço inicial (em hexadecimal) da memória disponível.

Se souber o endereço inicial (SA) do seu programa, PRINT TOP-SA irá indicar qual o tamanho do programa.

EXEMPLO 1

```
100 REM USANDO A FUNCAO TOP
110 IF TOP = 0 THEN 50
120 PRINT "A PARTE DISPONIVEL DA MEMORIA COMECA EM ";TOP
130 GOTO 150
140 PRINT "TOP NAO FOI ACEITO"
150 END
```

EXECUÇÃO DO PROGRAMA

```
RUN
A PARTE DISPONIVEL DA MEMORIA COMECA EM 285A
```

O endereço exato dependerá do tamanho de memória do microcomputador e da quantidade de memória que está sendo utilizada na atualidade .

VARIAÇÕES DE USO

Não se tem conhecimento de nenhuma.

VEJA TAMBÉM

FRE(0), MEM

Comando/Instrução **TRACE**

SINTAXE GERAL

TRACE
ou
Número de linha TRACE

TRACE pode ser utilizado como comando no modo direto ou como instrução no modo programa.

É reconhecido nos microcomputadores que seguem a linha APPLE e quando executado traça o fluxo da execução do programa mostrando o número da linha de cada instrução que estiver executando.

TRACE é utilizado para ativar uma característica que imprime números de linha de programa à medida que cada linha seja executada pelo computador. É utilizado como ajuda para detecção de defeitos. Essa característica é posta fora de uso pelo comando NOTRACE.

TRACE pode também ser utilizado como instrução de programa para permitir que se rastreiem apenas determinadas seções de programas.

Para ver como TRACE funciona, digite o exemplo abaixo digitando em seguida o comando TRACE e depois RUN.

EXEMPLO 1

```
100 REM USANDO A INSTRUCAO TRACE
110 PRINT "TRACE PASSA CADA LINHA"
120 PRINT "OU DETERMINADAS PARTES DO PROGRAMA"
130 GOTO 180
140 PRINT "ATE SER DESLIGADO PELA"
150 NOTRACE
160 PRINT "INSTRUCAO NOTRACE"
170 GOTO 200
180 PRINT "QUE SEGUE A INSTRUCAO TRACE"
190 GOTO 140
200 PRINT "CONFORME ILUSTRADO POR ESTA LINHA"
210 END
```

EXECUÇÃO DO PROGRAMA

```
RUN
#110 TRACE PASSA CADA LINHA
#120 OU DETERMINADAS PARTES DO PROGRAMA
#130 #180 QUE SEGUE A INSTRUCAO TRACE
#190#140 ATE SER DESLIGADO PELA
#150 INSTRUCAO NOTRACE
CONFORME ILUSTRADA POR ESTA LINHA
```

VARIAÇÕES DE USO

Não se tem conhecimento de nenhuma.

VEJA TAMBÉM

NOTRACE, TRON, TRACE ON, FLOW

Comando/Instrução **TRACE OFF**

SINTAXE GERAL

TRACE OFF
ou
Número de linha, TRACE OFF

TRACE OFF pode ser utilizado como comando no modo direto ou como instrução no modo programa. É reconhecido na versão BASIC da MOTOROLA e quando executado desliga a função TRACE.
TRACE OFF pode ser utilizado como instrução de programa para desligar o TRACE (rastreamento) em determinadas áreas do programa.

EXEMPLO 1

```
100 REM USANDO A INSTRUCAO TRACE OFF
110 TRACE ON
120 PRINT "CADA LINHA DEVE SER PASSADA"
130 TRACE OFF
140 PRINT "PELA INSTRUCAO TRACE ON"
150  PRINT  "ATE QUE ESTA SEJA DESLIGADA PELA INSTRUCAO TRACE
OFF"
160 END
```

EXECUÇÃO DO PROGRAMA

```
RUN
<120> CADA LINHA DEVE SER PASSADA
<130> PELA INSTRUCAO TRACE ON
ATE QUE ESTA SEJA DESLIGADA PELA INSTRUCAO TRACE OFF
```

VARIAÇÕES DE USO
Não se tem conhecimento de nenhuma.

VEJA TAMBÉM
TRACE ON, NOTRACE, TROFF, NOFLOW

Comando/Instrução **TRACE ON**

SINTAXE GERAL

TRACE ON
ou
Número de linha TRACE ON

TRACE ON pode ser utilizado como comando no modo direto ou como instrução no modo programa. É reconhecido na versão BASIC da MOTOROLA e quando executado traça o fluxo da execução do programa mostrando o número da linha de cada instrução que estiver sendo executada.

O comando TRACE ON é utilizado para ativar uma característica que imprime números de linha de programa à medida que cada linha seja executada pelo computador.

Também pode ser utilizado como auxiliar para a verificação de defeitos. Essa característica de rastreamento é colocada fora de ação pelo comando TRACE OFF.

TRACE ON pode ser utilizado sob forma de instrução de programa para rastrear apenas determinadas seções de um programa.

Para ver como TRACE ON funciona, digite o exemplo abaixo e em seguida tecle TRACE ON e RUN.

```
EXEMPLO 1
100 REM USANDO A INSTRUCAO TRACE ON
110 PRINT "TRACE ON PASSA CADA LINHA"
120 PRINT "OU DETERMINADAS PARTES DO PROGRAMA"
130 GOTO 180
140 PRINT "ATE SER DESLIGADO POR"
150 TRACE OFF
160 PRINT "A INSTRUCAO TRACE OFF"
170 GOTO 200
180 PRINT "QUE SEGUE A INSTRUCAO TRACE ON"
190 GOTO 140
200 PRINT "CONFORME ILUSTRADA POR ESTA LINHA"
210 END

EXECUÇÃO DO PROGRAMA
RUN
TACE ON PASSA CADA LINHA
<120> OU DETERMINADAS PARTES DO PROGRAMA
<130> <180> QUE SEGUE A INSTRUCAO TRACE ON
<190> <140> ATE SER DESLIGADO POR
<150> TRACE OFF
```

VARIAÇÕES DE USO
Não se tem conhecimento de nenhuma.

VEJA TAMBÉM
TRACE OFF, TRACE, TRON, FLOW

Comando/Instrução **TROFF**

SINTAXE GERAL

TROFF
ou
Número de linha, TROFF

TROFF pode ser utilizado como comando no modo direto ou como instrução no modo programa. É reconhecido por muitos interpretadores BASIC principalmente nos de microcomputadores que seguem a linha TRS-80.

TROFF (= TACE OFF) desativa o marcador (traço) desligando a característica de rastreamento em determinadas áreas do programa.

Para ver como TOFF funciona, digite o exemplo abaixo digitando em seguida TRON e depois RUN.

```
EXEMPLO 1
100 REM USANDO A INSTRUCAO TRON E TROFF
110 PRINT "AS TRES PRIMEIRAS LINHAS DESTE PROGRAMA"
120 TROFF
130 PRINT "SAO IMPRESSAS COM O MARCADOR LIGADO"
140 PRINT "ESTA LINHA E IMPRESSA COM O MARCADOR DESLIGADO"
150 END


EXECUÇÃO DO PROGRAMA
RUN
<100> <110> AS TRES PRIMEIRAS LINHAS DESTE PROGRAMA
<120> SAO IMPRESSAS COM O MARCADOR LIGADO
ESTA LINHA E IMPRESSA COM O MARCADOR DESLIGADO

VARIAÇÕES DE USO
Não se tem conhecimento de nenhuma.

VEJA TAMBÉM
TRON, NOTRACE, NOFLOW, TRACE OFF
```

Comando/Instrução **TRON**

SINTAXE GERAL

Número de linha, PRINT comando TRON

TRON pode ser usado como comando ou como instrução para ativar o marcador (traço). TRON é aceito no BASIC para microcomputadores que seguem a linha TRS-80.

O comando TRON liga a função TRACE e é utilizado para ativar uma ferramenta analítica que imprime números de linha de programa à medida que cada linha seja executada pelo computador. Essa característica é desligada pelos comandos TROFF ou NEW. TRON permite seguir o fluxo do programa para análise de execução e rastreamento de programa e para fins de detecção de erros.

Cada vez que o programa avança para uma nova linha, este número-de-linha aparecerá no vídeo entre parênteses.

```
EXEMPLO 1
100 REM USANDO TRON
110 GOTO 140
120 PRINT "DESTE PROGRAMA"
130 GOTO 160
140 PRINT "TRON PASSA CADA LINHA"
150 GOTO 120
160 PRINT "FIM DA EXECUCAO DO PROGRAMA"
170 END

EXECUÇÃO DO PROGRAMA
Tecle TRON antes de rodar o exemplo.

<110><140> TRON PASSA CADA LINHA
<150><120> DESTE PROGRAMA
<130><160> FIM DA EXECUCAO DO PROGRAMA
<170>
```

TRON pode ser utilizado também como instrução de programa para rastrear determinadas seções de programas. Para testar essa característica, tecle TROFF para ter certeza de que a característica de rastreamento esteja desligada, e em seguida acrescente ao exemplo 1 a seguinte linha, rodando-o em seguida:

```
125 TRON

EXECUÇÃO DO PROGRAMA
RUN
TRON PASSA CADA LINHA
DESTE PROGRAMA
<130><160> FIM DA EXECUCAO DO PROGRAMA
<170>

EXEMPLO 2
200 REM USANDO TRON
210 PRINT "LINHA 210"
220 INPUT "TECLE <ENTER> PARA INICIAR O LOOP";X
230 PRINT "AQUI VAMOS NOS..."
```

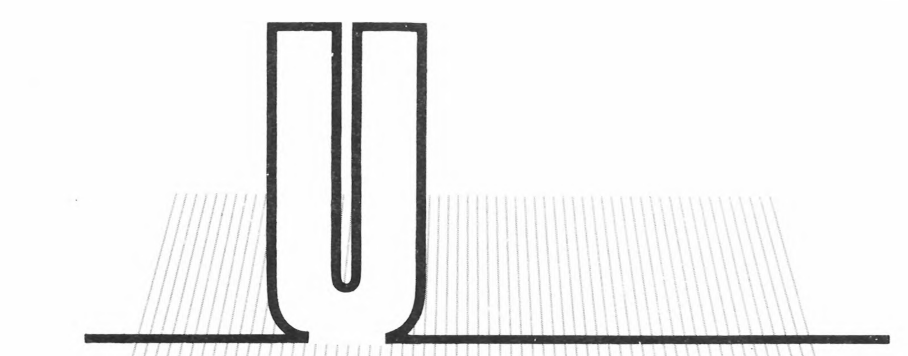
```
240 C = C +1
250 IF C = 2 THEN 270
260 GOTO 230
270 END
```

EXECUÇÃO DO PROGRAMA
Tecle TRON antes de rodar o exemplo.

```
<210> LINHA 210
<220> TECLE <ENTER> PARA INICIAR O LOOP?
<240><230> AQUI VAMOS NOS...
<230> AQUI VAMOS NOS...
<240><250><260><230> AQUI VAMOS NOS...
<240><250><270>
```

VARIAÇÕES DE USO
Não se tem conhecimento de nenhuma.

VEJA TAMBÉM
TROFF, NEW, FLOW, TRACE, TRACE ON



UNLOCK • USR

Comando/DOS **UNLOCK**

SINTAXE GERAL

UNLOCK nomearq [,Dn] [,Sn] [,Vn]

UNLOCK é um comando do DOS 3.3 utilizado nos microcomputadores que seguem a linha APPLE.

Este comando é usado quando se deseja desproteger um arquivo. Um arquivo protegido (travado) é mostrado no CATALOG por um asterisco (*)

Libera recursos bloqueados previamente. O operando do comando UNLOCK deve ser o mesmo citado em um comando LOCK anterior.

Quando se quer regravar ou eliminar um arquivo protegido, remove-se o LOCK com o comando UNLOCK.

As especificações de drive, slot e volume são opcionais e podem ocorrer em qualquer ordem.

EXEMPLO 1

```
100 HOME
110 D$ = CHR$(4)
120 PRINT D$; "OPEN NOMES"
130 PRINT D$; "WRITE NOMES"
140 PRINT "CLOVIS"
150 PRINT "ROSSANA"
160 PRINT D$
170 PRINT "OS NOMES FORAM PARA UM"
180 PRINT "ARQUIVO TEXTO CHAMADO"
190 PRINT "NOMES NO DISQUETE"
200 PRINT D$;"CLOSE NOMES"
```

Rode o programa e corrija erros se houver, depois digite:
SAVE CRIA NOMES
e dê um CATALOG
agora dê
LOCK CRIA NOMES
LOCK NOMES
Verifique com CATALOG
Depois use
UNLOCK CRIA NOMES
UNLOCK NOMES
e verifique o resultado dando CATALOG.

Se o arquivo que se quer proteger não existe, aparecerá uma mensagem de erro.

É um comando DOS e requer PRINT e CHR$(4) em modo programado.

VARIAÇÕES DE USO
Não se tem conhecimento de nenhuma.

VEJA TAMBÉM
LOCK

Função **USR**

SINTAXE GERAL

PRINT USR(X)

USR(X) é uma função reconhecida pela maioria dos interpretadores das várias linhas de microcomputadores e quando executada chama uma sub-rotina em linguagem de máquina armazenada e, então, continua a execução do programa em BASIC.

USR(X) onde X pode ser um número ou expressão e deve ter um valor entre -32767 e +32767, inclusive.

A rotina de linguagem de máquina pode ser lançada na memória a partir do teclado mediante a utilização da instrução POKE ou a partir de fita magnética ou disquete, pelo uso do comando SYSTEM.

Função USR pode ser utilizada em programas, à semelhança de qualquer outra função "embutida"; geralmente é precedida por uma instrução ou comando PRINT ou LET.

"Linguagem de máquina" é a linguagem de baixo nível, usada internamente pelo microcomputador. Consiste de instruções do microprocessador.

Sub-rotinas de linguagem de máquina são úteis para aplicações especiais (não executáveis em BASIC) e aplicações simples, desta forma, muito mais rapidamente.

EXEMPLO: Limpar o vídeo

Escrever tais sub-rotinas requer familiaridade com a programação em linguagem de máquina e com as instruções do microprocessador do seu microcomputador (estude o manual do seu microcomputador e verifique as instruções específicas desta função especial (USR) na sua máquina).

100 PRINT USR(X) irá imprimir o valor que é calculado pela rotina de linguagem de máquina do usuário.

Sendo armazenada na memória do computador uma rotina em linguagem de máquina para computar a raiz quadrada de X, o computador irá imprimir a raiz quadrada do número X.

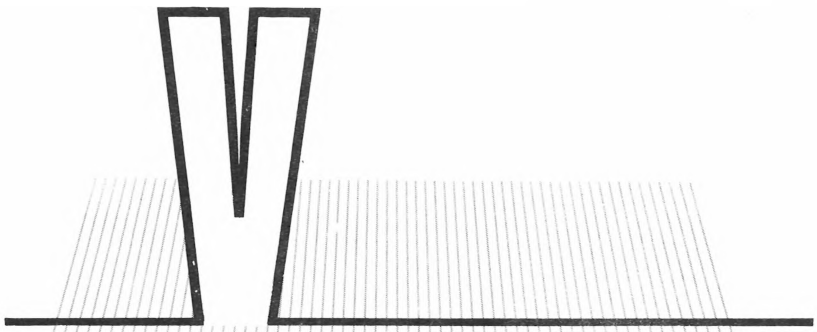
Para testar a função USR, deve-se carregar no computador uma rotina de linguagem de máquina (nos endereços apropriados), utilizando-se a instrução POKE ou o comando SYSTEM.

ORTOGRAFIA ALTERNATIVA

Alguns microcomputadores utilizam USER no lugar de USR.

VEJA TAMBÉM

POKE, SYSTEM



VAL • VERIFY
VARPTR • VLIN-AT • VTAB

Função **VAL**

SINTAXE GERAL

Número de linha, variável, função VAL (string)

VAL (string) é uma função reconhecida pela maioria dos interpretadores BASIC.

VAL (string) é uma função que pode ser utilizada no modo direto ou no modo programa devendo, entretanto, ser precedida por uma instrução geralmente PRINT ou LET.

A função VAL é utilizada para converter números escritos em "strings" de volta para a notação numérica. VAL tem o efeito de eliminar as aspas ou o sinal de dólar.

EXEMPLO 1

```
100 REM USANDO A FUNCAO VAL
110 R$="1985"
120 PRINT VAL(R$)
130 END
```

Imprime o número 1985 como valor numérico.

Esta função é muito útil porque podemos ter um número transformado em string, fazer as manipulações necessárias e depois transformá-lo de novo em um número (valor numérico).

EXEMPLO 2

```
200 REM USANDO AS FUNCOES STR$ E VAL
210 A = -85
220 B = -19
230 A$ = MID$(STR$(A),2)
240 B$ = MID$(STR$(B),2)
250 C$ = B$ + A$
260 C = VAL(C$)
270 PRINT C
280 END
```

EXECUÇÃO DO PROGRAMA

```
RUN
195
```

ADVERTÊNCIA: Alguns BASIC não interpretam um string tal como " -41"(sinal de mais ou menos precedido por espaços em branco) como valor numérico. Havendo problema no seu computador, o 0 será exibido ao se executar PRINT VAL (" -41").

VARIAÇÕES DE USO

Alguns microcomputadores, tais como as variantes Microsoft, aceitam em conjunto com a função VAL conjuntos de números e letras. Caso contrário, a função VAL produzirá um zero, como sinal de que não encontrou um número como primeiro carácter.

PRINT VAL ("519ABC"), imprime o número 519

EXEMPLO 3

```
100 REM USANDO A FUNCAO VAL COM STRING MISTO
110 R$="14 HORAS"
120 R=VAL(R$)
130 PRINT "SE O STRING ";R$;" FOR CONVERTIDO AO NUMERO ";R
140 PRINT "A FUNCAO VAL ACEITOU NUMEROS COM LETRAS".
150 END
```

EXECUÇÃO DO PROGRAMA

```
RUN
SE O STRING 14 HORAS FOR CONVERTIDO AO NUMERO 14
A FUNCAO VAL ACEITOU NUMEROS COM LETRAS
```

VEJA TAMBÉM

STR$, ASC, CHR$, LEN, LEFT$, MID$, RIGHT$, STRING$

Função **VARPTR**

SINTAXE GERAL

VARPTR (nome da variável)

VARPTR é uma função utilizada por algumas versões BASIC (tais como o BASIC Microsoft) para localizar o endereço de uma variável na memória. Esta função é reconhecida pelos microcomputadores que seguem a linha TRS-80 e geralmente quando usada deve ser precedida por uma instrução PRINT ou LET.

LET A = VARPTR (X)

atribui a A o endereço da primeira célula de memória utilizada para armazenar o valor da variável X. Para verificar o que é que está armazenado em A, tecle PRINT PEEK (A). A significação dada àquilo que se encontra depende do tipo de variável representada por X.

Sendo X uma variável de número inteiro, o endereço A retém o byte menos significativo (LSB) do valor de X. A localização A + 1 retém o byte mais significativo (MSB) do valor de X. VARPTR funcionará corretamente, se MSB*256 + LSB=X.

```
EXEMPLO 1
100 REM USANDO A FUNCAO VARPTR
110 A% = 23654
120 A = VARPTR(A%)
130 PRINT "VARPTR FOI ACEITO SE";
140 PRINT PEEK(A+1)*256+PEEK(A);"IGUAL A";A%
150 END

EXECUÇÃO DO PROGRAMA
RUN
VARPTR FOI ACEITO SE 23654 IGUAL A 23654
```

Quando esta função for do tipo VARPTR (variável de tipo inteiro), ao retornar o endereço - que aqui podemos representar por K, ocorrerá o seguinte: o endereço K conterá o byte menos significante (LSB) de um número inteiro de 2 bytes e o endereço K + 1 conterá o byte mais significante (MSB) do número inteiro. Você pode ver estes bytes (os dois em representação decimal) executando um PRINT PEEK (K) e um PRINT PEEK (K + 1)

Quando esta função for do tipo: VARPTR (variável de simples precisão), ao retornar o endereço K:

(K) = LSB (byte menos significante) do valor

(K + 1) = o byte seguinte, que é o mais significante do valor (MSB)

(K + 2) = MSB seguinte, sendo que o bit mais significativo é o símbolo do número.

(K + 3) = expoente do valor por excesso de 128 (128 é adicionado ao expoente).

Se a função for do tipo: VARPTR (variável de dupla precisão), ao retornar K:

(K) = LSB do valor

(K + 1) = MSB seguinte

(K +..) = MSB seguinte

(K + 6) = MSB seguinte, sendo que o bit mais significante é o símbolo do número.

(K + 7) = expoente do valor por excesso de 128 (128 é adicionado ao expoente).

Para valores de simples e dupla precisão, o número é armazenado em forma exponencial, de modo que o decimal é assumido antes do MSB. O valor 128 é somado ao expoente. Além disto, o bit mais alto de MSB é usado como bit de sinal. Ele é colocado para 0, se o número é positivo, ou, para 1, se o número é negativo.

Você pode ver estes bytes executando o apropriado PRINT PEEK (X), onde X = ao endereço que você quer ver. Lembre-se: O resultado será a representação decimal do byte, com o sétimo primeiro bit (do MSB) sendo usado como bit de sinal. O número estará na forma exponencial com o decimal posicionado antes do MSB. O valor 128 é adicionado ao expoente.

Se a função for do tipo: VARPTR (variável alfanumérica), ao retornar K, teremos:

K = número de caracteres do alfanumérico.
(K + 1) = LSB do endereço inicial do valor alfanumérico.
(K + 2) = MSB do endereço inicial do valor alfanumérico.

O endereço estará, provavelmente, na parte mais alta da RAM, onde são reservados os espaços para armazenar alfanuméricos. Mas se o seu alfanumérico é uma constante (uma literal alfanumérica), então ele irá apontar para a área de memória onde a linha de programa com a constante está armazenada; portanto na área do "buffer" do programa. Uma instrução de programa como A$ = "PALAVRA", não usa o espaço para armazenamento de alfanuméricos.

Para todas as variáveis acima, os endereços (K-1) e (K-2) armazenarão o código do carácter do nome da variável.

O endereço (K-3) conterá um código que informará ao computador qual é o tipo da variável. O código, para número inteiro, é 02; para simples precisão, é 04; para dupla precisão, é 08 e para alfanumérico é 03.

VARPTR (variável de matriz)

Retornará o endereço do primeiro byte deste elemento da matriz. O elemento consistirá de 2 bytes, se ele é um número inteiro; 3 bytes, se é um elemento alfanumérico; 4 bytes, se é um elemento de simples precisão e 8 bytes se o elemento é de dupla precisão.

O primeiro elemento na matriz é precedido por:

1 - Uma seqüência de dois bytes por dimensão, sendo que cada par de dois bytes indica o número de elementos de cada dimensão, respectivamente.

2 - Um byte, indicando o número total de dimensões na matriz.

3 - Dois bytes, indicando o número total de elementos na matriz.

4 - Dois bytes, contendo o nome da matriz em código ASC II.

5 - Um byte, para descrever o tipo da matriz, através de códigos (02 = números inteiros; 03 = alfanuméricos; 04 = simples precisão; 08 = dupla precisão)

Os elementos da matriz são armazenados seqüencialmente.

VEJA TAMBÉM
PEEK, POKE, USR, DEFINT, %

Comando/DOS **VERIFY**

SINTAXE GERAL

VERIFY nomearq [,Dn] [,Sn] [,Vn]

VERIFY é um comando do DOS 3.3 (Sistema Operacional para Disquete) utilizado pelos microcomputadores que seguem a linha APPLE II.

O comando VERIFY permite testar se um arquivo foi escrito num disquete corretamente e que o DOS pode ainda lê-lo.

Sendo um comando DOS requer PRINT e CHR$(4) em modo programado.

nomearq especifica o arquivo que você quer verificar. Qualquer tipo de arquivo pode ser verificado, incluindo arquivos-texto e arquivos binários.

O DOS apresenta no vídeo FILE NOT FOUND se o arquivo não existe.

Quando um arquivo é guardado em disquete com SAVE, BSAVE ou PRINT, é calculado um dígito de verificação para cada setor, que também é gravado no disquete.

O VERIFY recalcula este dígito e o compara com o gravado. Se eles coincidirem, o arquivo está intacto e nada aparece na tela. Se ocorrem erros em um ou mais setores, a mensagem I/O ERROR é gerada. Pode ser verificado qualquer tipo de arquivo.

OBSERVAÇÃO: Poeira, cinza de cigarro, ímãs quando em contato com partes visíveis do disquete podem prejudicar arquivos já gravados em disquetes.

VARIAÇÕES DE USO

Não se tem conhecimento de nenhuma.

Instrução **VLIN-AT**

SINTAXE GERAL

Número de linha, VLIN1, lin2, AT col
ou
número de linha VLIN AT coluna

VLIN...AT é uma instrução utilizada geralmente no modo programa e reconhecida nas versões BASIC (APPLESOFT e INTEGER) para os microcomputadores que seguem a linha APPLE. Esta instrução quando executada desenha uma linha vertical no vídeo na forma gráfica de baixa resolução em AT.

O comprimento da linha vertical é determinado por dois números que seguem a instrução VLIN. Esses números indicam os limites entre os quais a linha se vai estender. A linha pode ter qualquer comprimento entre as fileiras 0 e 39.

O número que segue AT é o número da coluna que a linha deve ocupar. Esse número pode variar de 0 a 39.

VLIN 5,30 AT 20, por exemplo, instrui o microcomputador para riscar uma linha vertical desde a fileira 5 até a fileira 30 AT (na) coluna 20.

A instrução GR deve ser executada antes que o microcomputador aceite a instrução VLIN-AT. A cor da linha é determinada pela instrução COLOR executada por último. Se a tela do vídeo estiver no modo texto, ou se a janela de texto estiver presente com a linha maior que 39, parte ou a totalidade da linha aparecerá como caracteres em vez de pontos.

EXEMPLO 1

```
100 REM USANDO A INSTRUCAO VLIN-AT
110 GR
120 B=0
130 FOR A=0 TO 39
140 COLOR = B
150 VLIN 0,39 AT A
160 B=B+1
170 IF B<16 THEN 190
180 B=0
190 NEXT A
200 END
```

EXECUÇÃO DO PROGRAMA

Se o computador tiver aceito a instrução VLIN-AT, a tela deve estar preenchida com 39 linhas verticais de diversas cores, se o vídeo for a cores.

VARIAÇÕES DE USO

Na versão INTEGER BASIC lin1 deve ter valor menor ou igual a lin2 ou então aparecerá a mensagem de erro "RANGE ERR".

VEJA TAMBÉM

GR, COLOR, PLOT, HLIN-AT, TEST

Instrução **VTAB**

SINTAXE GERAL

Número de linha, VTAB número (de 1 a 24)

VTAB número pode ser utilizada como comando no modo direto ou como instrução no modo programa.

VTAB é reconhecida pelos interpretadores BASIC de microcomputadores que seguem a linha APPLE II.

VTAB (tabulação vertical) é utilizada para especificar a localização vertical na tela de instrução PRINT ou INPUT com função de PRINT. Os valores VTAB desde 1 até 24, que representam as 24 linhas da tela, serão aceitas.

EXEMPLO 1

```
100 REM USANDO A INSTRUCAO VTAB
110 PRINT "DE ENTRADA NUM VALOR VTAB ENTRE 1 E 24";
120 INPUT R
130 VTAB R
140  PRINT  "VTAB FOI ACEITA SE ESTA MENSAGEM FOR IMPRESSA NA LINHA ";R
150 END
```

EXECUÇÃO DO PROGRAMA

```
RUN
DE ENTRADA NUM VALOR DE VTAB ENTRE 1 E 24? 12
VTAB FOI ACEITO SE ESTA MENSAGEM FOR IMPRESSA NA LINHA 12
```

EXEMPLO 2

```
200 REM USANDO AS INSTRUCOES VTAB E HTAB
210 HOME
220 VTAB 5:HTAB 1
230 INPUT "NOME: ";N$
240 INPUT "ENDERECO: ";E$
250 INPUT "BAIRRO: ";B$
260 VTAB 8:"HTAB 1
270 INPUT "CEP: ";CE$
280 VTAB 8:HTAB 12
290 INPUT "CIDADE: ";C$
300 VTAB 8:HTAB 32
310 INPUT "ESTADO: ";ET$
320 HOME
330 VTAB 5
340 PRINT N$
350 PRINT E$
360 PRINT B$;:HTAB 20:PRINT C$
370 PRINT CE$;:HTAB 20: PRINT ET$
380 END
```

EXECUÇÃO DO PROGRAMA

```
Digite seu nome e endereço.
```

SE O SEU COMPUTADOR NÃO TIVER ESSA FUNÇÃO

A maneira mais fácil de fazer com que a impressão comece um

certo número de linhas para baixo na tela é inicialmente limpar a tela mediante uma extensa série de instruções PRINT, uma após outra, ou mediante uma série de "avanços" ASCII ou CLS ASCII (limpar tela).

Examine a sua tabela ASCII para encontrar o "n" certo para PRINT CHR$(N).

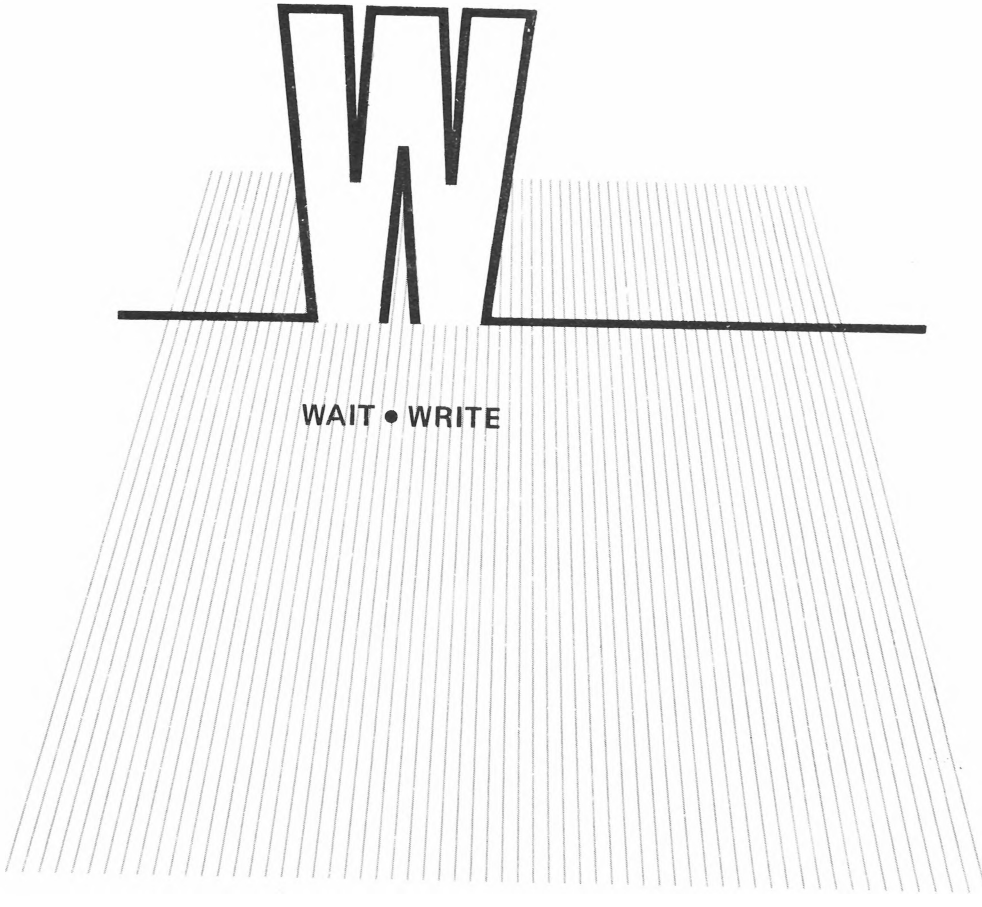
Em seguida, utilizando mais uma vez PRINTs ou um carácter ASCII, vá baixando na tela no número desejado de linhas, antes de imprimir.

VARIAÇÕES DE USO

Não se tem conhecimento de nenhuma.

VEJA TAMBÉM

TAB, PRINT-AT, HTAB, PRINT, ASC, CHR$, HOME, LIN


WAIT • WRITE

Instrução **WAIT**

SINTAXE GERAL

WAIT endmem, exprnm1, exprnm2
ou
WAIT X,Y,Z

WAIT é uma instrução reconhecida na versão BASIC APPLESOFT em microcomputadores que seguem a linha APPLEII.

A instrução WAIT endmem, exprnm1, exprnm2 quando executada verifica parte ou total dos oito bits da posição de endmem, para testar o padrão de 1 e 0 especificado por exprnm1.

O valor binário representado de exprnm1 determina quais os bits da posição de memória devem ser considerados e quais os que devem ser ignorados.

Se um bit da exprnm1 for 1, então o bit correspondente da posição endmem será testado.

O WAIT irá ignorar todos aqueles bits de memória que são correspondentes a 0 no valor binário de exprnm2.

A instrução WAIT quando executada:- Enquanto os bits significativos (especificados na exprnm1) de endmem forem todos diferentes do valor correspondente dos da exprnm2, haverá uma espera. No instante em que qualquer par for idêntico (ou 1 ou 0) a espera termina e a execução do programa em APPLESOFT continua.

A instrucão WAIT só pode ser interrompida por RESET, CTRL-RESET ou desligamento do microcomputador.

Não sendo especificado exprnm2, será adotado 0.

Os valores das expressões numéricas exprnm1 e exprnm2 devem estar na faixa de 0 a 255; valores fora desta faixa provocarão o aparecimento da mensagem de erro "QUANTIDADE ILEGAL".

EXEMPLO

WAIT X,Y,Z espera até que o conteúdo da localização X quando comparado (OR) com Z e comparado (AND) com Y, dê resultado diferente de 0 (zero).

OBS: Esta instrução não é disponível na versão INTEGER BASIC da linha APPLE II.

VARIAÇÕES DE USO

WAIT é utilizado por alguns microcomputadores americanos e europeus (os quais aceitam a versão MAX BASIC) para suspender a execução do programa durante um período especificado.

WAIT 20, por exemplo, instrui o microcomputador para esperar 20 segundos antes de executar a próxima instrução.

Alguns outros microcomputadores esperam por uma fração (1/10 ou 1/1000) do tempo especificado.

VEJA TAMBÉM

INP, FOR...NEXT, OR, AND

Comando/DOS **WRITE**

SINTAXE GERAL

WRITE nomearq [,Rn] [,Bn]

OBS: Estamos nos referindo ao comando WRITE para os micros da linha APPLE.

WRITE nomearq é um comando do DOS utilizado em arquivos seqüenciais ou de acesso direto.

Nos micros que seguem a linha APPLE, especifica um arquivo em disco para onde as saídas subseqüentes serão mandadas. É um comando DOS, e requer PRINT e CHR$(4) em modo programado.

Não pode ser usado em modo imediato.

Se o arquivo especificado não está aberto, ele o é, através do OPEN. Ambos os comandos, OPEN e WRITE, devem se referir ao mesmo arquivo.

Uma vez executado, o comando WRITE faz com que qualquer comando PRINT, subseqüente, envie todos os caracteres para o disquete ao invés da tela de vídeo. Se o arquivo não estiver no disco, aparecerá uma mensagem de erro.

O arquivo será tratado como seqüencial se o parâmetro R estiver ausente. Num arquivo seqüencial, o parâmetro B especifica em que byte o WRITE inicia. Sem o parâmetro B, o WRITE inicia no primeiro byte do arquivo (byte 0).

Se o parâmetro R está presente, o arquivo será gravado como arquivo de acesso aleatório.

O WRITE é para o registro especificado pelo parâmetro R.

Num arquivo de acesso aleatório, o parâmetro B especifica em que byte dentro de um registro especificado o WRITE inicia. Sem o parâmetro B, o WRITE inicia no primeiro byte (byte 0) do registro.

O parâmetro B pode ser usado para gravar iniciando num ponto além do último carácter do arquivo (ou registro). Este dado pode ser lido (com READ), porém qualquer tentativa de ler bytes intermediários não usados vai gerar a mensagem OUT OF DATA.

Os números à frente de R e B devem ser inteiros na faixa de 0 até 32767. Se não especificados, 0 (zero) é usado.

Enquanto WRITE estiver em uso, todo carácter que o APPLE II mandaria normalmente para a tela será mandado para o disquete, incluindo pontos de interrogação gerados por INPUT e as mensagens de erro.

A operação WRITE (escrita no disquete) será cancelada se aparecer no programa qualquer comando do DOS precedido de PRINT CHR$(4) mesmo o PRINT CHR$(4) sozinho ou a instrução INPUT.

EXEMPLO 1

```
100 REM USANDO WRITE
110 D$ = CHR$(4)
120 PRINT D$; "OPEN ARQUIVO"
130 PRINT D$; "WRITE ARQUIVO"
140 PRINT "SULLIVAN"
150 PRINT "CLOVIS"
160 PRINT "ROSSANA"
170 PRINT "MICROCOMPUTADOR"
180 PRINT "D$"
185  PRINT "AS INFORMACOES ANTERIORES FORAM TODAS GRAVADAS NO
```

```
DISQUETE"
     190 PRINT D$; "CLOSE ARQUIVO"
```

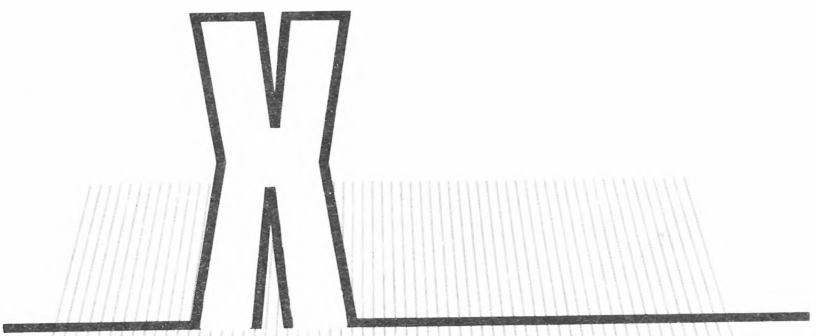
Salve este programa no disquete com o nome de CRIA ARQUIVO.

VARIAÇÕES DE USO

WRITE nomearq deve ser utilizado no modo de execução indireta, isto é, de dentro de um programa e sendo precedido por PRINT CHR$(4).

VEJA TAMBÉM

OPEN, READ nomearq, CLOSE



XDRAW

Instrução XDRAW

SINTAXE GERAL

XDRAW exprnm AT colh, linh

XDRAW é uma instrução utilizada geralmente no modo programa em microcomputadores que seguem a linha APPLE II.

É uma instrução reconhecida na versão BASIC APPLESOFT. Esta instrução quando executada desenha uma forma gráfica de alta resolução na tela, e se usada uma segunda vez com os mesmos parâmetros apaga o desenho sem apagar o fundo.

XDRAW exprnm AT colh, linh em que XDRAW verifica exprnm (número) da cor da coordenada desenhada e desenha a forma numa cor complementar:

CORES E DESENHOS

BRANCO	PRETO
VIOLETA	VERDE
LARANJA	VIOLETA
AZUL	LARANJA

XDRAW 2 AT 90,96 por exemplo, irá desenhar a configuração (da tabela de configurações do usuário) na linha 90 e coluna 96 da tela de vídeo.

XDRAW utiliza sempre o complemento da cor existente em cada ponto da respectiva configuração. De outra maneira, XDRAW tem as mesmas características de DRAW.

VARIAÇÕES DE USO.

Se não se especificar uma posição na instrução XDRAW a forma é desenhada iniciando na posição desenhada por último na instrução DRAW, XDRAW ou PLOT anterior. Se não especificar a posição, o desenho inicia na coordenada (colh,linh).

OBS: Esta instrução não é disponível para a versão INTEGER BASIC, da linha APPLE II.

VEJA TAMBÉM

DRAW

OPERADORES

" • @ • * • ** • # # • <> • ><
• / • $ • ! • ^ • = • ? • > • >=
=> • > • + • < • <= • =< • <
– • ( ) • % • . • ; • ,

Operador *

O operador (*) é o símbolo da multiplicação. É utilizado como sinal de multiplicação aritmética (em vez da letra "X") para encontrar o produto de dois números ou variáveis numéricas.

A operação é realizada com a precisão do mais exato operando.

EX.

```
PRINT 33 * 11 multiplicação de inteiros
PRINT 33 * 11.1 multiplicação de precisão simples
PRINT 12.34567890123456 * 11 multiplicação de precisão dupla
```

EXEMPLO 1

```
100 REM USANDO OPERADOR MATEMATICO *
110 X = 7
120 Y = X * 5
130 PRINT "* FOI ACEITO SE Y FOR ";Y
140 END
```

EXECUÇÃO DO PROGRAMA

```
RUN
* FOI ACEITO SE Y FOR 35
```

Alguns computadores utilizam o sinal * também como operador "matemático lógico" para "AND".

```
IF (X = 7) * (Y = 5) THEN 120
```

significa que "se o valor de X for igual a 7 e o valor de Y for igual a 5, a condição IF-THEN fica satisfeita e a execução prossegue na linha 120.

Observe que tanto (X=7) como (Y=5) estão entre parênteses.

Essa a "dica" para determinar se um "*" está sendo utilizado para efeitos de multiplicação ou na qualidade de AND lógico.

Operador **

O ** (asterisco duplo) é utilizado como sinal aritmético de exponenciação em alguns computadores americanos e europeus para computar o valor de um número de base numa potência determinada.

3**4 é a mesma coisa que o cubo de 3, ou seja, 3a4.

```
EXEMPLO 1
100 REM USANDO ** (EXPONENCIACAO)
110 PRINT "DE ENTRADA NUM NUMERO DE BASE";
120 INPUT Y
130 PRINT " DE ENTRADA NO EXPOENTE"
140 INPUT R
150 X = Y**R
160 PRINT "O NUMERO ";Y;" NA POTENCIA DE ";R;" E " X
170 END

EXECUÇÃO DO PROGRAMA
RUN
DE ENTRADA NUM NUMERO DE BASE? 3
DE ENTRADA NO EXPOENTE? 4
O NUMERO 3 NA POTENCIA DE 4 E 61
```

O ** (asterisco duplo) é também utilizado por alguns computadores que utilizam o BASIC Microsofft na instrução PRINT USING. O asterisco (*) pode ser impresso nos espaços não utilizados para a esquerda do ponto decimal de determinado número. O principal motivo para fazer isto é evitar que alguém aumente o montante de um cheque impresso no computador.

PRINT USING "*******.##";456.25 imprimirá ****456.25

O sinal de # representa os espaços designados para fins de impressão do valor numérico. Os espaços não utilizados são preenchidos com o sinal *.

Operador #

Cancela ## é utilizado para especificar variáveis como sendo de "dupla precisão". As variáveis de dupla precisão são capazes de armazenar números que contêm 17 dígitos (imprimindo-se apenas 16 dígitos). As variáveis de precisão simples têm um grau de exatidão de 6 dígitos.

O sinal ## deve ser colocado depois de uma variável para defini-la como sendo de dupla precisão, cada vez que essa variável seja utilizada no programa.

O operador ## é utilizado na instrução PRINT USING, pela maioria dos computadores que utiliza BASIC Microsoft, para indicar a posição de cada dígito numa variável numérica ou num número. Se a instrução PRINT USING contiver mais sinais ## do que dígitos num número, o computador imprime um espaço para cada sinal ## não utilizado, para a esquerda do ponto decimal, e um zero para cada sinal ## não utilizado, para a direita do ponto decimal.

EXEMPLOS:

Tem como função definir um campo numérico (um dígito por #).

###.### Tem como função definir a posição do ponto decimal.

+###.## Tem como função imprimir sinais dianteiros e traseiros.

+ para números positivos - para números negativos.

###.##- Tem como função imprimir sinal traseiro apenas se o valor impresso for negativo.

**###.## Tem como função preencher espaços em branco com asteriscos.

$$####### Tem como função colocar o sinal cifrão imediatamente à esquerda do primeiro dígito.

Os computadores com capacidade de manipulação de arquivos utilizam o operador ## em instruções tais como INPUT##, PRINT##, READ##, CLOSE##, para indicar um número de dispositivo destinado ao armazenamento de dados e para recuperar dados de dispositivos de memória externos tais como disquetes e fita cassete.

Operador <> • >< •

O sinal <> é utilizado como operador relacional "diferente", para fins de comparação de dois valores numéricos em instruções numéricas.

IF A <> B THEN 1000 manda o computador se desviar para 1000 se o valor da variável A for diferente do valor da variável B.

Todos os operadores relacionais têm a mesma precedência; eles são calculados da esquerda para a direita.

Os operadores relacionais permitem que se comparem dois valores para ver qual a relação entre um e outro. Os valores comparados podem ser constantes, variáveis ou qualquer tipo de expressão.

```
EXEMPLO 1
100 REM USANDO OPERADOR RELACIONAL <>
110 X = 15
120 IF X <> 25 THEN 150
130 PRINT "O SINAL <> NAO FOI ACEITO"
140 GOTO 160
150 PRINT "O SINAL <> FOI ACEITO"
160 END

EXECUÇÃO DO PROGRAMA
RUN
O SINAL <> FOI ACEITO
```

O sinal <> pode ser utilizado pela maioria dos computadores para comparar strings. O sinal <> compara o código ASCII de cada carácter (da esquerda para a direita) em dois strings. A primeira diferença de igualdade encontrada determina o respectivo relacionamento.

```
EXEMPLO 2
200 REM USANDO OPERADOR DE STRING <>
210 X$ = "ABCDE"
220 IF X$ <> "ABCD" THEN 250
230 PRINT "O SINAL <> NAO FOI ACEITO NA LINHA 220"
240 GOTO 300
250 IF X$ <> "ABCDE" THEN 270
260 GOTO 290
270 PRINT "O SINAL <> NAO FOI ACEITO NA LINHA 250"
280 GOTO 290
290 PRINT "O SINAL <> FOI ACEITO"
300 END

EXECUÇÃO DO PROGRAMA
RUN
O SINAL <> FOI ACEITO
```

Operador /

O símbolo / é utilizado para indicar a divisão aritmética e para se achar o cociente de duas variáveis numéricas.

Se um dos operandos for de precisão dupla, então ambos serão convertidos em precisão dupla e a divisão será realizada em 8 bytes.

Se nenhum dos dois for de precisão dupla, então ambos serão convertidos em precisão simples e a divisão será realizada em 4 bytes.

EX.

```
PRINT 3/4 divisão de precisão simples
PRINT 3.8/4 divisão de precisão simples
PRINT 3/1.2345678901234567 divisão de precisão dupla
```

EXEMPLO 1

```
100 REM USANDO SINAL DE DIVISAO /
110 X = 10
120 Y = 2
130 Z = X/Y
140 PRINT "Z = ";Z
150 PRINT "O SINAL / FOI APROVADO SE Z = 2"
160 END
```

EXECUÇÃO DO PROGRAMA

```
RUN
Z = 2
O SINAL / FOI APROVADO SE Z = 2
```

Operador $

O símbolo $, seguindo uma letra ou um conjunto de letras e números, é utilizado para indicar que a respectiva variável é de string.

EX.

A$; 22$

String é um tipo de variável que além de guardar números pode guardar qualquer carácter alfanumérico.

A variável é um string quando se acrescenta o sufixo $ ao nome da variável. Num programa as variáveis devem geralmente estar entre aspas.

EX.

A$ = "COMPUTADOR"

```
EXEMPLO 1
100 REM USANDO $ COM INSTRUCAO DE STRING
110 X$="APPLE"
120 PRINT "COMPUTADOR ";X$
130 END

EXECUCAO DO PROGRAMA
RUN
COMPUTADOR APPLE
```

É limitado pelo interpretador do computador o número de caracteres passíveis de serem atribuídos a uma variável string.

O tipo de nome de variável pode ser modificado acrescentando-se um símbolo de declaração no final.

EX.

R$ BA$ PAT$ COMP$

São todas variáveis string, sem se considerar quais os atributos que foram determinados às letras R, B, P e C.

A maioria dos computadores com capacidade para manipular strings permite a todas as letras do alfabeto servirem como designadoras de variáveis de string.

Muitos computadores com capacidade de manipular strings permitem combinações de letras, números e símbolos para especificarem variáveis de string e numéricos. Cada variável deve iniciar com uma letra, mas em muitos casos apenas os primeiros (geralmente 2) caracteres alfanuméricos são reconhecidos e processados pelo interpretador.

As palavras que forem instruções ou funções intrínsecas não podem ser utilizadas como variáveis numéricas.

Consulte o manual do seu computador para uma lista de palavras reservadas.

É possível fazer comparações de string. Podem ser comparados carácter por carácter, com outro string ou variável de string, que utilize operadores relacionais. Os strings devem estar entre aspas ao serem comparados com uma variável de string.

O APPLE II utiliza $ como prefixo para indicar um endereço de máquina e/ou um número hexadecimal.

EX. $8A = 138 decimal

Operador !

O ! (ponto de exclamação) declara que uma variável é de precisão simples.

As variáveis de precisão simples são capazes de armazenar cifras com mais de 7 dígitos (sendo impressos apenas 6 dígitos).

Com o sufixo ! ou 'E' temos variáveis de precisão simples com seis algarismos significativos.

EX.

A! 3.14159

O operador ! é utilizado em programas para alterar uma variável de volta à precisão simples. Envolve a conversão de um número com até 17 dígitos significativos, em um número com no máximo sete. O BASIC elimina os números menos significativos, para se conseguir um número de 7 dígitos. Antes da impressão de tal número o BASIC arredonda para 6 dígitos.

EX.

A! = 1.23456789012457

armazena

1.234567 em A!

No entanto, a instrução PRINT A! mostrará na tela o valor 1.23457, porque somente 6 dígitos serão impressos. Os sete dígitos são armazenados na memória.

A! = 2.77777777

armazena

2.77777 em A!

Quando as variáveis já são de precisão simples, o operador ! é utilizado em programas para alterar uma variável de volta para a precisão simples depois que tiver sido declarada de dupla precisão por uma instrução DEFDBL ou operador ## anteriores.

O operador ! é utilizado por alguns computadores na instrução PRINT USING para permitir que se imprima apenas o carácter mais para a esquerda de um string.

PRINT USING "!"; "SULLIVAN"

deverá imprimir a letra S.

Operador ^

O sinal ^ é utilizado como sinal de exponenciação aritmético para computar o valor de um número de base numa potência determinada.

A operação é geralmente precisa para 6 dígitos significativos.

O símbolo [também representa potenciação. Ele converte ambos os operandos em precisão simples e o resultado obtido também será de precisão simples.

```
EXEMPLO 1
100 REM USANDO ^ (EXPONENCIACAO)
110 PRINT "DE ENTRADA NUM NUMERO DE BASE ";
120 INPUT R
130  PRINT "EM SEGUIDA DE ENTRADA NUM NUMERO DE EXPONENCIACAO
";
140 INPUT P
150 T = R ^ P
160 PRINT "O NUMERO ";R;" NA POTENCIA DE ";P;" E ";T
170 END

EXECUÇÃO DO PROGRAMA
RUN
DE ENTRADA NUM NUMERO DE BASE? 5
EM SEGUIDA DE ENTRADA NUM NUMERO DE EXPONENCIACAO
? 4
O NUMERO 5 NA POTENCIA DE 4 E 625
```

Nos computadores que usam PRINT USING o sinal de ^ irá imprimir a forma exponencial ou 'E' de um número, ao se colocarem no string de formatação 4 ^s. Por exemplo, PRINT USING "##.###^^^^", 12345 imprime 12.345 E+03.

Alguns computadores utilizam ** para elevar determinado número a uma potência.

Operador =

O símbolo = pode ser utilizado como operador de atribuição, em expressões numéricas.

X = 7+3 atribui o valor 10 à variável X

Todos os operadores relacionais têm a mesma precedência; eles são calculados da esquerda para a direita. Os operadores relacionais permitem que se comparem dois valores para ver qual a relação entre um e outro. Os valores comparados podem ser constantes, variáveis ou qualquer tipo de expressão.

```
EXEMPLO 1
100 REM USANDO = COMO OPERADOR DE ATRIBUICAO
110 X = 5
120 Y = 10
130 Z = X+Y
140 PRINT "Z = ";Z
150 PRINT "O SINAL = FOI APROVADO SE Z = 15"
160 END

EXECUÇÃO DO PROGRAMA
RUN
Z = 15
O SINAL = FOI APROVADO SE Z = 15
```

O sinal = é utilizado também na maioria dos computadores como operador relacional para comparação dos valores numéricos do ponto de vista de sua igualdade.

IF A=B THEN 1000 indica ao computador que deve desviar o programa para a linha 1000 caso a variável numérica A seja igual à variável numérica B.

O sinal = é utilizado também para comparação de strings. Essa característica permite comparar um string ou variável de string, carácter por carácter, com outro string ou variável de string.

```
EXEMPLO 2
200 REM O SINAL = COMO OPERADOR DE COMPARACAO NUMERICA
210 X = 10
220 IF X = 10 THEN 250
230 PRINT "SINAL = NAO FOI APROVADO NA COMPARACAO NUMERICA"
240 GOTO 260
250 PRINT "SINAL = FOI APROVADO NA COMPARACAO NUMERICA"
260 END

EXECUÇÃO DO PROGRAMA
RUN
SINAL = FOI APROVADO NA COMPARACAO NUMERICA
```

Diversos interpretadores permitem comparar strings de caracteres de extensões diferentes. Alguns permitem apenas comparar uma letra com outra letra individual, ao passo que outros aceitam um número suficiente de caracteres para compararem um nome e endereços inteiros, ou até mais.

Operador ?

O ? (ponto de interrogação) é utilizado por muitos computadores como abreviatura de PRINT. A maioria (mas não todos) mudam automaticamente o sinal ? para a palavra PRINT, ao se LISTar o programa.

```
EXEMPLO 1
100 REM UTILIZANDO ? COMO PRINT
110 ? "O SINAL ? FOI APROVADO"
120 END

EXECUÇÃO DO PROGRAMA
RUN
O SINAL ? FOI APROVADO
```

O computador pode utilizar ? como indicador de INPUT, como sinal de que está esperando para que se dê entrada em alguns dados ou numa resposta.

```
EXEMPLO 2
200 REM UTILIZANDO ? COMO INPUT
210 PRINT "SINAL ? FOI APROVADO"
220 PRINT "SE A PROXIMA LINHA CONTIVER O SINAL ?"
230 INPUT X
240 END

EXECUÇÃO DO PROGRAMA
RUN
SINAL ? FOI APROVADO
SE A PROXIMA LINHA CONTIVER O SINAL ?
?
```

Pode-se utilizar o sinal ? em conjunto com o programa CLOAD para comparar um programa aramazenado na memória do computador com outro programa aramazenado em fita cassete.

Operador >

O sinal > é utilizado como operador relacional de "maior que", em expressões numéricas.

IF X > Y THEN 150 instrui o computador a se desviar para a linha 150 caso o valor da variável X seja superior ao da variável Y.

Todos os operadores relacionais têm a mesma precedência; eles são calculados da esquerda para a direita.

Os operadores relacionais permitem que se comparem dois valores para ver qual a relação entre um e outro.

Os valores comparados podem ser constantes, variáveis ou qualquer tipo de expressão.

```
EXEMPLO 1
100 REM USANDO OPERADOR RELACIONAL >
110 X = 25
120 IF X > 15 THEN 150
130 PRINT "O SINAL > NAO FOI APROVADO"
140 GOTO 160
150 PRINT "O SINAL > FOI APROVADO"
160 END

EXECUÇÃO DO PROGRAMA
RUN
O SINAL > FOI APROVADO
```

O sinal > pode ser utilizado na maioria dos computadores para comparar strings. O sinal > compara o código ASCII de cada carácter (da esquerda para a direita) entre dois strings. A primeira diferença de igualdade encontrada determina seu relacionamento.

O string "ABD" é maior do que o string "ABCDEF", embora o primeiro string tenha menor número de caracteres. Uma vez que o código ASCII de D (decimal 68) do primeiro string é superior ao que se segue, o código ASCII de C (decimal 67) no segundo.

Tendo cada string a mesma seqüência de caracteres, o string mais comprido é considerado o maior. O string "ABCD", por exemplo, é maior do que o string "ABC".

Os strings são comparados carácter por carácter, iniciando com o colocado mais à esquerda. O primeiro carácter de um string com o primeiro carácter do outro, o segundo com o outro, o terceiro com o outro, e assim por diante até que os dois strings terminem ou ocorra uma diferença.

Operador >= • => • >

O sinal >= é utilizado como operador relacional "maior ou igual a" para comparar dois valores numéricos (ou de string, onde permitido pelo computador), em instruções IF-THEN.

IF A >= B THEN 1000 instrui o computador a se desviar para a linha 1000 se o valor da variável A for maior do que ou igual ao valor da variável B. Todos os operadores relacionais têm a mesma precedência; eles são calculados da esquerda para a direita. Os operadores relacionais permitem que se compare dois valores para ver qual a relação entre um e outro. Os valores comparados podem ser constantes, variáveis ou qualquer tipo de expressão.

```
EXEMPLO 1
100 REM USANDO OPERADOR RELACIONAL >=
110 X = 25
120 IF X >= 150 THEN 150
130 PRINT "O SINAL >= NAO FOI ACEITO NA LINHA 120"
140 GOTO 190
150 IF X >= 25 THEN 180
160 PRINT "O SINAL >= NAO FOI ACEITO NA LINHA 150"
170 GOTO 180
180 PRINT "O SINAL >= FOI ACEITO"
190 END

EXECUÇÃO DO PROGRAMA
RUN
O SINAL >= FOI ACEITO
```

VARIAÇÕES DE USO

O operador >= é permitido por alguns computadores para efeitos de comparação entre strings. Compara o código ASCII de cada carácter, da esquerda para a direita, entre dois strings. A primeira diferença encontrada determina a relação.

```
EXEMPLO 2
200 REM USANDO OPERADOR DE STRING >=
210 X$ = "ABCD"
220 Y$ = "ABC"
230 Z$ = "B"
240 IF X$ >= Y$ THEN 270
250 PRINT "O SINAL >= NAO FOI ACEITO NA LINHA 240"
260 GOTO 340
270 IF X$ >= "ABCD" THEN 300"
280 PRINT "O SINAL >= NAO FOI ACEITO NA LINHA 270"
290 GOTO 340
300 IF Z$ >= Y$ THEN 330
310 PRINT "O SINAL >= NAO FOI ACEITO NA LINHA 300"
320 GOTO 330
330 PRINT "O SINAL >= FOI ACEITO"
340 END
```

Operador +

O operador (+) é o símbolo da adição; sua aplicação mais comum é para a adição aritmética. PRINT X + Y imprime a soma das variáveis X e Y.

A adição é feita com a precisão do mais exato operando, o menos exato é convertido. Quando um operando é do tipo inteiro e o outro de precisão simples, o inteiro é convertido em precisão simples e a adição é realizada em 4 bytes. Quando um operando é de precisão simples e o outro é de dupla precisão, o número de precisão simples é convertido em dupla precisão e a adição é realizada em 8 bytes.

EX.

```
PRINT 2 + 3  adição de inteiros
PRINT 3.1 + 3 adição de precisão simples
PRINT 1.2345678901234567 + 1 adição de precisão simples
```

Quando nenhum operador sinal aparecer na frente do termo numérico o BASIC deduzirá que seja o sinal positivo (+).

EXEMPLO 1

```
100 REM USANDO OPERADOR MATEMATICO +
110 PRINT "DE ENTRADA NUM VALOR PARA A VARIAVEL X";
120 INPUT X
130 PRINT "DE ENTRADA NUM VALOR PARA A VARIAVEL Y";
140 INPUT Y
150 Z = X + Y
160 PRINT "A SOMA DE ";X;"+";Y;" E ";Z
170 END
```

EXECUÇÃO DO PROGRAMA

```
RUN
DE ENTRADA NUM VALOR PARA A VARIAVEL X ? 7
DE ENTRADA NUM VALOR PARA A VARIAVEL Y ? 18
A SOMA DE 7 + 18 E 25
```

O Applesoft usa o sinal (+) para unir duas strings em uma. Este operador deve ser usado como parte de uma expressão string, na qual os operandos são strings e o valor resultante é um trecho de dados string. O operando (+) une o string à direita do sinal ao string à esquerda.

EX.

```
PRINT "INSTITUTO" + "SULLIVAN"
imprime
INSTITUTO SULLIVAN
```

O BASIC permite 255 caracteres, se o seu string for maior ocorrerá um erro.

Alguns computadores utilizam o sinal (+) como "OR" lógico numa instrução IF-THEN.

100 IF (X=10)+(Y=5) THEN 150 significa que se o valor de X for igual a 10 ou o valor de Y for igual a 5, é satisfeita a condição IF-THEN e a execução prossegue na linha 150.

Quando o sinal (+) for utilizado como operador OR lógico as equações deverão estar entre parênteses.

O sinal (+) é utilizado por alguns computadores em instruções PRINT USING para aplicar automaticamente um sinal (+) ou (-) a um número que vai ser impresso.

Operador <

O sinal < é utilizado como operador relacional "menor que", em expressões numéricas.

IF A < B THEN 1000 indica ao computador que se deve desviar para a linha 1000 se o valor da variável A for inferior ao valor da variável B.

Todos os operadores relacionais têm a mesma precedência; eles são calculados da esquerda para a direita. Os operadores relacionais permitem que se comparem dois valores para ver qual a relação entre um e outro. Os valores comparados podem ser constantes, variáveis ou qualquer tipo de expressão.

```
EXEMPLO 1
100 REM USANDO OPERADOR RELACIONAL <
110 X = 15
120 IF X < 25 THEN 150
130 PRINT "O SINAL < NAO FOI APROVADO"
140 GOTO 160
150 PRINT "O SINAL < FOI APROVADO"
160 END

EXECUÇÃO DO PROGRAMA
RUN
O SINAL < FOI APROVADO
```

O sinal < pode ser utilizado pela maioria dos computadores para comparar strings. O sinal < compara o código ASCII de cada carácter (da esquerda para direita) em dois strings. A primeira diferença encontrada determina a relação.

O BASIC compara expressões string, carácter por carácter. Quando ele encontra um carácter não coincidente, ele analisa qual aquele que tem o menor valor no código ASCII, isto é, o carácter de menor prioridade nos dois strings.

Se ao se efetuarem comparações, o BASIC chegar ao fim de um string antes de encontrar caracteres não coincidentes, o string mais curto será o de maior prioridade.

```
EXEMPLO 2
200 REM OPERADOR DE STRING <
210 X$ = "ABC"
220 Y$ = "ABCD"
130 Z$ = "B"
140 IF X$ < Y$ THEN 170
150 PRINT "O SINAL < NAO APROVOU NA LINHA 140"
160 GOTO 210
170 IF Y$ < Z$ THEN 200
180 PRINT "O SINAL < NAO APROVOU NA LINHA 170"
190 GOTO 200
200 PRINT "O SINAL < FOI ACEITO"
210 END

EXECUÇÃO DO PROGRAMA
RUN
O SINAL < FOI ACEITO
```

Operador <= • =< • <

O sinal <= é utilizado como operador relacional "menor ou igual a" para comparar dois valores numéricos em instruções numéricas.

IF A <= B THEN 1000 instrui o computador a se desviar para a linha 1000 se o valor da variavel A for menor que ou igual ao valor da variável B.

Todos os operadores relacionais têm a mesma precedência; eles são calculados da esquerda para a direita. Os operadores relacionais permitem que se comparem dois valores para ver qual a relação entre um e outro. Os valores comparados podem ser constantes, variáveis ou qualquer tipo de expressão. Estes operadores são úteis tanto para instruções IF...THEN como para aritmética lógica.

```
EXEMPLO 1
100 REM USANDO OPERADOR RELACIONAL <=
110 X = 15
120 IF X <= 25 THEN 150
130 PRINT "O SINAL <= NAO FOI ACEITO NA LINHA 120"
140 GOTO 190
150 IF A <= 15 THEN 180
160 PRINT "O SINAL <= NAO FOI ACEITO NA LINHA 150"
170 GOTO 190
180 PRINT "O SINAL <= FOI ACEITO"
190 END

EXECUÇÃO DO PROGRAMA
RUN
O SINAL <= FOI ACEITO
```

O sinal <= pode ser utilizado na maioria dos computadores para comparação de strings. O sinal <= compara o código ASCII de cada carácter (da esquerda para a direita) em dois strings. A primeira diferença encontrada determina a relação.

```
EXEMPLO 2
200 REM USANDO OPERADOR STRING <=
210 X$ = "ABC"
220 Y$ = "ABCD"
230 Z$ = "B"
240 IF X$ = Y$ THEN 270
250 PRINT "O SINAL <= NAO FOI ACEITO NA LINHA 240"
260 GOTO 340
270 IF X$ = "ABC" THEN 300
280 PRINT "O SINAL <= NAO FOI ACEITO NA LINHA 270"
290 GOTO 340
300 IF Y$ <= Z$ THEN 330
310 PRINT "O SINAL <= NAO FOI ACEITO NA LINHA 300"
320 GOTO 330
330 PRINT "O SINAL <= FOI ACEITO"
340 END
```

Operador —

O operador (-) é o símbolo da subtração aritmética para se encontrar a diferença aritmética entre dois números ou duas variáveis numéricas.

PRINT X - Y imprime o valor da variável X menos o valor da variável Y.

Assim como para a adição, a operação é realizada com a precisão do mais exato operando, o menos exato é convertido.

EX.

PRINT 33 - 11 subtração de inteiros

PRINT 33 - 11.1 subtração de precisão simples

PRINT 12.345678901234567 - 11 subtração de dupla precisão

Pode-se também usar o sinal (-) para indicar um valor numérico negativo. As operações são executadas na seguinte ordem: primeiro o negativo (-), depois a exponenciação, a multiplicação, a divisão e por último a adição.

EX.

-120/2 + 100 resulta em 40

O sinal - é utilizado para fins de negação em operações aritméticas. A negação significa mudar o sinal do atual para o contrário.

PRINT -(4-10) subtrai 10 de 4, dando 6 negativo. O primeiro - (sinal de negação) inverte o sinal entre os parênteses e imprime 6 (sendo implícito o sinal de +).

```
EXEMPLO 1
100 REM USANDO O SINAL -
110 X = 5
120 Y = 10
130 Z = Y - X -(Y-X)
140 PRINT "Z = ";Z
150 PRINT "O SINAL - FOI APROVADO SE Z = 0"
160 END

EXECUÇÃO DO PROGRAMA
RUN
Z = 0
O SINAL - FOI APROVADO SE Z = 0
```

O sinal - é utilizado por alguns computadores em instruções PRINT USING para ligar automaticamente um sinal - posterior a um número negativo que está sendo impresso.

Operador ()

Parênteses são utilizados em operações aritméticas para determinar a seqüência de realização de operações matemáticas. As operações matemáticas contidas entre parênteses se realizam antes das que ficam fora dos parênteses. Quando uma expressão completa incluir parênteses, o BASIC sempre dará prioridade as operações matemáticas contidas dentro deles. Quando houver parênteses superpostos, o BASIC começará a calcular de dentro para fora.

EX: X = 7 + (((2*5)-3)*2)

O computador realiza essa operação matemática de acordo com a seqüência assinalada abaixo:

X = 7 + ((10-3)*2)
X = 7 + (7*2)
X = 7 + 14
X = 21

Nos microcomputadores que seguem a linha APPLE pode-se trocar a ordem para resolver as expressões através do uso de parênteses. Quando mais de um conjunto de parênteses está presente, os micros da linha APPLE calculam da esquerda para a direita. Quando um conjunto de parênteses está envolvido por outro, e considerado como aninhado, é calculado primeiro o mais interno, e depois os subseqüentes.

É preciso utilizar parênteses com determinados "operadores matemáticos" para identificar as duas instruções que estão sendo comparadas. Elas são essenciais com TRS-80 Nivel 1.

EX: IF (X = Y) * (Y + 6) THEN 150

Operador %

O operador % declara que uma variável é inteira.

Sendo colocado o sinal % à direita de uma variável, essa variável passa então a ser capaz de armazenar unicamente valores de número inteiro.

Uma variável é inteira (números inteiros de -32769 a 32767) quando se usa o sufixo %.

EX.

A% = -10.5 atribui-se a A% o valor -11

A% = 3276.9 atribui-se a A% o valor 32767.

A% = 2.5D3 atribui-se a A% o valor 2500.

A% = -123.456789012345678 atribui-se a A% o valor -124.

A% = -32768.1 produz um erro de Overflow (fora da faixa dos inteiros).

```
EXEMPLO 1
100 REM USANDO % DE NUMEROS INTEIROS
110 I% = 3.678
120 IF I% = 3 THEN 150
130 PRINT "O OPERADOR DE NUMEROS INTEIROS % NAO FOI ACEITO"
140 GOTO 160
150 PRINT "O OPERADOR DE NUMEROS INTEIROS % FOI ACEITO"
160 END

EXECUÇÃO DO PROGRAMA
RUN
O OPERADOR DE NUMEROS INTEIROS % FOI ACEITO
```

O operador % é utilizado por alguns computadores na instrução PRINT USING.

Faz com que se imprimam num string tantos caracteres para a esquerda como os espaços existentes entre dois sinais %.

EX.

% %

é um string com comprimento igual a 2 mais o número de espaços entre os símbolos de percentagem.

EX.

PRINT USING "% %"; "SULLIVAN"

Deve imprimir as três primeiras letras, porque foi incluído um espaço entre os símbolos de percentagem. Dois sinais %/mais um espaço, igual a três letras.

Alguns computadores utilizam o sinal % na instrução PRINT USING, para assinalar um número como tendo ultrapassado os limites do especificador de campo (##).

Operador .

O ponto (.) é utilizado como ponto decimal por quase todos os microcomputadores, exceto os que têm apenas o BASIC de números inteiros.

O ponto é utilizado em algumas versões BASIC da MICROSOFT para fazer com que o computador LISTe ou EDITe a última linha de programa a que se deu entrada.

```
EXEMPLO 1
100 REM USANDO O PONTO
110 PRINT "O PONTO QUE SEGUE O COMANDO LIST"
120 PRINT "DEVE LISTAR A ULTIMA LINHA  QUE SE DEU ENTRADA"
150 END
```

EXECUÇÃO DO PROGRAMA

Tecle o comando LIST (caso omita o ponto após LIST, o programa inteiro será evidentemente LISTado). O computador deve imprimir

```
150 END
```

Acrescente a seguinte linha ao exemplo

```
140 PRINT "O PONTO FOI APROVADO"
```

Tecle o comando: EDIT.(inclusive o ponto)

Se o seu computador tiver essa capacidade de EDIT, o computador irá imprimir o número 140 seguido de um cursor. Isto indica que o computador está no modo EDIT, estando pronto para modificar a linha 140(última linha à qual se deu entrada).

VARIAÇÕES DE USO

Os computadores da linha TRS-80 nível 1 utilizam o ponto como parte das abreviaturas de palavras.

A letra I, por exemplo, é utilizada normalmente como variável, mas I. pode ser utilizada como abreviatura de INPUT ou INTeger, P.= PRINT, R.= RUN, L.= LIST etc.

VEJA TAMBÉM

EDIT, LIST, INPUT, INT

Operador ;

Ponto-e-vírgula (;) é um carácter especial do alfabeto BASIC e utilizado na declaração PRINT para juntar diversas seções impressas, numa só linha, causa o aparecimento do próximo carácter após o último carácter mostrado.

EX: PRINT "A"; "B"
produz um formato de saída fechado ou compactado.
AB

EXEMPLO 1

```
100 REM USANDO STRING COM PONTO-E-VIRGULA
110 PRINT "SE SULLIVAN FOR IMPRESSO ";
120 PRINT "NUMA SO LINHA, O TESTE APROVOU"
130 END
```

EXECUÇÃO DO PROGRAMA

```
RUN
SE SULLIVAN FOR IMPRESSO NUMA SO LINHA O TESTE APROVOU
```

Ao se utilizar o ponto-e-vírgula para separar a impressão de valores numéricos ou variáveis numéricas, um espaço deve ser inserido. Deve-se deixar um espaço adicional depois do número, uma vez que se admite haver sempre necessidade desse espaço.

O ponto-e-vírgula concatena cordões de dados mostrando itens, um atrás do outro sem espaços entre eles.

EXEMPLO 2

```
200 REM USANDO PONTO-E-VIRGULA
210 A = 7
220 FOR R = 1 TO 20
230 PRINT A;
240 NEXT R
250 PRINT "ACABOU"
260 END
```

EXECUÇÃO DO PROGRAMA

```
RUN
77777777777777777777ACABOU
```

Operador ,

Vírgula (,) é um operador com amplo uso de utilizações. Uma das aplicações mais comuns é a que acompanha a instrução PRINT; o aparecimento de vírgulas entre variáveis ou no final da declaração trata os dados como se estivessem tabelados em zonas predeterminadas.

Cada zona permite geralmente um máximo de dezesseis caracteres. O número de zonas permitido em cada linha varia de 4 a 8, dependendo da largura de linha da tela (ou da impressora).

```
EXEMPLO 1
100 REM USANDO VIRGULAS COM CORDOES
110 X$ = "FOGO"
120 Y$ = "TERRA"
130 Z$ = "AGUA"
140 FOR R = 1 TO 5
150 PRINT X$,Y$,Z$
160 NEXT R
170 END

EXECUÇÃO DO PROGRAMA
RUN
FOGO          TERRA         AGUA
FOGO          TERRA         AGUA
FOGO          TERRA         AGUA
FOGO          TERRA         AGUA
FOGO          TERRA         AGUA
```

A vírgula é utilizada também para separar elementos em campos de matrizes, em X (A,B,C). A vírgula separa A,B e C em elementos individuais dentro da matriz de três dimensões.

Pode-se utilizar a vírgula de maneira semelhante em DATA, DIM, INPUT, ON-GOTO e READ, para separar entre si dados individuais.

O Integer BASIC tem cinco paradas de tabulação na tela. São as colunas 1,9,17,25 e 33 existindo 8 espaços entre cada uma.

```
EXEMPLO 2
> 200 REM USANDO O OPERADOR VIRGULA
> 210 A = 150
> 220 FOR R = 1 TO 25
> 230 PRINT A,
> 240 NEXT R
> 250 PRINT "ACABOU"
> 260 END

EXECUÇÃO DO PROGRAMA
> RUN
150     150     150     150     150
150     150     150     150     150
150     150     150     150     150
150     150     150     150     150
150     150     150     150     150
ACABOU
```

Nos computadores da linha SINCLAIR pode-se escrever mais de uma coisa ao mesmo tempo, podendo-se separá-las com vírgulas(,) ou ponto-e-vírgula(;).

Se utilizar uma vírgula, então o próximo número aparecerá no vídeo começando ou na margem esquerda ou no meio da linha (16 coluna). Se utilizar um ponto-e-vírgula, então o número seguinte aparecerá no vídeo imediatamente ao último.

EXEMPLO

```
PRINT 1;2;3;4;5;6;7;8;9;10
```

e

```
PRINT 1,2,3,4,5,6,7,8,9,10
```

para ver as diferenças. Podem-se misturar vírgulas com pontos-e-vírgulas dentro de apenas uma instrução PRINT.

GLOSSÁRIO

COMANDOS, INSTRUÇÕES E FUNÇÕES PARA PROGRAMAÇÃO BASIC.

STOP - Causa a parada lógica da execução do programa e a mensagem BREAK IN aparece e informa o número da linha na qual o programa parou.

END - Causa a parada física do programa; não é colocada nenhuma mensagem.

NEW - Elimina o programa corrente e todas as variáveis a ele associadas na memória.

RUN - Carrega e executa um programa, começando pelo menor número de linha.

SAVE - Grava um programa da memória do microcomputador para fita cassete.

SAVE nomearq - Grava um programa da memória do computador para um disquete.

LOAD - Carrega um programa de fita cassete para a memória do computador

LOAD nomearq - Carrega um programa da memória do computador para um disquete.

CONT - Devolve a execução na próxima instrução após a parada, de um programa que foi parado por STOP, END ou CTRL-C.

CALL N - Pula para a sub-rotina em linguagem de máquina, iniciando na localização N da memória.

GOTO - Desvia a execução do programa para uma outra linha.

30 GOTO 60

Onde 30 é o número da linha e 60 é a linha para onde foi enviada a execução.

GOSUB XX - Desvia para a sub-rotina. Faz o programa pular para a linha indicada por XX.

RETURN - Manda o programa para a instrução após a última instrução GOSUB, executada. Fim da sub-rotina.

FOR...TO...STEP... Inicia um loop que repete uma série de instruções até que uma variável de incremento automático atinja um determinado valor.

NEXT - Determina o fim do loop, iniciado com a instrução FOR.

IF...THEN...ELSE... Define uma condição de instrução ou instruções do programa.

IF...GOTO - Desvia para outra linha do programa se houver a condição.

ON K GOTO XX,YY,...WW - Desvia para um dos vários pontos do programa, dependendo do valor corrente de uma expressão.

ON KGOSUB XX,YY...WW - Possibilita chamado opcional para uma das várias sub-rotinas num programa K.

ONERR GOTO - Pula para a linha de número indicado quando houver erro num programa.

CLEAR - Limpa todas as variáveis numéricas e strings. Coloca as variáveis em zero.

HOME - Limpa a tela e posiciona o cursor no canto superior esquerdo.

AUTO - Numera automaticamente as linhas, gera numeração automática.

TEXT - Aciona o modo texto, retorna a tela para o modo texto a partir de qualquer um dos modos gráficos.

INVERSE - Exibe os caracteres com cores invertidas.

NORMAL - Volta os caracteres ao modo normal usado após FLASH

ou INVERSE.

POP - Pula um endereço de RETURN, retorna ao comando que segue ao segundo mais recente GOSUB.

RESUME - Faz o programa voltar a execução no início da instrução onde o erro ocorreu.

TRACE - Lista cada número de linha de cada declaração executada.

NOTRACE - Usado para o acompanhamento da execução do programa iniciado com TRACE.

LOMEM - Determina o limite mais baixo de memória disponível para variáveis de programa.

HIMEM - Gera o limite máximo de memória disponível para programas, incluindo área de variáveis.

USR(X) - Passa o valor X para uma sub-rotina em linguagem de máquina.

WAIT - Pára a execução de um programa até que um endereço de memória tenha uma condição especificada.

PEEK - Função que permite ler o valor guardado em qualquer endereço da memória.

PEEK(X) Função que fornece o conteúdo da memória na localização X.

POKE X,12 - Muda o conteúdo da memória, na localização X, para o valor 12.

PDL - Retorna o valor corrente do controlador de jogo (paddle) especificado.

FRE - Indica quantos bytes estão livres para uso.

TRON - Aciona o modo de apuração de erros (TRACE).

TROFF - Desliga o TRON.

INP - Lê dados diretamente de uma porta de entrada ou saída.

OUT - Envia dados diretamente a uma porta de entrada ou saída.

INSTRUÇÕES E COMANDOS GRÁFICOS DE BAIXA RESOLUÇÃO

COLOR=X— Determina a cor a ser utilizada, gera cor para gráficos de baixa resolução.

GR - Converte a tela para gráfico de baixa resolução (40x40) deixando as quatro linhas de baixo para texto.

HLIN - Desenha uma linha horizontal na tela dentro do modo gráfico em baixa resolução.

X1,X2 AT Y — Desenha uma linha horizontal do ponto X1,Y ao ponto X2,Y.

PLOT X,Y - Mostra um ponto de tela gráfica de baixa resolução. X e Y variam de 0 a 39.

Desenha um elemento na tela (GR ou HGR).

VLIN - Desenha uma linha vertical na tela,na forma gráfica de baixa resolução.

SCRN(X,Y) - Coloca cor na tela no ponto X,Y. Retorna o código da cor do ponto gráfico de baixa resolução com as coordenadas especificadas.

INSTRUÇÕES E COMANDOS GRÁFICOS DE ALTA RESOLUÇÃO

HCOLOR - Fornece a cor para desenho em modo gráfico em alta resolução, e determina a cor (0 a 7).

HGR - Converte a tela em modo gráfico de alta resolução (260x160), mantendo quatro linhas embaixo livres para texto.

HGR2 - Converte a tela inteira para modo gráfico de alta

resolução (280x192), e coloca em modo gráfico de alta resolução a página 2.

HPLOT X,Y - Coloca ponto colorido na coordenada horizontal X e Y. X varia de 0 a 279; Y de 0 a 159.

DRAW - Desenha uma forma gráfica de alta resolução na tela, com a cor indicada por HCOLOR.

XDRAW - Desenha uma forma gráfica de alta resolução na tela, a cor de cada ponto plotado e complemento da cor da tela.

ROT - Muda a orientação do desenho da forma gráfica de alta resolução feito por DRAW ou XDRAW.

SCALE - Dá o tamanho da forma gráfica de alta resolução desenhada por DRAW ou XDRAW.

FUNÇÕES PARA OPERAR COM STRINGS

LEN(A$) - Fornece o número de caracteres de A$, determina o comprimento da string.

STR$(X) - String com valor numérico do argumento.

VAL(A$) - Converte um string em um valor numérico, fornece o valor numérico de A$, até o primeiro carácter não numérico.

ASC(A$) - Retorna o código ASCII de um carácter especificado. Fornece a A$ o código ASCII.

CHR$(4) - Carácter do codigo ASCII especificado.

LEFT$ - Fornece o carácter mais à esquerda do string.

MID$ - Parte central do string, retorna à posição especificada de um string.

RIGHT$ - Fornece os caracteres mais à direita de um string.

+ Sinal de operação para concatenar strings.

DIM - Reserva espaço na memória para um arranjo ou string.

DIM A(X,Y,Z) - Dimensiona o máximo espaço para A.

DIM A$(X,Y) - Dimensiona o máximo espaço para A$.

COMANDOS DE FORMATO

CTRL X - Apaga a linha sobre a qual está o cursor.

<--- Move o cursor um espaço para a esquerda, e apaga da memória o carácter sobre o qual passou.

---> Move o cursor um espaço para a direita, e coloca na memória o carácter sobre o qual passou.

LIST - Lista todo ou uma parte do programa que está na memória.

DEL - Elimina as linhas de programa especificadas.

REM - Usada para comentários a serem colocados no programa para efeito de documentação, não afeta o programa.

POS(0) - Retorna o cursor à posição horizontal normal (1 a 39).

SPC(X) - Nos comandos PRINT; coloca X espaços entre o último item impresso e o próximo.

FLASH - Faz com que os caracteres fiquem piscando no vídeo.

TAB(X) - Posiciona o cursor numa coluna específica na linha corrente.

HTAB(X) - Posiciona o cursor numa coluna especificada, para a posição X (1 a 40).

VTAB - Posiciona o cursor para uma linha especificada na coluna atual.

INSTRUÇÕES DE ENTRADA OU SAÍDA

INPUT - Aceita caracteres entrados pelo teclado ou outro dispositivo de entrada, calcula e associa o valor das variáveis (entrada de dados).

LET - É opcional; é uma declaração de atribuição; atribui um valor a uma determinada variável.

PRINT - Exibe dados, manda caracteres para a tela ou outro dispositivo de saída. O símbolo ? também significa PRINT. Itens separados por (;) são impressos em seqüência; itens separados por (,) em três campos tabulados.

IN#6 - Toma entrada do periférico que está no slot 6, de onde as entradas subseqüentes serão aceitas.

PR#6 - Dá saída ao periférico que está no slot 6, que irá receber as próximas saídas.

GET$ - Aceita um carácter único do teclado sem mostrar na tela. Espera que seja teclado um carácter para A$.

READ A$ - Lê o DATA. Assinala o próximo elemento DATA para A$.

RESTORE - Inicializa o DATA, coloca o apontador do DATA no início da lista.

DATA - Dados dentro de um programa. Cria uma lista de valores a serem usados pela declaração READ.

FUNÇÕES MATEMÁTICAS

ABS(X) - Fornece o valor absoluto de um argumento.

ABS(X) Dá o valor de X (valor positivo e absoluto de X). Se X for igual a -9; ABS(X)=9

SQR(X) - Fornece a raiz quadrada positiva de X; se X for um valor negativo ocorrerá uma mensagem de erro.

DEF FN - Define uma função.

INT - Função que permite obter a parte inteira de um número decimal ou fracionário.

SGN - Função que permite obter o sinal de uma expressão. O sinal é representado pelo valor -1, se o sinal for negativo, +1, se positivo e 0, se for nulo.

FUNÇÕES TRIGONOMÉTRICAS

ATN - Arcotangente em radianos.

COS - Co-seno fornece o co-seno em radianos.

SIN - Seno fornece o seno em radianos.

TAN - Tangente fornece a tangente em radianos.

São utilizadas com um parâmetro entre parênteses. O parâmetro representando a variável ou a expressão entre parênteses é expresso em radianos.

O parâmetro dessas funções pode ser expresso na forma de uma expressão aritmética qualquer, onde o resultado é um número que expressa valores em radianos.

FUNÇÕES EXPONENCIAIS E LOGARÍTMICAS

EXP(X) - Calcula o valor da base do logaritmo natural (e=2.71828...) elevado à potência X.

LOG(X) - Logaritmo natural de X; X deve ser um número maior do que zero.

NÚMEROS ALEATÓRIOS

RND - Número aleatório sucessivo de uma seqüência incorporada de números casuais uniformemente distribuídos.

RND(1) - Fornece um número aleatório (randônico) maior ou igual a zero e menor do que um.

RND(0) - Fornece o último número randônico.

OPERADORES MATEMÁTICOS

() Ordem de avaliação
   2*(4+5)=18
^ Exponenciação
   21^2=4
* Multiplicação
   5*2=10
/ Divisão
   9/3=3
+ Adição
   3+3=6
- Subtração
   9-6=3
= Igualdade
   8=3 falso
   8=8 verdadeiro
<> ou >< Diferente de A
   6<> 3 verdadeiro

OPERADORES RELACIONAIS

< Menor que
  1<3 verdadeiro
> Maior que
  1>3 falso
<= ou => Menor ou igual
  1<=3 verdadeiro
>= ou => Maior ou igual
  1>=3 falso
<> Diferente de

OPERADORES BOOLEANOS

NOT complemento lógico
    NOT(7) >= 6 verdadeiro
AND lógico
    5 AND (A$=B$)
OR lógico
   verdadeiro se:
   A$ = B$

COMANDOS, FUNÇÕES E INSTRUÇÕES POUCO CONHECIDAS E ACEITAS POR ALGUNS MICROCOMPUTADORES.

DOT - Função, indica se determinado bloco de gráficos na tela de vídeo está ou não ligado.

DET - Função, devolve o valor numérico único associado com uma matriz quadrada (isto é, um conjunto de duas dimensões tendo o mesmo número de linhas e de colunas).

EXAM - Função, lê o conteúdo de determinados endereços na memória do computador.

FETCH - Função, lê o conteúdo dos endereços na memória do computador.

FMT - Função, formata a saída de uma instrução PRINT.

MAT INV - Instrução, utilizada numa matriz quadrada, a fim de formar uma matriz que seja o inverso da matriz inicial.

MAT ZER - Instrução, atribui a cada elemento de um arranjo um valor zero.

MAT = Instrução, atribui os valores armazenados numa matriz às correspondentes células de outra matriz.

MAT + Instrução, adiciona os elementos correspondentes de duas matrizes do mesmo tamanho e armazena os resultados numa terceira matriz com as mesmas dimensões.

MAT - Instrução, substitui os elementos correspondentes de duas matrizes do mesmo tamanho e armazena os resultados numa terceira matriz.

MAT * Instrução, utilizada para multiplicar uma matriz.

MAT CON - Instrução, estabelece o valor de cada elemento de um arranjo (array) em alguma constante, tipicamente o número 1.

MAT IDN - Instrução, é utilizada numa matriz quadrada, para formar a matriz de identidade, isto é, uma matriz com 1 na diagonal e 0 em todos os demais lugares.

MAX - Instrução, utilizada para determinar qual de dois valores é o maior.

PAUSE - Instrução, exibe dados na janela.

PIN - Função, lê o valor decimal de um byte de informações em determinado ponto do computador.

REPET$ - Função, cria um string de caracteres que contém determinado conjunto de caracteres, repetindo um número especificado de vezes.

SEG$ - Função, extrai um segmento de um string.

SETDOT - Instrução, liga um bloco de gráficos na tela de raios catódicos.

SINH - Função, calcula o seno hiperbólico de um número.

SCRATCH - Comando, apaga o programa anterior.

SLEEP - Instrução, suspende a execução do programa durante um número determinado de décimos de segundo.

SKIPF - Comando, avança a fita cassete até o fim de um arquivo.

SPACE$ - Função, insere determinado número (N) de espaços.

STUFF - Instrução, utilizada para inserção de valores inteiros entre 0 e 255, para lugares específicos na memória.

SWAP - Instrução, permuta os valores de duas variáveis ou elementos de matriz.

TANH - Função, calcula a tangente hiperbólica de um número.

TLOAD - Comando, carrega um programa em fita cassete para o computador.

TSAVE - Comando, grava um programa do computador em fita cassete.

UNTIL - Instrução, utilizado como modificador.

XOR - Operador, é utilizado em instruções IF THEN como operador lógico.

WHILE - Instrução, executa uma série de comandos em um laço, até que determinada condição se torne falsa.

COMANDOS, INSTRUÇÕES E FUNÇÕES USADAS PELO MICROCOMPUTADOR SINCLAIR E SIMILARES.

ABS - Valor absoluto; torna o argumento positivo (módulo).
AND - E
ARCCOS - Arco de co-seno.
ARCSIN - Arco de seno.
AT - Em.
BREAK - Suspensão do programa.
CHR$ - Manda imprimir um carácter através de um número.
CLEAR - Limpa as memórias.
CLS - Limpa a tela de vídeo.
CODE - Código; número decimal do carácter.
CONT - Continue
COPY - Copia o conteúdo da tela.
COS - Co-seno.
DELETE - Apagar.
DIM - Dimensionar
EDIT - Editar.
ENTER - Nova linha.
EXP - Expoente.
FAST - Rápido.
FOR - Para.
FUNCTION - Função.
GOSUB - Vá para a sub-rotina que começa na linha...
GOTO - Vá para a linha...
GRAPHICS - Gráficos.
IF - Se.
INKEY$ - Leitura do teclado.
INPUT - Entre.
INT - Inteiro; anula a parte decimal de um número.
LEN - Número de caracteres da variável alfanumérica.
LET - Atribui.
LIST - Listar (dar a listagem do programa).
LLIST - Listar na impressora.
LN - Logaritmo natural.
LOAD - Chamar o programa (carregar o programa na memória).
LPRINT - Escrever na impressora.
NEW - Apaga a memória (para um novo programa).
NEWLINE - Nova linha.
NEXT - Voltar à linha do FOR e pegar o próximo valor.
NOT - Não.
OR - Ou.
PAUSE - Pausa.
PEEK - Mostra o valor de um determinado byte.
PLOT - Põe um ponto em X e Y.
POKE - Dá um valor (hexadecimal) a um byte determinado.
PRINT - Imprime (escreve na tela).
RAND - Inicialização da rotina RND.
REM - Comentário.
RETURN - Instrução de fim e retorno de sub-rotinas.
RND - Número casual ou aleatório.
RUBOUT - Apagar.
RUN - Executar (rodar-correr o programa).
SAVE - Armazenar (guardar).
SCROLL - Move toda tela (vídeo) uma linha para cima.
SNG - Sinal do argumento de um número.
SHIFT - Quando e apertada a tecla SHIFT e ao mesmo tempo uma

tecla de letra será introduzido um outro símbolo ou segunda função que depende do teclado.

SIN - Seno.

SLOW - Lento.

SPACE - Espaço.

SQR - Raiz quadrada.

STEP - Passo (incremento).

STOP - Parada lógica do programa.

STR$ - "STRING"; transforma um valor numérico em alfanumérico.

TAB - Tabulação (tabula a impressão).

TAN - Tangente.

THEN - Então.

TO - Até.

UNPLOT - Tirar um ponto de X e Y.

USR - Chama a sub-rotina em linguagem de máquina.

VAL - Valor da variável alfanumérica.

π - PI.

** - Elevado a...

IF...THEN... - Quando o computador encontra esta instrução ele testa a expressão que está entre IF...THEN...

Esta expressão pode ser verdadeira ou falsa. Se for verdadeira o computador executa a instrução que vem depois do THEN. Se for falsa o computador ignora a instrução depois do THEN e executa a próxima linha de programação.

IF...THEN...ELSE... - A instrução que vem depois de ELSE é a instrução que o computador deve executar caso a expressão seja falsa. Portanto se no comando IF...THEN... você não usa ELSE e a expressão for falsa ela será executada na próxima linha do programa.

Se usar ELSE o computador executa a instrução que vem após o ELSE.

Comandos, funções e instruções utilizadas com o DOS (sistema operacional para disquete) em microcomputadores que seguem a linha TRS-80 assim como o CP-500 e outros.

APPEND - Anexa arquivo-fonte arquivo-destino. Acrescenta o arquivo1 no fim do arquivo2. O arquivo-fonte não é alterado, o arquivo-destino é ampliado para incluir o arquivo-fonte.

ATTRIB - Modifica a senha de um arquivo. ATTRIB arquivo (I ou N),(ACC=nome1), (UPD=nome2), (PROT=nível).

(I ou N) = arquivo invisível (I) ou não invisível (N)

ACC = senha de acesso.

UPD = senha atualização do arquivo.

PROT = senha de acesso.

AUTO - Comando, automático após o acionamento do sistema. Executa o comando a cada reinicialização do DOS-500.

BACKUP - Copia o conteúdo do disquete fonte no disquete destino.

BASIC (ENTER) - Carrega o DOS-500, estando em DOS-500 ativo.

BASIC* - Coloca o sistema em BASIC READY, recuperando um programa que estava na memória.

BUILD - Cria um arquivo de introdução automática com comandos do DOS-500. CLEAR não pode ser usado num arquivo DO.

CLEAR - Limpa a memória do usuário, não pode ser usado em arquivo DO.

CLOCK - Ativa visualização do relógio, controla a visualização da hora real no canto superior direito da tela.

CLOSE - Fecha o acesso a um arquivo, através de um ou mais buffers especificados.

CLS - Limpa a tela e coloca no modo de 64 caracteres por linha.

COPY - Copia de um ou mais arquivos; copia de um arquivo-fonte para um arquivo-destino.

COPY arquivo d: copia o arquivo do drive d0 para o drive d1.

COPY/EXT - Copia todos os arquivos que tenham uma extensão compatibilizante, não importando o nome do arquivo. Copia todos os arquivos do d0 para o d1.

CREATE - Pré-aloca espaço em disquete para um novo arquivo.

CREATE nome do arquivo [LRL=aaa,REC=bbb]

LRL=aaa comprimento de uma gravação. aaa número decimal entre 0 e 255.

REC=bbb número de gravações a serem armazenadas. bbb número de gravações desejadas.

DATA - Atualiza a data interna.

DATA mm/dd/aa especificação do mês, dia e ano.

DEBUG - Comando, permite a detecção de erros em linguagem de máquina.

DIR - Mostra o diretório do disquete.

d - diretório do drive desejado.

INV - Lista os arquivos invisíveis do usuário.

SYS - Lista arquivos de sistema e de usuário.

PRT - Lista o diretório na impressora.

DO - Executa os comandos de um arquivo BUILD. Este comando lê e executa as linhas armazenadas em um arquivo de formato especial criado com o comando BUILD.

DUAL - Copia toda saída de vídeo na impressora. A impressora deverá estar preparada quando executar o comando.

DUMP - Grava um programa em um arquivo de disco. Os parâmetros são dados em hexadecimal.

DUMP - Arquivo [START=aaaa, END=bbbb,TRA=cccc,RELO=dddd]

START=aaaa - Endereço inicial bloco de memória, aaaa número hexadecimal.

END=bbbb - Endereço final bloco de memória, bbbb número hexadecimal.

TRA=cccc - Endereço de transferência, cccc número hexadecimal.

RELO=dddd - Endereço inicial para relocar ou carregar o programa de volta à memória.

EOF - Função, verifica se teve acesso a todos os caracteres até o fim do arquivo.

FIELD - Organiza um buffer aleatório de arquivo em campos.

FORMAT - Inicializa um disquete, criando um diretório.

FORMS - Determina os parâmetros da impressão, número máximo de caracteres por linha e número de linhas por páginas.

FORMS [WIDTH=w,LINES=I]

WIDTH=w - Número máximo de caracteres por linha.

LINES=I - Número de linhas por paginas.

FREE -Mostra o mapa de alocação do disquete.

GET - Lê um registro do disquete, acesso aleatório.

HELP - Comando, explicação dos comandos do DOS-500. É fornecida a lista dos comandos disponíveis.

INIT - Cria uma nova entrada de arquivo no diretório.

INPUT# - Instrução, introduz seqüencialmente dados de um arquivo de disco.

INSTR - Função, possibilita a procura de um string contido em outro string.

KILL - Comando, apaga um arquivo ou um grupo de arquivos, liberando o espaço por eles ocupado.

LIB - Comando, lista o conteúdo de um arquivo do DOS-500.

LINE INPUT - Instrução, introdução de linhas pelo teclado.

LIST - Exibe o conteúdo de um arquivo.

LIST - Arquivo [PRT,SLOW,ASCII]

PRT - Listagem através da impressora.

SLOW - Breve pausa.

ASCII - Lista o arquivo no formato ASCII.

LOAD - Comando, carrega na memória um arquivo em linguagem de máquina.

LOC - Função, número corrente de registros.

LOF - Função, número do registro de fim de arquivo.

LSET - Instrução, coloca dados em um campo do buffer do arquivo de acesso aleatório.

MASTER - Comando, especifica o drive mestre de leitura/gravação.

MERGE - Comando, combina programas em disquete com programa residente na memória do computador.

MID$ - Função, substitui uma parte de um string.

NAME - Comando, renumera o programa existente na memória RAM.

NEW - Comando, apaga o programa da memória.

OPEN - Instrução, abre um arquivo.

PATCH - Comando, modifica o conteúdo do arquivo em disquete. Todos os parâmetros devem ser apresentados em hexadecimal.

PATCH - Arquivo [ADD=aaaa,FIND=bb,CHG=cc]

ADD=aaaa - endereço no qual está a informação. aaaa parâmetro.

FIND=bb - string que se deseja procurar. bb seqüência hexadecimal.

CHG=cc - novo conteúdo dos bytes. cc seqüência hexadecimal.

PAUSE - Comando, interrompe o processamento e espera a.ação

do operador, usado em arquivo DO.

POSEOF - Rotina, posiciona o indicador de arquivo no último registro de arquivo.

POSN - Rotina, posiciona um arquivo lógico aleatoriamente escolhido.

PRINT - Comando, grava em disquete no modo seqüencial.

PROT - Modifica a senha mestra de todos os arquivos desprotegidos.

PROT: d [PW,LOCK]

d: especificação da drive; opcional

PW - Modifica a senha mestra.

LOCK - Atribui a senha mestra aos arquivos desprotegidos.

PUEGE - Comando, elimina arquivos de um disquete.

PURGE* - Comando, o asterisco mostra ao DOS-500, para perguntar se serão apagados todos os arquivos do disquete.

PUT - Comando, grava no disquete no modo de acesso direto.

PUTEXT - Rotina, coloca uma extensão no nome do arquivo.

RANDIR - Rotina, examina diretório de um disquete.

RELO - Comando, modifica o endereço do programa na memória. Não muda o programa em si.

RENAME - Comando, permite a redenominação de um arquivo ou de um programa. Muda o nome do arquivo1 para o arquivo2.

REWIND - Rotina, posiciona o indicador de arquivo no primeiro registro do arquivo.

ROUTE - Comando, muda os equipamentos de E/S.

ROUTE [SOURCE=aa,DESTIN=bb,]

SOURCE=aa - Equipamento de origem de E/S.

DESTIN=bb - Equipamento de E/S de destino.

DO - Tela.

PR - Impressora.

KB - Teclado.

RI - Entrada RS - 232

RO - Saída RS - 232

RSET - Instrução, coloca dados em um campo do buffer do arquivo de acesso aleatório.

RUN - Comando, carrega e executa um programa em disquete.

SAVE - Comando, armazena o programa num disquete.

SETCOM - Especifica parâmetros de inicialização de comunicação da interface.

TAPE - Comando, transferência da fita cassete para disquete, do disquete para fita e da fita para a memória RAM.

TIME - Comando, obtém ou atualiza as horas. hh:mm:ss (horas,minutos,segundos).

TESTMEM - Modifica o conteúdo da RAM.

USR - Função, transfere o controle para rotinas em linguagem de máquina.

WP - Comando, protege o conteúdo da gravação em disquete.

ENCICLOPÉDIA DA LINGUAGEM BASIC

Comandos, Instruções e Funções dos Equipamentos das Diversas Linhas

A Linguagem BASIC, simples e prática, é atualmente usada, nas suas diferentes versões, em todos os microcomputadores disponíveis no mercado.

Apresentar ao usuário os comandos, instruções e funções nos equipamentos das diversas linhas, esclarecendo-lhe as dúvidas e fornecendo-lhe os elementos indispensáveis para o máximo aproveitamento dos mesmos é a proposta desta ENCICLOPÉDIA DA LINGUAGEM BASIC que, além disso, inclui:

- numerosos exemplos, com o intuito de ilustrar os argumentos utilizados
- pequenos programas para testar o uso dos comandos
- sub-rotinas que facilitam o uso dos programas

O Professor Clóvis Pereira é Bacharel em Matemática e Analista de Sistemas, exercendo atualmente as funções de Diretor Executivo do Instituto Sullivan e Diretor-Geral da Sullivan Informática e Tecnologia.

Rossana B. Alcantara é Bacharel em Ciências Sociais e Programadora em BASIC e COBOL.

ISBN 85-7001-235-7