MICROCOMPUTADOR

EXATO MC-4000

MANUAL DE BASIC

EXATO

MANUAL DE BASIC

MICROCOMPUTADOR MC-4000

2ª EDIÇÃO

CCE - Indústria e Comércio de Componentes Eletrônicos S/A - DIVISÃO DE INFORMÁTICA
Av. Otaviano Alves de Lima, 2724
CEP 02501 - São Paulo-SP - Brasil

SUMARIO

CAPÍTULO 1

INÍCIO DE OPERAÇÃO

CAPÍTULO 2

DEFINIÇÕES

55- DEFINIÇÕES SINTÁTICAS

62- AVALIAÇÃO DE EXPRESSÕES

63- CONVERSÃO DE TIPOS

63- MODALIDADES DE EXECUÇÃO

CAPÍTULO 3

COMANDOS ESPECIAIS

CAPÍTULO 4

COMANDOS DE EDIÇÃO

No Capítulo 3, veja também CTRL C.

CAPíTULO 5

ARRANJOS E CADEIAS

CAPÍTULO 6

COMANDOS DE ENTRADA/SAÍDA

No capítulo 3, veja também LOAD e SAVE

No capítulo 5, veja também STORE e RECALL

101- INPUT

103- GET

104- DATA

106- READ

107- RESTORE

107- PRINT

108- IN#

109- PR#

110- LET

110- DEF FN

CAPITULO 7

COMANDOS DE DESVIO LÓGICO

CAPÍTULO 8

GRÁFICOS
E CONTROLADORES DE JOGOS

CAPÍTULO 9

FORMAS DE ALTA RESOLUÇÃO

CAPÍTULO 10

ALGUMAS FUNÇÕES MATEMÁTICAS

147- AS FUNÇÕES EMBUTIDAS : SIN (seno), COS (coseno), TAN (tangente), ATN (arcotangente), INT (inteiro), RND (randômica), SGN (função), ABS (absoluto), SQR (raiz quadrada) EXP (potência), LOG (logaritmo)

148- FUNÇÕES DERIVADAS

APÊNDICES

VISÃO GERAL

INTRODUÇÃO

CCE BASIC é a linguagem de programação residente do microcomputador EXATO . O BASIC foi ampliado porque existem muitas características no EXATO que não são disponíveis em outros computadores que usam o BASIC.

Entre as características suportadas pelo CCE BASIC estão as características gráficas e as entradas analógicas diretas (os controladores de jogos).

Outro aspecto do CCE BASIC é este manual. Ele não é um manual de auto-ensino ou de instrução programada, uma vez que a CCE fornece um outro manual (o Manual de Instrução), que o auxiliará a aprender programar mesmo que você nunca tenha tocado em um computador.

Este manual assume que você saiba programar em BASIC e somente deseja aprender as características adicionais oferecidas pelo CCE BASIC. O capítulo 1 fornece uma rápida visão do que esta linguagem tem a oferecer. O resto do manual é uma exata e cuidadosa descrição de todos os comandos da linguagem e de como cada um deles funciona. Para poupá-lo de frustação e aborrecimentos, este manual destaca os pontos onde erros de programação podem causar dificuldades.

O método usado para descrever o CCE BASIC é quase uma simples linguagem em si mesmo. Após pouco tempo de uso, sua compreensão sobre o que é legal e ilegal na linguagem, se acelerará. Os exemplos apresentados permitirão eliminar suas dúvidas.

Os programadores mais adiantados acharão este manual especialmente valioso, e os iniciantes em breve se adiantarão, e apreciarão o esforço extra que a CCE fez para fornecer um manual bastante completo.

USO DESTE MANUAL

Este manual de referência assume que você tenha um mínimo conhecimento da linguagem BASIC. Se você está desfamiliarizado com essa linguagem, consulte o Manual de Instrução.

Recomendamos que seu EXATO esteja instalado, e em funcionamento durante a leitura deste manual, de modo que você possa experimentar em seu computador tudo o que está descrito e sugerido. Se o CCE BASIC estiver ativo em seu sistema, o caráter "]" será exibido.

Existem dois termos que você precisará conhecer quando estiver lendo este manual. A palavra "sintaxe" refere-se à estrutura de um comando ao computador, a ordem e a forma correta das várias partes do comando. A expressão "análise gramatical" refere-se ao modo pelo qual o computador interpreta o que você digitou.

Por exemplo, a sintaxe do CCE BASIC permite que você digite

```
20 Y6 = 4 * 2^2
```

Quando o CCE BASIC analisa esta entrada, ele inicialmente toma 20 como o número da linha do programa, em seguida interpreta Y6 como nome de uma variável aritmética. Finalmente, analisa 2^2 como 4, e o multiplica por 4, determinando o valor 16 para a variável cujo nome é Y6.

O capítulo 1 fornece uma rápida visão dos diversos comandos CCE BASIC. Muitos conceitos elementares são apresentados, usando exemplos que você pode digitar. O apêndice B dá orientação na edição do programa. A notação apresentada no início do capítulo 2 é usada para descrever a sintaxe do CCE BASIC de maneira concisa e inequívoca. Ela poupará seu tempo e esforço na compreensão de como os comandos precisam ser estruturados. Por exemplo, colchetes ([e]) são usados para indicar porções opcionais de um comando; as chaves ({ e }) são usadas para indicar aquelas porções que podem ser repetidas. Assim:

```
[ LET ]  C = 3
```

indica que a palavra LET é opcional e pode ser omitida; e

```
REM  [{caráter}]
```

indica que o comando REM (comentário) consiste da palavra REM opcionalmente seguida por um ou mais caracteres.

As definições sintáticas e abreviações na primeira parte do capítulo 2 são apresentadas em uma ordem lógica para aqueles que querem ver como nós construímos nosso sistema de símbolos e definições. Você pode preferir ignorar estes símbolos e definições até encontrar um deles no texto. Nesse momento, você pode consultar o glossário alfabético de termos sintáticos, dado no apêndice N.

Os capítulos 3 a 10 apresentam explicações detalhadas dos comandos do CCE BASIC, agrupados por assunto. As informações adicionais não tratadas pelos capítulos podem ser encontradas nos apêndices.

INTERFERÊNCIA: RÁDIO/TELEVISÃO

O equipamento descrito neste manual gera rádio frequência, podendo causar interferência na recepção de rádio e televisão.

Você pode verificar se seu computador está causando interferência desligando-o. Se a interferência cessar, provavelmente ela é proveniente do mesmo. Neste caso, você pode eliminá-la pelo uso de uma ou mais das seguintes medidas:

- vire a antena do rádio ou TV até que a interferência pare;

- mova o computador para um lado ou outro da TV ou rádio;

- mova o computador para mais longe da TV ou rádio;

- Ligue o computador em uma tomada de uma rede diferente daquela da TV ou rádio (isto é, certifique-se que o computador e a TV ou rádio estejam em redes controladas por interruptores ou fusíveis diferentes).

Se necessário, consulte o revendedor ou um técnico experiente de rádio/TV para sugestões adicionais.

CAPÍTULO 1

INICIO DE OPERAÇÃO

MODO DE EXECUÇÃO IMEDIATA

Experimente digitar o seguinte:

```
PRINT 65 - 11
```

em seguida aperte a tecla [CR].

A resposta será dada imediatamente.

O comando PRINT que você digitou foi executado tão logo você apertou a tecla [CR] . O CCE BASIC avaliou a fórmula após o PRINT, e em seguida devolveu seu valor, neste caso "54".

Agora experimente digitar isto:

```
PRINT 3/2,3*11
```

("*" significa multiplique e "/" significa divida)

Quando você digitar a tecla [CR], o CCE BASIC exibirá:

```
 1.5          33
```

Como você pode ver, o CCE BASIC faz divisão e multiplicação, bem como subtração. Observe como uma vírgula (,) foi usada no comando PRINT para exibir 2 valores em lugar de apenas um. O uso da vírgula com o comando PRINT divide a linha de 40 caracteres em 3 partes ou pontos de tabulação. Veja a discussão das margens no capítulo 6, na seção "PRINT".

MODO PROGRAMADO

O modo através do qual PRINT foi usado na seção anterior é chamado "imediato". Entretanto, geralmente, o CCE BASIC é usado no modo "programado". Nesse segundo modo, todas as linhas de comando são iniciadas por um número inteiro. Esses números inteiros devem estar no intervalo entre 0 e 63999.

Experimente digitar as seguintes linhas:

```
10 PRINT 6 + 8
20 PRINT 6 - 8
```

(lembre-se que ao fim de cada linha é necessário digitar [CR].)

Uma seqüência de comandos no modo "programado" é chamada de "programa". Nesse modo, ao invés dos comandos serem executados imediatamente, eles são armazenados na memória do EXATO. Quando você digita RUN, o CCE BASIC executa em primeiro lugar o comando que tenha o mais baixo número de linha, em seguida o comando com o próximo maior número de linha, etc., até que o programa completo tenha sido executado.

Digitando agora RUN (lembre-se de teclar [CR] no fim de cada linha), será exibido na tela:

```
14
-2
```

No exemplo anterior, nós digitamos primeiro a linha 1Ø e em seguida a linha 2Ø. Todavia, não faz diferença em que ordem você digita os comandos no modo programado. Eles sempre serão colocados em ordem numérica crescente conforme seus números de linha.

Para ver uma listagem completa do programa atualmente na memória, com as instruções de acordo com a ordem crescente da numeração de linhas, digite:

```
LIST
```

E será exibido:

```
1Ø PRINT 6 + 8
2Ø PRINT 6 - 8
```

Algumas vezes é necessário apagar uma linha de um programa. Isto é feito pela digitação do número da linha que se queira apagar, seguida da digitação da tecla [CR].

Digite o seguinte:

```
1Ø
LIST
```

A resposta será:

```
2Ø PRINT 6 - 8
```

Você apagou a linha 1Ø do programa. Não há jeito de obtê-la de volta. Para inserir uma nova linha 1Ø, digite 1Ø seguido da nova instrução que você quer que o CCE BASIC execute. Por exemplo:

```
10 PRINT 6 * 8
LIST
```

e então você terá:

```
10 PRINT 6 * 8
20 PRINT 6 - 8
```

Há uma maneira mais fácil para alterar a linha 10 do que apagá-la e então inserir uma nova linha. Você pode fazer isto diretamente digitando a nova linha 10 (e apertando a tecla CR).

Automaticamente a velha linha 10 será substituída pela nova.

Agora digite o seguinte:

```
10 PRINT 6 - 6
LIST
```

A resposta será:

```
10 PRINT 6 - 6
20 PRINT 6 - 8
```

Não se recomenda que as linhas de um programa sejam numeradas consecutivamente: pode ser necessário, posteriormente, inserir uma nova linha entre duas já existentes. Geralmente se usa um incremento de 10 entre os números de linha.

Caso você queira apagar o programa completo, atualmente armazenado na memória, digite:

```
NEW
```

Se você está terminando a execução de um programa, e está prestes a começar outro programa, certifique-se de digitar NEW primeiro. Isto deve ser feito para evitar a mistura de programas velhos e novos.

Digite o seguinte:

```
NEW
```

Será exibido o caráter de prontidão:

```
]
```

Agora digite

LIST

A resposta será novamente o carácter de prontidão, (]) mostrando que seu programa anterior não está mais armazenado na memória.

Quando estiver desenvolvendo um programa ou entrando com dados no computador, antes de digitar NEW não se esqueça de salvar os dados em disco ou fita.

FORMATO DOS NÚMEROS

Faremos uma pequena pausa para explicar o formato dos números exibidos pelo CCE BASIC.

Os números são armazenados internamente com mais de nove dígitos de precisão. Quando um número é exibido, somente nove dígitos são mostrados. Todos os números podem ter um expoente (um fator de escala de potência de dez).

Em CCE BASIC números de "precisão real" (também chamados de "ponto flutuante") devem estar no intervalo de -1 * 1Ø^38 a 1 * 1Ø^38. Um número fora desse intervalo poderá ocasionar uma mensagem de erro. Usando adição ou subtração, você pode às vezes ser capaz de gerar números tão grandes quanto 1,7 * 1Ø^38 sem mensagem de erro.

Um número cujo valor absoluto seja menor do que 3 * 1Ø^-39 será convertido a zero.

Adicionalmente a estas limitações, números inteiros devem estar no intervalo de -32767 a 32767.

Quando um número é exibido, as seguintes regras são usadas para determinar o formato exato:

1) se o número é negativo, um sinal de menos (-) é exibido;

2) se o valor absoluto de um número é um inteiro dentro do intervalo de Ø a 999999999, ele é exibido como um inteiro;

3) se o valor absoluto de um número é maior ou igual a 0.01 e menor que 99999999.2, o número é exibido em notação com ponto fixo, sem expoente;

4) se o número não se enquadra nas categorias 2 ou 3, é usada a notação científica.

A notação científica é usada para exibir números com precisão real e é formada como segue:

SX.XXXXXXXESTT

onde cada "X" é um inteiro de 0 a 9. O "S" inicial é o sinal do número, nada para um número positivo e um sinal de menos (-) para um número negativo. Um dígito diferente de zero é exibido antes do ponto decimal. Este é seguido pelo ponto decimal e pelos oito dígitos da mantissa. Um "E" é então exibido (de Expoente), seguido pelo sinal "S" do expoente, e em seguida os 2 dígitos "TT" do expoente.

Zeros à esquerda nunca são exibidos, isto é, o dígito antes do ponto nunca é zero. Igualmente zeros não significativos não são exibidos.

Se existe somente um dígito para exibir depois que os zeros não significativos foram suprimidos, o ponto decimal não é exibido. O sinal do expoente será mais (+) para positivos e menos (-) para negativo. Os dois dígitos são sempre exibidos, isto é, não há supreção de zeros no campo do expoente.

O valor de qualquer número, expresso na forma de notação científica conforme descrito acima, é o número à esquerda do "E", vezes 10 elevado à potência do número à direita do "E".

Segue-se exemplos de vários números e o formato que o CCE BASIC usará para exibi-los.

NÚMERO	FORMATO DE SAÍDA
+3	3
-3	-3
2104	2104
-14.0600	-14.06

28.10E5	2810000
1 * 10^20	1E + 20
-25.54325096 * 10^10	-2.5543251E + 11
10000000000	1E + 09
999999999	999999999

Um número digitado, ou uma constante numérica usada num programa CCE BASIC, pode ter quantos dígitos for desejado, entretanto, somente os primeiros 10 dígitos são normalmente significativos, e o décimo dígito é arredondado.

Por exemplo, se você digitar:

```
PRINT 2.01234567890123
```

O CCE BASIC responde com

```
2.01234568
```

EXEMPLO DE GRÁFICO A CORES

Digite

```
GR
```

Isto fará apagar as 20 primeiras linhas de texto da tela e deixará somente as 4 últimas linhas de texto.O seu EXATO está agora no modo gráfico de Baixa Resolução.

Agora digite

```
COLOR = 13
```

Você terá o caráter de prontidão e o cursor, porém, internamente o EXATO registrou que foi selecionada a cor amarela.

Digite agora:

```
PLOT 20,20
```

O CCE BASIC responderá colocando um pequeno quadrado amarelo no centro da tela. Se o quadrado não for amarelo, seu vídeo não está sintonizado adequadamente: ajuste os controles de cor e saturação até conseguir um tom amarelo limão claro.

Agora digite

```
HLIN Ø,3Ø AT 2Ø
```

Surgirá na tela uma linha horizontal atravessando entre as colunas Ø e 3Ø, na 2Øª linha da tela.

Digite então

```
COLOR = 6
```

para mudar para uma nova cor, e em seguida

```
VLIN 1Ø,39 AT 3Ø
```

Você aprenderá mais sobre gráfico a cores em seções posteriores.

Para voltar ao modo texto, digite

```
TEXT
```

Os caracteres apresentados na tela é a maneira do EXATO mostrar informação de cores como texto.

EXIBIÇÃO NA TELA

Conforme explicado anteriormente, a inclusão de uma vírgula (,) em um comando PRINT causa um espacejamento para a próxima coluna tabulada, antes de exibir o valor que segue a vírgula.

Se usarmos o ponto e vírgula (;) em lugar da vírgula, o próximo valor será exibido imediatamente após o anterior. Digite os exemplos:

```
PRINT 1,2,3 [CR]

1          2          3
```

PRINT 1;2;3 [CR]

123

PRINT -1;2;-3 [CR]

-12-3

Observação: digite somente os trechos em itálico. Os demais valores são apresentados pelo computador.

A seguir, é apresentado um exemplo de programa, que lê um valor a partir do teclado, e usa o mesmo para calcular e exibir um resultado.

```
10 INPUT R [CR]
20 PRINT 3.14159 * R*R [CR]
RUN [CR]
?10 [CR]
314.159
```

Aqui está o que acontece. Quando o CCE BASIC encontra o comando INPUT ele exibe um ponto de interrogação na tela, e espera que você digite um número. Quando você o faz (no exemplo acima, 10), a variável que segue INPUT (R) é associada ao valor digitado.

Em seguida, na execução da linha 20, a fórmula após o comando PRINT é avaliada com o valor 10 substituindo a variável R, cada vez que R aparece na fórmula. Consequentemente, a fórmula se torna 3,14159 x 10 x 10 ou 314,159.

Se você ainda não tinha adivinhado, o programa acima calcula a área de um círculo com raio R.

Se quisermos calcular a área de vários círculos, nós podemos repetir a execução do programa para cada círculo sucessivo. Porém, há uma maneira mais fácil de fazê-lo, simplesmente acrescentando outra linha ao programa, conforme segue:

```
30 GOTO 10
```

Digite agora:

RUN

?10 [CR]

```
 314.159

 ?3 [CR]

 28.2741

 ?4.7 [CR]

 69.3977231

 ?
```

Ao colocar um comando GOTO 10 no fim do seu programa, você o fará voltar à linha 10 após exibir cada resposta. Isto poderia continuar indefinidamente. Para interromper o programa, digite CTRL-C (pressione [C] enquanto mantém apertada a tecla [CTRL]) e depois a tecla [CR]. Isto ocasiona uma "interrupção" na execução do programa. Usando o CTRL-C, quase todos os programas podem ser interrompidos após a execução da instrução corrente.

NOMES DE VARIÁVEIS

A letra R no programa executado na seção anterior foi denominada uma "variável". Isto é simplesmente uma posição de memória no computador identificada pelo nome R. Um nome de variável deve começar com um caráter alfabético, e pode ser seguido qualquer caráter alfanumérico. Um caráter alfanumérico é qualquer letra de A a Z, ou qualquer dígito de 0 a 9.

Um nome de variável pode ter até 238 caracteres, porém o CCE BASIC usa somente os 2 primeiros caracteres para distinguir um nome do outro. Assim, os nomes EDUARDO e EDSON referem-se a mesma variável.

Certas palavras usadas nos comandos do CCE BASIC são "reservadas" para sua finalidade específica. Você não pode usá-las como nomes de variáveis ou como parte do nome de variáveis. Por exemplo, COLORIDO seria ilegal porque COLOR é uma palavra reservada. As palavras reservadas do CCE BASIC estão relacionadas e comentadas no apêndice G.

Nomes de variáveis terminadas em $ ou % tem um significado especial, conforme discutido em seção posterior deste capítulo.

A seguir estão alguns exemplos de nomes de variáveis legais e ilegais.

LEGAL	ILEGAL
TP	TO (nomes de variáveis não podem ser palavras reservadas)
PSTE$	
COUNT	RGOTO (nomes de variáveis não podem conter palavras reservadas)
N1%	

Você pode associar valores para uma variável com o comando, INPUT, com o comando LET, ou simplesmente com o sinal de igualdade. Experimente os exemplos abaixo:

```
A = 4

PRINT A, A*2

4            8

LET  Z = 9

PRINT Z, Z - A

9            5
```

Observação: você deve digitar somente os trechos em itálico. Os demais trechos representam a resposta do computador.

Como pode ser visto pelos exemplos, o LET é opcional em um comando de definição de valor.

Os valores das variáveis são abandonados, e o espaço na memória usado para armazená-las é liberado quando um dos quatro fatos ocorrem:

1) uma nova linha é digitada no programa ou uma linha velha é apagada;

2) um comando CLEAR é emitido;

3) um comando RUN é emitido;

4) NEW é digitado.

Aqui está outro fato importante: até que você defina algum valor para elas, todas as variáveis numéricas são automaticamente definidas com valor igual a zero. Digite o exemplo:

```
PRINT Q, Q + 2, Q *2

Ø                2              Ø
```

REM

Outro comando é o REM. É a forma abreviada da palavra REMARK (comentário). Este comando é utilizado para inserir comentários ou observações dentro de um programa. Quando o CCE BASIC encontra um comando REM o resto da linha é ignorado. Isto serve principalmente como uma ajuda ao programador e não tem função alguma na execução do programa, ou na resolução de um problema específico.

IF...THEN

Vamos executar um programa para verificar se um número digitado é zero ou não. Com os comandos que vimos até aqui, isto não pode ser feito. O que precisamos é um comando que forneça um desvio condicional para outra instrução. O comando IF...THEN faz exatamente isto.

Digite NEW, e em seguida esse programa:

```
1Ø INPUT B

2Ø IF B = Ø THEN GOTO 5Ø

3Ø PRINT "NAO E ZERO"

4Ø GOTO 1Ø

5Ø PRINT "ZERO"

6Ø GOTO 1Ø
```

Quando este programa for executado, ele exibirá um ponto de interrogação e esperará que você digite um valor para B. Digite qualquer número que quiser. O computador irá então para o comando IF. Entre as porções IF e THEN do comando, há uma "asserção".

Uma asserção consiste de duas expressões separadas por um dos seguintes símbolos:

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

O comando IF é ou verdadeiro ou falso, dependendo se a asserção é verdadeira ou não. Em nosso programa, por exemplo, se Ø é digitado para B a asserção B = Ø é verdadeira. Consequentemente, o comando IF é verdadeiro e a execução do programa continua com a porção THEN do comando: GOTO 5Ø. Seguindo este comando o programa saltará para a linha 5Ø. A palavra "ZERO" será exibida e o comando na linha 6Ø mandará a execução de volta à linha 1Ø.

Suponha que um 1 seja digitado para B. Uma vez que a asserção B = Ø agora é falsa, o comando IF é falso e a execução do programa continua na próxima linha, ignorando a porção THEN do comando, bem como qualquer comando naquela linha. Consequentemente, "NAO E ZERO" será exibido, e o comando GOTO na linha 4Ø mandará a execução de volta à linha 1Ø.

Agora digite o seguinte programa para comparar 2 números (lembre-se de digitar NEW, para apagar seu último programa):

```
1Ø INPUT A,B

2Ø IF A <= B THEN GOTO 5Ø
```

```
30 PRINT "A E MAIOR"

40 GOTO 10

50 IF A < B THEN GOTO 80

60 PRINT "ELES SAO IGUAIS"

70 GOTO 10

80 PRINT "B E MAIOR"

90 GOTO 10
```

Quando este programa for executado, a linha 10 apresentará um ponto de interrogação e esperará que você digite 2 números separados por uma vírgula. Na linha 20, se A é maior que B, A <= B é falsa e THEN GOTO 50 é ignorado. A execução do programa então salta para o comando seguinte, exibindo "A E MAIOR", e finalmente a linha 40 desvia a execução do programa de volta à linha 10.

Na linha 20, se A tem o mesmo valor de B, A <= B é verdadeiro de modo que THEN GOTO 50 é executado, desviando o programa para a linha 50 e desde que A tenha o mesmo valor de B, A < B é falsa. Consequentemente, THEN GOTO 80 é ignorado e o programa continua na linha seguinte, onde lhe é dito para apresentar "ELES SAO IGUAIS". Finalmente, a linha 70 desvia a execução de volta ao início.

Na linha 20, se A é maior que B, A <= B é verdadeiro de sorte que a execução do programa prossegue com THEN GOTO 50. Na linha 50, A < B é verdadeiro de modo que THEN GOTO 80 é executado. Finalmente, "B E MAIOR" é exibido, e novamente o programa volta à linha 10.

Tente rodar os 2 últimos programas várias vezes. Em seguida, tente escrever seu próprio programa usando o comando IF...THEN. Realmente experimentar seus próprios programas é a maneira mais rápida e fácil de entender como o CCE BASIC funciona.

Lembre-se que, para interromper estes programas, deve-se digitar CTRL-C e aperte a tecla [CR].

NOVO GRAFICO A CORES

Verifique o programa a seguir. Note que o caráter dois pontos (:) é usado para separar múltiplas instruções em uma mesma linha numerada de programa. Após digitá-lo use o comando LIST, e certifique-se que ele foi digitado corretamente. Em seguida, execute-o, com o comando RUN.

```
100 GR: REM POSICIONA MODO GRAFICO EM CORES
110 HOME: REM LIMPA AREA DE TEXTO
120 X = 0: Y = 5: REM POSICIONA O INICIO
130 XV = 2: REM POSICIONA VELOCIDADE DE X
140 YV = 1: REM POSICIONA VELOCIDADE DE Y
150 REM CALCULA NOVA POSICAO
160 NX = X + XV: NY = Y + YV
170 REM SE A BOLA EXCEDER LIMITE DA TELA, ENTAO
    DESVIA
180 IF NX > 39 THEN NX = 39 : XV = -XV
190 IF NX < 0 THEN NX = 0 : XV = -XV
200 IF NY > 39 THEN NY = 39 : YV = -YV
210 IF NY < 0 THEN NY = 0 : YV = -YV
220 REM TRACE NOVA POSICAO EM AMARELO
230 COLOR = 13 : PLOT NX, NY
240 REM APAGUE POSICAO VELHA
250 COLOR = 0 : PLOT X, Y
260 REM SALVA POSICAO ATUAL
```

```
270 X = NX: Y = NY

280 REM PARA APOS 250 MOVIMENTOS

290 I = I + 1: IF I < 250 THEN GOTO 160

300 PRINT "PARA VOLTAR AO SEU PROGRAMA, DIGITE
    TEXT"
```

O comando GR diz ao EXATO para operar no modo gráfico de Baixa Resolução. Ele também limpa a área gráfica de 40 por 40 escurecendo-a, posiciona a saída do texto em uma janela de 4 linhas de 40 caracteres cada na parte inferior da tela, e fixa a próxima cor a ser utilizada como preta.

HOME é usado para limpar a área de texto e posicionar o cursor no canto esquerdo superior da janela de texto definida neste momento. No modo gráfico esta operação tem início na linha 20, uma vez que as linhas 0 até 19 estão agora sendo usadas como área de traçado de gráfico.

Os comandos COLOR = nas linhas 230 e 250 definem a próxima cor a ser traçada conforme o valor da expressão seguinte ao sinal de igualdade.

O comando PLOT NX, NY na linha 230 traça um pequeno quadrado, na cor definida pelo mais recente comando COLOR= (no caso, correspondente à cor amarela), na nova posição especificada pelas expressões NX e NY. Lembre-se de que NX e NY devem ser números no intervalo de 0 a 39, ou o quadrado estará fora da tela e haverá uma mensagem de erro.

De forma similar, PLOT X,Y na linha 250 apresenta um pequeno quadrado na posição especificada pelas expressões X e Y. Mas X e Y são simplesmente as coordenadas anteriores NX e NY, salvas após a apresentação do quadrado anterior.

Consequentemente, PLOT X,Y reapresenta o quadrado amarelo anterior com um quadrado cuja cor é definida por COLOR = 0. Esta cor é preta, a mesma cor do fundo, de modo que o quadrado amarelo anterior parece ser apagado.

Observação: Para sair do modo gráfico para o modo texto digite TEXT e em seguida aperte a tecla [CR].

Se você não entende a linha 29Ø, seja paciente. Isto será explicado mais adiante.

Como você tem visto o EXATO pode fazer mais do que apenas usar números. Nós voltaremos aos gráficos em cores depois que você tiver aprendido mais sobre o CCE BASIC.

FOR. . .NEXT

Uma das vantagens dos computadores é a sua capacidade para executar tarefas repetitivas.

Suponha que queiramos uma tabela de raízes quadradas, para os números inteiros de 1 a 1Ø. O comando para extrair a raiz quadrada no CCE BASIC é SQR; sendo sua sintaxe

```
SQR (X)
```

onde X é o número cuja raiz quadrada você deseja calcular. Nós poderíamos escrever o programa como segue:

```
1Ø PRINT 1, SQR (1)
2Ø PRINT 2, SQR (2)
3Ø PRINT 3, SQR (3)
4Ø PRINT 4, SQR (4)
5Ø PRINT 5, SQR (5)
6Ø PRINT 6, SQR (6)
7Ø PRINT 7, SQR (7)
8Ø PRINT 8, SQR (8)
9Ø PRINT 9, SQR (9)
1ØØ PRINT 1Ø, SQR (1Ø)
```

Este programa executará a tarefa, porém pode-se executar a mesma operação de modo bem mais simples usando comando IF recém apresentados, como segue:

```
10 N = 1

20 PRINT N, SQR (N)

30 N = N + 1

40 IF N <= 10 THEN GOTO 20
```

Quando este programa é executado, sua saída fica exatamente igual à do programa antecedente. Vejamos como ele funciona.

Na linha 10 há um comando que associa o valor 1 à variável N. Na linha 20, o computador é instruído para exibir N e a raiz quadrada de N, usando o valor corrente de N. Por exemplo, para N=1:

```
20 PRINT 1, SQR (1)
```

e o resultado deste cálculo é apresentado na tela.

Na linha 30, há o que parece ser a primeira vista uma operação um tanto incomum. Matematicamente, o comando N= N+1 não tem sentido, entretanto, um aspecto importante a ser lembrado é que em um comando LET, o símbolo "=" não significa igualdade. Neste caso "=" significa "ser substituido por". O comando simplesmente toma o valor corrente de N e soma 1 ao valor. Assim, após a primeira passagem pela linha 30, N torna-se 2.

Na linha 40, sendo N = 2; a asserção N <= 10 é verdadeira de modo que a porção THEN desvia o programa de volta à linha 20, agora com N valendo 2.

O resultado geral é que as linhas 20 a 40 são repetidas, cada vez somando-se 1 ao valor de N. Quando N chega a 10 na linha 20, a linha seguinte o aumenta para 11, o que resulta numa falsa asserção na linha 40. A porção THEN é consequentemente ignorada, e uma vez que não há comandos a seguir, o programa para.

Essa técnica é referida como "laço" (loop). Desde que seu uso é bastante intenso em programação, existem comandos especiais no CCE BASIC para usá-la.

anterior. Se não houver STEP em um comando FOR, o CCE BASIC assume que deve somar 1 cada vez.

O STEP pode ser seguido por qualquer expressão. Suponha que queiramos contar para trás, de 1Ø a 1. Um programa que faz isto é:

```
1Ø I = 1Ø

2Ø PRINT I

3Ø I = I - 1

4Ø IF I > = 1 THEN GOTO 2Ø
```

Observe que agora o programa verifica se I é maior ou igual ao valor final. A razão é que agora estamos subtraindo. Nos exemplos anteriores era o contrário, de modo que estávamos conferindo para ver se a variável era menor ou igual ao valor final.

O comando STEP, anteriormente mostrado, também pode ser usado com números negativos para atingir o mesmo objetivo. Verifique o programa abaixo:

```
1Ø FOR I = 1Ø TO 1 STEP -1

2Ø PRINT I

3Ø NEXT I
```

Os laços FOR também podem ter vários níveis. Segue-se um exemplo deste procedimento:

```
1Ø FOR I = 1 TO 5

2Ø FOR J = 1 TO 3

3Ø PRINT I, J

4Ø NEXT J

5Ø NEXT I
```

Observe que o NEXT J vem antes do NEXT I. Isto é porque o laço J está dentro do laço I.

O programa seguinte está incorreto. Execute-o e veja o que acontece.

```
10 FOR I = 1 TO 5

20 FOR J = 1 TO 3

30 PRINT I, J

40 NEXT I

50 NEXT J
```

Ele não funciona porque quando NEXT I é encontrado, todo o conhecimento do laço J é perdido.

MATRIZES

Frequentemente é necessário o uso de diversos números agrupados em uma tabela.

O CCE BASIC permite que isto seja feito pelo uso de matrizes. Uma matriz é uma tabela de números. O nome da tabela, chamado nome da matriz, é qualquer nome legal de variável, por exemplo A.

O nome da matriz A é distinto e separado do nome de uma simples variável A e você pode usar ambos e um mesmo programa.

Para selecionar um elemento da tabela, damos a A um índice, isto é, para selecionar o I - ésimo elemento colocamos o I entre parênteses o qual segue o nome da matriz A. Consequentemente, A (I) é o I- ésimo elemento da matriz A.

Observação: Nesta seção do manual nos preocuparemos somente com matrizes unidimensionais; para discussões adicionais de comandos CCE BASIC relativos a matrizes, veja o capítulo 5.

Antes de utilizar uma matriz, é necessário que você defina seu tamanho ou sua dimensão, o que é feito através do comando "DIM (X)". Por exemplo:

```
DIM A (15)
```

Neste caso, definimos que a dimensão de A, com os índices das matrizes sempre começam com Ø; portanto, no exemplo acima permitimos 16 elementos numéricos da matriz A.

Se A (I) é usado em um programa antes que a matriz tenha sido dimensionada, o CCE BASIC reserva espaço para 11 elementos (índices de Ø a 1Ø).

Como exemplo de uso de matrizes, execute o prograna seguinte, o qual ordena uma lista de 8 números digitados por você.

```
9Ø  DIM A(8): REM DIMENSIONA A MATRIZ COM NO MAXIMO
    DE 9 ELEMENTOS

1ØØ REM PEDE 8 NUMEROS

11Ø FOR I = 1 TO 8

12Ø PRINT "DIGITE UM NUMERO";

13Ø INPUT A(I)

14Ø NEXT I

15Ø REM PASSA ATRAVES DOS 8 NUMEROS, TESTANDO POR
    PARES

16Ø F = Ø: REM REPOSICIONA O INDICADOR DE ORDEM

17Ø FOR I = 1 TO 7

18Ø IF A(I) <= A(I+1) THEN GOTO 24Ø

19Ø REM INVERTE A(I) COM A(I+1)

2ØØ T = A(I)

21Ø A(I) = A(I+1)

22Ø A(I+1) = T

23Ø F = 1: REM ORDEM NAO ERA CORRETA

24Ø NEXT I

25Ø REM F = Ø SIGNIFICA ORDEM CORRETA

26Ø IF F = 1 THEN GOTO 16Ø: REM TENTA DE NOVO
```

```
270 PRINT: REM PULA 1 LINHA

280 REM APRESENTA OS NUMEROS ORDENADOS

290 FOR I = 1 TO 8

300 PRINT A(I)

310 NEXT I
```

Quando a linha 90 é executada, o CCE BASIC arruma espaço para 9 valores numéricos, A(0) até A(8). As linhas 110 até 140 obtém a lista de números desordenados do usuário. A ordenação dos números em si é feita pelas instruções contidas nas linhas 170 a 240, percorrendo-se a lista de números e invertendo quaisquer 2 números que não estejam ordenados. F é o indicador de ordem correta: F = 1 indica que uma troca foi feita. Se qualquer troca foi feita, a linha 260 faz com que o programa volte e confira. Se uma passagem completa é feita através dos 8 números sem inverter nenhum (significando que todos eles estavam ordenados), as linhas 290 a 310 apresentarão a lista ordenada.

Observe que um índice pode ser definido por qualquer expressão.

GOSUB...RETURN

Outro par de comandos muito útil é GOSUB e RETURN. Se seu programa executa a mesma ação em vários trechos diferentes, você pode usar os comandos GOSUB e RETURN para evitar a duplicação dos mesmos comandos.

Quando um comando GOSUB é encontrado, o programa é desviado para a linha cujo número segue a palavra GOSUB. Entretanto, o CCE BASIC registra a linha em que estava o programa antes do desvio.

Quando o comando RETURN é encontrado, o programa volta ao primeiro comando, seguinte ao último GOSUB executado. Considere o seguinte programa:

```
20  PRINT "QUAL E O PRIMEIRO NUMERO"

30  GOSUB 100

40  T = N REM SALVA A ENTRADA
```

```
5Ø  PRINT "QUAL E O SEGUNDO NUMERO"

6Ø  GOSUB 1ØØ

7Ø  PRINT "A SOMA DOS DOIS NUMEROS E "; T + N

8Ø  STOP : REM FIM DO PROGRAMA PRINCIPAL

1ØØ INPUT N: REM COMECO DA SUB-ROTINA DE ENTRADA

11Ø IF N = INT (N) THEN GOTO 14Ø

12Ø PRINT "DESCULPE, O NUMERO DEVE SER UM INTEIRO,
    TENTE DE NOVO"

13Ø GOTO 1ØØ

14Ø RETURN : REM FIM DA SUB-ROTINA
```

Este programa pede 2 números, que devem ser inteiros, e exibe a soma desses.

A sub-rotina neste programa está nas linhas 1ØØ a 14Ø. Ela pede um número, e se o número digitado em resposta não é um inteiro, pede um novo número.

O programa principal apresenta "QUAL E O PRIMEIRO NUMERO", e então chama a sub-rotina para obter o valor do número N. Após RETURN, o programa volta para a linha 4Ø. O número que foi digitado (N) é salvo na variável T. Isto é feito de modo que quando a sub-rotina é chamada uma segunda vez, o valor do primeiro número não é perdido.

"QUAL E O SEGUNDO NUMERO" é então apresentado, e a sub-rotina é chamada de novo, desta vez para obter o segundo número. Quando a sub-rotina retorna a segunda vez (para a linha 7Ø), a mensagem "SOMA DOS DOIS NUMEROS E" é exibida, seguida pelo valor de sua soma. T contém o valor do primeiro número digitado e N contém o valor do segundo.

O próximo comando no programa é o comando STOP. Ele ocasiona a interrupção do programa na linha 8Ø. Se o comando STOP não fosse incluido neste ponto, a execução do programa "cairia" na sub-rotina na linha 1ØØ, o que não é desejado. Cada GOSUB em um programa deve ter um RETURN correspondente, executado posteriormente. Um RETURN deve ser encontrado somente se ele é parte de uma sub-rotina que tenha sido chamada por um GOSUB.

Ou um STOP ou um END pode ser usado para separar um programa de suas sub-rotinas. STOP exibirá uma mensagem dizendo em que linha e a interrupção foi efetuada. END terminará o programa sem mensagem alguma.

Ambos os comandos retomam o controle ao usuário, apresentando o caráter de prontidão (]).

DATA ... READ ... RESTORE

Suponha um programa que use constantes numéricas, mas que sejam fáceis de mudar se necessário. O CCE BASIC contém comandos especiais para esta finalidade, chamados READ e DATA. Considere o seguinte programa:

```
10  PRINT "ADIVINHE UM NUMERO";

20  INPUT G

30  READ D

40  IF D = -999999 THEN GOTO 90

50  IF D <> G THEN GOTO 30

60  PRINT "VOCE ACERTOU"

70  END

90  PRINT "MAU CHUTE, TENTE DE NOVO"

95  RESTORE

100 GOTO 10

110 DATA 7, 230, -39, 58, 194,-18, 0, 6.16, 28

120 DATA 87, 6, 10, 35, -34, -999999
```

A finalidade deste programa é jogar um joguinho no qual você tenta adivinhar um dos números contidos nos comandos DATA.

Ao se executar o programa, ocorre o seguinte: quando o comando READ é encontrado, o efeito é o mesmo do comando INPUT, porém ao invés de obter um número do teclado, o número é lido do comando DATA. A primeira vez que um número é solicitado por um

READ o primeiro número no primeiro comando DATA é "lido", a segunda vez que um número é solicitado, o segundo número no primeiro comando DATA é "lido". Quando todo o conteúdo do primeiro comando DATA tiver sido lido deste modo, o segundo comando DATA então será usado. DATA é lido sempre sequencialmente nesta maneira, e pode haver qualquer número de comandos DATA em seu programa.

Para cada tentativa digitada, o computador lê e compara todos os números nos comandos DATA procurando achar um que se iguale com a tentativa. Se o READ devolve -999999, todos os números disponíveis no DATA foram usados, e uma nova tentativa deve ser feita.

Antes de voltar para a linha 1Ø para outra tentativa, é preciso fazer com que READ recomece a leitura dos dados esta é a função do RESTORE. Depois que o RESTORE é encontrado, a próxima unidade de dado "lida" será novamente o primeiro ítem no primeiro comando DATA.

Os comandos DATA podem ser colocados em qualquer lugar dentro do programa. Somente o comando READ faz uso dos comandos DATA em um programa.

VARIÁVEIS INTEIRAS

Existem três tipos de variáveis no CCE BASIC: reais, inteiras e cadeias. Até agora utilizamos somente as reais. Nesta seção são apresentadas as variáveis inteiras, e na próxima seção as cadeias.

O último caráter de uma variável inteira é o símbolo de porcentagem (%). Alguns exemplos desse tipo de variável são: B%, C1%, AB% e X%. Elas podem ser associadas a qualquer valor entre -32767 e 32767.

As variáveis inteiras não podem ser usadas nos comandos FOR ou DEF. Quando usadas em operações aritméticas com as funções SIN, COS, ATN, TAN, SQR, LOG, EXP e RND, são convertidas para precisão real.

Quando um número real é associado a uma variável inteira, ele é arredondado para o menor inteiro nele contido, de forma semelhante ao uso do comando INT. Observe os exemplos a seguir:

```
I% = 1.999

PRINT I%

1

A% = -.00⁹

PRINT A%

-1
```

A maior vantagem da variável inteira se dá quando é necessário economizar espaço na memória. Nesse caso, sempre que possível, deve-se utilizar tal tipo de variável.

VARIAVEIS TIPO CADEIA

Uma sequência de caracteres alfanuméricos é referida como uma "literal". Uma cadeia é uma literal entre aspas. Essas são algumas cadeias:

```
"EDGARD"

"CERES"

"ISTO E UM TESTE"
```

A exemplo das variáveis numéricas (reais e inteiras), as variáveis tipo cadeia podem ser atribuídos valores específicos. Elas são distinguidas das variáveis numéricas pelo caráter dólar ($), após o nome da variável. Por exemplo, tente o seguinte:

```
A$ = "MARCIO KLEIN"

PRINT A$

MARCIO KLEIN
```

Neste exemplo, nós atribuímos à variável, A$ o valor "MARCIO KLEIN".

Observação: não se pode fazer qualquer tipo de associação imediata entre variáveis numéricas e cadeias. Mais adiante, serão vistos comandos especiais, que permitem tal associação.

Agora que atribuímos um valor à variável A$, podemos descobrir qual é a extensão de seu valor (o número de caracteres que ela contém). Fazemos isto do seguinte modo:

```
PRINT LEN (A$), LEN ("PERLA")

12            5
```

A função LEN devolve um inteiro igual ao número de caracteres da cadeia.

O número de caracteres em uma cadeia pode variar de Ø a 255. Uma cadeia que contém Ø caracteres é chamada de cadeia "nula".

Antes que uma variável tipo cadeia tenha um valor atribuído no programa, ela é inicializada como uma cadeia nula. O uso do comando PRINT com uma cadeia nula, não terá qualquer efeito. Por exemplo, digite:

```
PRINT LEN (Q$); Q$; 3

Ø3
```

Outra maneira de criar uma cadeia nula é usar

```
Q$ = ""
```

ou o comando equivalente

```
LET Q$ = ""
```

A criação de uma cadeia nula pode ser usada para liberar o espaço ocupado por uma cadeia não-nula. Porém, você pode ter problemas atribuindo uma cadeia nula a uma variável tipo cadeia, conforme discutido no capítulo 7 na seção IF.

Existem certas funções para a manipulação de cadeias. Por exemplo, agora que atribuímos o conteúdo "MARCIO KLEIN" à cadeia A$, poderíamos querer exibir somente os seis primeiros caracteres de A$. Para tanto, digite:

```
PRINT LEFT$ (A$,6)

MARCIO
```

LEFT$ (A$,N) é uma função para tratamento de cadeias que devolve uma subcadeia composta pelos N caracteres mais à esquerda da cadeia argumento, A$ neste caso. Veja outro exemplo:

```
A$ = "CCE INFORMATICA" :FOR N=1 TO LEN (A$) STEP2: PRINT LEFT$
(A$,N): NEXT N
```

```
C

CCE

CCE I

CCE INF

CCE INFOR

CCE INFORMA

CCE INFORMATI

CCE INFORMATICA
```

Uma vez que A$ tem 15 caracteres, este laço será executado com N = 1,3,5...,13,15. Na primeira vez será exibido somente o primeiro caráter; na segunda vez serão exibidos os três primeiros caracteres etc.

Outra função para tratamento de cadeias é a chamada RIGHT$, RIGHT$ (A$,N) devolve os N caracteres mais à direita da cadeia A$. Tente substituir a função LEFT$ pela RIGHT$ no exemplo anterior, e veja o que acontece.

Existe ainda uma função de tratamento de cadeias que nos permite tomar caracteres do meio de uma cadeia. Execute o seguinte:

```
FOR N = 1 TO LEN (A$): PRINT MID$ (A$,N): NEXT N
```

MID$ (A$,N) devolve uma subcadeia começando na N ésima posição de A$ para o fim (último caráter) de A$. A primeira posição da cadeia é a posição 1, e a última posição possível da cadeia é a posição 255.

Pode ser necessário extrair somente o N- ésimo caráter de uma cadeia. Isto pode ser feito chamando-se MID$ com três argumentos: MID$ (A$,N,1). O terceiro argumento especifica o número de caracteres a ser devolvido, começando pelo caráter N. Por exemplo:

```
A$ = "BOM DIA"

FOR N=1 TO LEN(A$): PRINT MID$(A$,N,1) MID$(A$,N,2):

NEXT N
```

```
B                    BO

O                    OM

M                    M

                      D

D                    DI

I                    IA
                     A
A
```

Veja o capítulo 5 para mais detalhes sobre o funcionamento de LEFT$, RIGHT$ e MID$.

As cadeias também podem ser concatenadas (juntadas) através do uso do sinal (+). Tente o seguinte:

```
B$ = A$ + " " + "JOSE ALBERTO"

PRINT B$
```

```
BOM DIA JOSE ALBERTO
```

Note que há um espaço digitado entre as aspas que seguem A$+. A concatenação é especialmente útil se você deseja tomar uma cadeia em separado, e depois colocá-la de volta com ligeiras modificações. Por exemplo:

```
C$ = RIGHT$ (B$,12) + "-" + LEFT$ (B$,3) + "-"
     + MID$ (B$,5,3)

PRINT C$
```

```
JOSE ALBERTO-BOM-DIA
```

Para converter um número em sua representação de cadeia, e vice-versa, utiliza-se as funções VAL e STR$. Execute o seguinte:

```
SEQ$ = "567.8"

PRINT VAL (SEQ$)
```

```
567.8

SEQ$ = STR$ (3.1415)

PRINT.SEQ$, LEFT$ (SEQ$,5)

3.1415          3.141
```

A função STR$ pode ser usada para mudar o formato de um número para entrada ou saída. Você pode converter um número em uma cadeia e então usar LEFT$, RIGHT$, MID$ e concatenação para reformatar o número conforme desejar.

O seguinte programa curto demonstra como as funções para tratamento de cadeias podem ser usadas para formatar saída numérica:

```
100  INPUT "DIGITE UM NUMERO QUALQUER"; X

110  PRINT: REM PULA UMA LINHA

120  PRINT "APOS CONVERSAO PARA PRECISAO REAL."

130  INPUT"QUANTOS DIGITOS A DIREITA DA VIRGULA?";D

140  GOSUB 1000

150  PRINT "***": REM SEPARADOR

160  GOTO 100

1000 X$=STR$(X):REM CONVERTE ENTRADA PARA CADEIA

1010 REM ACHA A POSICAO DE E, SE EXISTENTE

1020 FOR I = 1 TO LEN (X$)

1030 IF MID$ (X$,I,1) <> "E" THEN NEXT I

1040 REM I ESTA AGORA NA PORCAO DO EXPOENTE (OU FIM)

1050 REM ACHA POSICAO DO PONTO DECIMAL, SE EXISTENTE

1060 FOR J = 1 TO I - 1

1070 IF MID$ (X$,J,1) <> "." THEN NEXT J

1080 REM J ESTA AGORA NO PONTO (OU FIM DA PORCAO
     DO NUMERO)
```

(segue na próxima página)

```
1090 REM EXISTEM D DIGITOS A DIREITA DO PONTO?

1100 IF J + D <= I-1 THEN N = J+D: GOTO 1130: REM SIM

1110 N = I+1: REM NAO, PORTANTO APRESENTE TODOS OS
     DIGITOS

1120 REM APRESENTA A PORCAO DO NUMERO E PORCAO DO
     EXPOENTE

1130 PRINT LEFT$ (X$,N) + MID$ (X$,I)

1140 RETURN
```

O programa acima usa uma sub-rotina começando na linha 1000 para exibir uma variável real predefinida X truncada, não arredondada, para D dígito após o ponto decimal. As variáveis X$, I e J são usadas na sub-rotina como variáveis locais.

A linha 1000 converte a variável real X para a variável tipo cadeia X$.

As linhas 1020 e 1030 pesquisam a cadeia para ver se existe um E. I é posicionado na posição do E, ou no comprimento de X$+1, se não existe E. As linhas 1060 e 1070 pesquisam a cadeia para achar o ponto decimal. J é posicionado na posição do ponto, ou em I-1, se não existe o ponto.

A linha 1100 testa se existe pelo menos D dígitos à direita do ponto decimal. Se eles existirem, a extensão da cadeia deve ser truncada no comprimento J + D, o qual é D posições à direita de J, a posição do ponto decimal.

O valor J + D é atribuido à variável N. Se existir menos que D dígitos à direita da vírgula decimal, a porção inteira do número pode ser usada. A linha 1110 associa à variável N este valor (I-1).

Finalmente, a linha 1130 exibe a variável X como a concatenação de duas sub-cadeias: LEFT$ (X$,N) devolve os dígitos significativos da posição do número, e MID$ (X$,I) devolve a posição do expoente, se existente.

STR$ também pode ser usada para se descobrir quantas posições um número usará. Por exemplo:

```
PRINT LEN (STR$ (12345.678))

 9
```

Se você tem uma aplicação onde um usuário está digitando uma pergunta tal como

QUAL E O VOLUME DE UM CILINDRO DE RAIO 5,36 CM E ALTURA DE 5,1 CM?

Você pode usar a função VAL para extrair os valores numéricos 5,36 e 5,1 da pergunta.

Informações adicionais sobre estas funções , CHR$ e ASCII estão no capítulo 5.

O seguinte programa ordena lista de dados tipo cadeia e exibe a lista em ordem alfabética. Este programa é muito semelhante àquele programa dado anteriormente para ordenar uma lista de números.

```
100 DIM A$ (15)

110 FOR I = 1 TO 15 : READ A$(I) : NEXT I

120 F = 0 : I = 1

130 IF A$(I) <= A$(I+1) THEN GOTO 180

140 T$ = A$(I + 1)

150 A$(I+1) = A$(I)

160 A$(I) = T$

170 F = 1

180 I= I+1 : IF I < 15 THEN GOTO 130

190 IF F=1 THEN GOTO 120

200 FOR I = 1 TO 15 : PRINT A$(I) : NEXT I

220 DATA EDGARD, LILIAN, MARCIO, CASSIANO, ADAMIL,
     RICARDO, BERNARDO, EDUARDO

230 DATA EDSON, PAULA, DAGOBERTO, ADELIA, EXATO, CCE,
     GERALDO
```

MAIS GRAFICOS A CORES

Conforme já apresentado, sabemos que o EXATO apresenta no modo gráfico de Baixa Resolução 4 linhas de texto na parte inferior da tela. O eixo horizontal ou X é padrão, com Ø na posição mais à esquerda, e 39 na mais à direita. O eixo vertical ou Y é não-padrão pois está invertido. Ø é a posição mais acima e 39 é a mais abaixo. Dessa forma, o EXATO exibe até 16ØØ pequenos quadrados em diferentes coordenadas, em qualquer das 16 cores possíveis. Verifique o programa a seguir.

```
1Ø GR : REM INICIALIZA GRAFICO A CORES
        POSICIONA 4Ø x 4Ø EM PRETO
        POSICIONA JANELA DE TEXTO NAS 4 LINHAS
        INFERIORES

2Ø  HOME : REM LIMPA TODA AREA DE TEXTO

3Ø  COLOR = 1: PLOT Ø,Ø : REM QUADRADO MAGENTA EM Ø,Ø

4Ø  LIST 3Ø : GOSUB 1ØØØ

5Ø  COLOR = 2: PLOT 39,Ø : REM QUADRADO AZUL EM X=39,
                Y=Ø

6Ø  HOME: LIST 5Ø : GOSUB 1ØØØ

7Ø  COLOR = 12 : PLOT Ø,39 : REM QUADRADO VERDE EM
                  X=Ø, Y=39

8Ø  HOME : LIST 7Ø : GOSUB 1ØØØ

9Ø  COLOR = 9 : PLOT 39,39 : REM QUADRADO LARANJA EM
                 X=39, Y=39

1ØØ HOME : LIST 9Ø : GOSUB 1ØØØ

11Ø COLOR = 13: PLOT 19,19 : REM QUADRADO AMARELO NO
                             CENTRO DA TELA

12Ø HOME : LIST 11Ø : GOSUB 1ØØØ

13Ø HOME : PRINT "TRACE SEUS PROPRIOS PONTOS"

14Ø PRINT "LEMBRE-SE, X e Y DEVEM SER >=Ø e <= 39
```

```
15Ø INPUT "ENTRE X, Y"; X,Y

16Ø COLOR = 8 : PLOT X, Y: REM QUADRADOS MARRONS

17Ø PRINT "DIGITE  CTRL-C  E APERTE CR PARA
    PARAR"

18Ø GOTO 15Ø

1ØØØ PRINT "*** BATA QUALQUER TECLA PARA CONTINUAR
     ***";: GET A$ : RETURN
```

Após a digitação do programa, liste-o, e confira eventuais erros. Em seguida execute o programa.

O comando GR diz ao computador para mudar para o modo gráfico de Baixa Resolução.

O comando COLOR define a próxima cor a ser traçada. A definição permanece até que seja alterada por um novo comando COLOR. Por exemplo, a cor definida na linha 16Ø permanece a mesma, não importa quantos pontos sejam demarcados.

O valor da expressão seguinte ao comando COLOR deve estar no intervalo de Ø a 255, ou um erro pode ocorrer. Contudo, existem somente 16 cores diferentes, usualmente numeradas de Ø a 15.

Altere o programa, redigitando as linhas 15Ø e 16Ø como segue:

```
15Ø INPUT "ENTRE X,Y COR: "; X,Y,Z

16Ø COLOR = Z : PLOT X, Y
```

Agora execute o programa, e você poderá escolher suas próprias cores, bem como demarcar a posição dos pontos.

O comando PLOT X, Y traça um pequeno quadrado na cor definida pelo último comando COLOR na posição especificada pelas expressões X e Y. Lembre-se, X e Y devem estar no intervalo de Ø a 39.

A instrução GET na linha 1ØØØ é semelhante à instrução INPUT. Ela espera por um único caráter a ser digitado no teclado, e atribui àquele caráter, a variável seguinte ao GET. Não é necessário apertar a tecla CR . Na linha 1ØØØ, GET A$ é usada para parar o programa até que qualquer tecla seja pressionada.

Lembre-se: Para sair do modo gráfico para o modo texto, digite

```
TEXT
```

e aperte a tecla [CR].

Digite o seguinte programa e execute-o para exibir a variedade de cores do EXATO. (lembre-se de digitar NEW para "apagar" o programa anterior).

```
10 GR : HOME

20 FOR I = 0 TO 31

30 COLOR = I/2

40 VLIN 0, 39 AT I

50 NEXT I

60 FOR I = 0 TO 14 STEP 2 : PRINT TAB (I * 2 + 1);
          I; : NEXT I

70 PRINT

80 FOR I = 1 TO 15 STEP 2 : PRINT TAB (I * 2 + 1);
          I; : NEXT I

90 PRINT : PRINT "CORES PADRAO EM BARRAS  EXATO"
```

As barras coloridas são exibidas no dobro de sua largura normal. A barra mais à esquerda é preta, conforme definida por COLOR = 0; a barra mais à direita, branca, é definida por COLOR = 15.

Estando o controle da tonalidade da unidade de vídeo (T.V.) devidamente ajustada, a segunda barra definida por COLOR = 1 será magenta e a terceira (COLOR = 2) será azul escuro.

Um comando da forma VLIN Y1, Y2 AT X foi usado na linha 40. Este comando, traça uma linha vertical a partir de coordenada Y, especificada pela expressão Y1 para a coordenada Y, especificada pela expressão Y2, na posição horizontal especificada pela expressão X. Y1, Y2 e X devem conter os valores no intervalo 0 a 39. Y2 pode ser maior, igual ou menor que Y1. O comando HLIN X1, X2 AT Y é semelhante ao VLIN, exceto que ele traça uma linha horizontal.

Observação: O EXATO desenha uma linha inteira tão facilmente como desenha um simples ponto.

GRÁFICOS DE ALTA RESOLUÇÃO

Agora que você está familiarizado com os gráficos de Baixa Resolução do EXATO, você achará fácil entender os gráficos de Alta Resolução. Os comandos tem aparência semelhante: usualmente são formados pela simples adição de um H (que indica Alta Resolução) àqueles que você já conhece. Por exemplo, o comando HGR passa o computador para a modo gráfico de Alta Resolução, limpa a tela com a cor preta e deixa 4 linhas de texto na parte inferior da tela.

Neste modo, você traça pontos sobre um reticulado, que possui 28Ø posições horizontais por 16Ø posições verticais. Isto, permite desenhar na tela muito mais detalhadamente do que na grade 4Ø x 4Ø dos gráficos de Baixa Resolução. Digitando TEXT você volta o modo texto.

Em adição à tela HGR, existe também uma segunda tela de Alta Resolução que você pode usar em seu computador, através do HGR2.

Esta instrução limpa a tela por completo com a cor preta, dando a você uma superfície para desenho de 28Ø posições horizontais, por 192 posições verticais, e sem texto na parte inferior. De novo, digite TEXT para voltar ao modo texto.

No modo gráfico de Alta Resolução, você dispõe de menos cores. As cores são definidas por um comando de forma:

```
HCOLOR = N
```

onde N é um número de Ø (preto) a 7 (branco). Veja no capítulo 8 a lista das cores disponíveis. Devido a construção das TV coloridas, estas cores podem variar de TV para TV e de uma linha traçada para a próxima.

Finalmente, há uma instrução fácil para todos os traçados em gráficos de Alta Resolução. Para vê-la em ação, digite:

```
HCOLOR = 3
HGR
HPLOT 13Ø, 1ØØ
```

O último comando, traça um ponto de Alta Resolução na cor que você definiu através de HCOLOR (branca) no ponto X=130, Y=100. Como nos gráficos de Baixa Resolução, X=0 está na borda esquerda da tela, aumentando para a direita; Y=0 está no topo da tela, aumentando para baixo. O valor máximo para X é 279; o máximo para Y é 191 com HGR2 ou 159 com HGR. Agora digite:

```
HPLOT 20,15 TO 145,80
```

Uma linha branca, é então desenhada a partir do ponto X=20, Y=15 para o ponto X=145, Y=80. HPLOT pode desenhar linhas entre 2 pontos quaisquer da tela horizontal, vertical, ou qualquer ângulo. Você pode conectar outra linha ao fim da linha anterior, digitando:

```
HPLOT TO 12,80
```

Esta forma do comando toma seu ponto de partida e cor iguais às do último ponto anteriormente utilizado, mesmo que você tenha emitido um novo comando HCOLOR, desde que aquele ponto foi traçado. Você pode ainda "encadear" estes comandos em uma instrução. Tente isto:

```
HPLOT 0,0 TO 279,0 TO 279,159 TO 0,159 TO 0,0
```

Você deve ter agora uma moldura branca contornando completamente os 4 lados da tela.

Aqui está um programa que desenha padrões ondeados na sua tela:

```
80  HOME : REM LIMPA A AREA DE TEXTO
100 VTAB 24 : REM MOVE CURSOR PARA LINHA DE BAIXO
120 HGR : REM PASSA PARA O MODO GRAFICOS DE ALTA
          RESOLUCAO
140 A = RND (1) * 279 : REM PEGA UM X PARA CENTRO
160 B = RND (1) * 159 : REM PEGA UM Y PARA CENTRO
180 I% = (RND (1) * 4) + 2 : REM PEGA UM TAMANHO DE
         PASSO
200 HTAB 15 : PRINT "AVANCANDO A"; I%;
220 FOR X=0 TO 278 STEP I%: REM AVANCO ATE FIM DE X
240 FOR S = 0 TO 1 : REM 2 LINHAS, A PARTIR DE X E
            X + 1
260 HCOLOR = 3 * S : REM PRIMEIRA LINHA PRETA,
    PROXIMA BRANCA
280 REM DESENHA LINHA DIRETA DO CENTRO AO LADO OPOSTO
300 HPLOT X+S, 0 TO A,B TO 279 - X - S, 159
320 NEXT S,X
```

```
340 FOR Y=0 TO 158 STEP I% REM PASSO NOS VALORES DE Y
360 FOR S= 0 TO 1: REM 2 LINHAS, DE Y E Y+1
380 HCOLOR = 3*S: REM PRIMEIRA LINHA PRETA, PROXIMA
    BRANCA
400 REM TRACA LINHA DO "CENTRO" AO LADO OPOSTO
420 HPLOT 279,Y+S TO A,B TO 0,159-Y-S
440 NEXT S,Y
460 FOR PAUSE=1 TO 1500: NEXT PAUSE: REM ATRASO
480 GOTO 120: REM DESENHA UM NOVO PADRAO
```

Esse é um programa um pouco longo. Digite-o com cuidado e utilize o comando LIST a cada porção de linhas digitadas para ver se não houve erros. Quando tudo estiver devidamente conferido, rode (RUN) o programa.

VTAB e HTAB são comandos de movimentação do cursor, usados para exibir caracteres em posições predeterminadas da tela. VTAB1 posiciona o cursor na primeira linha da tela e VTAB 24 na última. HTAB 1 põe o cursor na margem esquerda da tela, e HTAB 40 na margem direita.

Na linha 200, o ponto e vírgula após a instrução PRINT, elimina o avanço de linha ("line feed") automático, executado normalmente após esta instrução.

A função RND (N), onde N é um número positivo, devolve um número aleatório no intervalo entre 0 e.99999999 (veja o capítulo 10 para uma discussão completa da função RND). Assim, a linha 180 associa a variável inteira I% a um número aleatório de 2 a 5 (um número é sempre arredondado para baixo, quando convertido em um inteiro). O tamanho do STEP (passo) em um laço FOR...NEXT não tem que ser inteiro, porém poderia ser mais fácil predizer os resultados para um STEP (passo) inteiro.

Conforme você viu nas linhas 320 e 440, uma instrução pode fornecer o próximo valor (NEXT) para mais de um comando FOR. Seja cuidadoso ao utilizar tal recurso, evitando laços cruzados (as variáveis após NEXT em ordem incorreta).

A linha 460 é apenas um "laço de espera", que dá a você um momento para admirar um padrão antes que comece o próximo. Toda vez que a linha 480 desviar o programa de volta à linha 120, o comando HGR limpa a tela para o próximo padrão.

CAPÍTULO 2

DEFINIÇÕES

DEFINIÇÕES SINTATICAS

(Para uma lista alfabética destas definições, veja apêndice N/

As definições a seguir, usam meta-símbolos tais como, os caracteres { e /, caracteres usados para indicar de maneira inequívoca as estruturas ou relações no CCE BASIC. Os meta-símbolos não são parte do CCE BASIC.

Em adição aos meta-símbolos usuais, utilizamos os seguintes símbolos especiais:

| := meta-símbolo usado para separar alternativas (observação: um ítem também pode ser definido separadamente para cada alternativa)
[] := meta-símbolo usado para ítem opcional
{} := meta-símbolo usado para ítem que pode ser repetido
/ := meta-símbolo usado para incluir ítem cujo valor deve ser usado : o valor de x é escrito /x/
- := meta-símbolo que indica um espaço requerido

META-SÍMBOLO
:= |[]{}/

META-SÍMBOLO
:= letra minúscula

dígito
:= 1|2|3|4|5|6|7|8|9|0

METANOME
:= {meta-símbolo} {dígito}

SÍMBOLOS ESPECIAIS
:= !|#|$|%|&|'|(|)|:|;|=|-|@|+|;|?|/|>|.|<|,|[|^|"

Caracteres de controle (caracteres que são digitados com a tecla CTRL comprimida) e o caráter nulo também são especiais. O CCE BASIC usa o colchete direito (]) somente como caráter de prontidão.

LETRA
:=A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R|S|T|U|V|W|X|Y|Z

CARÁTER
:= letras|dígito|símbolos especiais

CARÁTER ALFANUMÉRICO
:= letra dígito

NOME
:= letra [{letra|dígito}]

Um nome pode ter até 238 caracteres de extensão.

Para distinguir nomes, o CCE BASIC utiliza somente os dois primeiros caracteres, ignorando os demais. Assim, os nomes CAVALO e CARTA não são diferenciados pelo CCE BASIC. Note entretanto que, mesmo as partes ignoradas dos nomes não devem ter qualquer símbolo especial, aspas ou palavras reservadas (veja no apêndice G a lista das palavras reservadas e comentários sobre as exceções dessa regra).

INTEIRO
:= [+|-] {dígito}
Inteiros devem estar dentro do intervalo de -32767 a 32767.

Quando convertendo não inteiros para inteiros, o CCE BASIC usualmente trunca o não-inteiro para o menor inteiro nele contido. Contudo, isto não é verdadeiro no limite à medida que o não-inteiro se aproxima do próximo maior inteiro. Por exemplo:

```
A% = 123,99999959999          B% = 123,9999999
PRINT A%                         PRINT B%
123                              124

C% = 12345 999 995 999        D% = 12345,99999999
PRINT C%                      PRINT D%
12345                            12346
```

Um inteiro ocupa 2 bytes (16 bits) na memória.

NOME DE VARIÁVEL INTEIRA
:= nome %

Um número real pode ser armazenado como uma variável inteira, porém o CCE BASIC converte antes o número real para um inteiro.

REAL
:=[+|-] [dígito [.[dígito [E[+|-] dígito [dígito]]
:=[+|-] [[dígito.[[dígito [E[+|-]dígito [dígito]]

A letra E, quando usada na notação de número real (uma forma de "notação científica"), vale por "expoente". É a forma rápida de escrever *1Ø . Dez é elevado à potência do número à direita do E, e o número à esquerda do E é multiplicado pelo resultado.

No CCE BASIC, os números reais devem estar no intervalo -1E38 a 1E38, ou você se arrisca a uma mensagem de erro do tipo

?S/ESPACO ERRO

Usando adição ou subtração, pode-se gerar números tão grandes como 1.7E38 sem receber esta mensagem.

Um número real, cujo valor ansoluto é menor do que 2.9388E-39, é convertido para zero. O CCE BASIC reconhece os seguintes como números reais, quando apresentados por si mesmos, e os avalia em zero:

```
.    +.    -.    .E    +.E   -.E
.E+  .E-   +.E-  +.E+  -.E+  -.E-
```

Consequentemente, o elemento de matriz M(.) é o mesmo que M(Ø).

Em adição aos reais abreviados listados acima, os seguintes são reconhecidos como reais e avaliados como zero, quando usados como respostas numéricas para INPUT, ou como elementos numéricos de DATA:

```
+    -     E     +E    -E    espaço
E+   E-    +E+   +E-   -E+   -E-
```

A instrução GET avalia todos os reais de um caráter nas listas acima como zero.

Quando exigimos um número real, o CCE BASIC mostrará até o máximo de nove dígitos, excluindo o expoente (se houver). Quaisquer dígitos posteriores serão truncados.

À esquerda do ponto decimal, quaisquer zeros precedendo o dígito não-zero mais à esquerda, não são exibidos. À direita do ponto decimal, quaisquer zeros seguintes ao dígito não-zero mais à direita, não serão exibidos. Se não existirem dígitos não-zero à direita do ponto decimal, o ponto decimal não é exibido. Seguem alguns exemplos irregulares de arredondamento:

```
PRINT 99 999 999.9
99 999 999.9

PRINT 99 999 999.99
100 000 000

PRINT 11.111 111 450 00
11.111 111 5

PRINT 11.111 111 451 9
11.111 111 4
```

Se o valor absoluto de um número real é maior ou igual a .01 e menor do que 999 999 999.2, o número real é impresso em notação de ponto fixo. Que significa que nenhum expoente é exibido.

No intervalo .100 000 000 5 a .0 999 999 999 os reais são exibidos com até DEZ dígitos, incluindo o zero imediatamente à direita do ponto decimal. Esta é a única exceção ao limite de nove dígitos, excluído o expoente.

Se você tentar usar um número com mais de 38 dígitos, tal como:

```
211.111 111 111 111 111 111 111 111 111 111 111 111 111
```

então a mensagem

```
?S/ESPACO ERRO
```

é apresentada, mesmo que o número esteja claramente dentro do intervalo -1E38 a 1E38. Isto é verdadeiro mesmo que a maioria dos dígitos sejam zeros não significativos, como no número

```
211.000 000 000 000 000 000 000 000 000 000 000 000 000
```

Zeros à esquerda do ponto decimal, contudo, são ignorados. Se o primeiro dígito é um "1" e o segundo dígito é menor ou igual a 6, números com 39 dígitos podem ser usados sem receber mensagem de erro.

Um número real ocupa 5 bytes (40 bits) na memória.

NOME DE VARIAVEL REAL
:= nome

VARIÁVEL ARITMÉTICA

:= avar

AVAR

:= nome|nome%

Todas as variáveis simples ocupam 7 bytes na memória, 2 bytes para o nome, e 5 bytes para o valor real ou inteiro.

DELIMITADOR

:= ||(|)|=|-|+|^|>|<|/|*|,|;|:

Um nome não tem que ser separado de uma precedente ou seguinte palavra reservada por qualquer destes delimitadores.

OPERADOR ARITMÉTICO

:= aop

AOP

:= +|-|*|/|^

OPERADOR LÓGICO ARITMÉTICO

:= alop

ALOP

:= AND | OR|=|>|<|><|<>|>=|=>|<=|=<

NOT não está incluído aqui propositadamente.

OPERADOR

:= op

OP

:= aop | alop

EXPRESSÃO ARITMÉTICA

:= aexpr

AEXPR

:= avar|real|inteiro
:= (aexpr)

Se os parênteses estiverem com mais de 36 níveis, será exibida a mensagem:

?FALTA MEMORIA ERRO

:= [+|-|NOT] aexpr

NOT aparece somente aqui, acompanhado por "+" e "-".

:= aexpr op aexpr

ÍNDICE

:= (aexpr [{, aexpr}])

O número máximo de dimensões é 89, embora na prática ele será limitado pela extensão de memória disponível. "aexpr" deve ser positiva, e em uso ela é convertida em um inteiro.

AVAR

:= avar índice

AEXPR

:= avar índice

LITERAL

:= [{caráter}]

CADEIA

:= "[caracter]"
Uma cadeia ocupa 1 byte (8 bits) para seu comprimento, 2 bytes para seu endereço, e 1 byte para cada caráter da cadeia.

:= "[{caracter}] [CR]

Essa forma de cadeia somente pode ser usada no fim de uma linha.

CADEIA NULA

:= ""

NOME DE VARIÁVEL TIPO CADEIA

:= nome $

VARIÁVEL CADEIA

:= svar

SVAR

:= nome$ | nome$ índice

O endereço e o nome de variável, ocupam 2 bytes cada, na memória. O comprimento e cada caráter ocupam 1 byte cada.

OPERADOR DE CADEIA
:= sop

sop
:= +

EXPRESSÃO DE CADEIA
:= sexpr

SEXPR
:= svar | cadeia
:= sexpr sop sexpr

OPERADOR LÓGICO DE CADEIA
:= slop

SLOP
:= | = | > | >= | => | < | <= | =< | <> | ><

AEXPR
:= sexpr slop sexpr

VARIÁVEL
:= var

VAR
:= avar | svar

EXPRESSÃO
:= expr

EXPR
:= aexpr | sexpr

CARÁTER DE PRONTIDÃO
:=]

O colchete de prontidão (]) é exibido quando o CCE BASIC está pronto para aceitar outro comando.

REINICIALIZAÇÃO
:= uma depressão de tecla [RESET]

ESC
:= uma depressão da tecla [ESC]

ENTRA
:= uma depressão da tecla [CR]

CONTROL

:= mantenha comprimida a tecla [CTRL], enquanto a tecla definida a seguir é digitada.

NÚMERO DA LINHA

:= numlinha

NUMLINHA

:= {dígito}

Os números de linha, devem estar dentro do intervalo Ø a 63.999, ou resultará a mensagem:

?SINTAX ERRO

LINHA

:= numlinha [{instrução:}] instrução

Uma linha pode ter até 239 caracteres. Isto inclui todos os espaços digitados pelo usuário, porém não inclui os espaços adicionados pelo CCE BASIC na formatação da linha.

AVALIAÇÃO DE EXPRESSÕES

Os operadores estão listados verticalmente em ordem de execução, a partir da mais alta prioridade (parênteses) para a mais baixa prioridade (OR). Os operadores listados na mesma linha são de mesma prioridade. Os operadores de mesma prioridade em uma expressão são executados da esquerda para a direita.

( )
+ - NOT
^
* /
+ -
> < >= <= => =< < > >< =
AND
OR

CONVERSÃO DE TIPOS

Quando um inteiro e um real estão ambos presentes em um cálculo, todos os números são convertidos em reais antes que o cálculo seja efetuado. Os resultados são convertidos para o tipo aritmético (inteiro ou real) da variável final para os quais eles são definidos. Funções que são definidas em dado tipo aritmético, converterão argumentos de outro tipo, ao tipo para os quais eles são definidos. Cadeias e tipos aritméticos não podem estar misturados. Cada um pode ser convertido para o outro, por funções fornecidas para essa finalidade.

MODALIDADES DE EXECUÇÃO

Imediata (imed) - Algumas instruções podem ser usadas na modalidade de execução imediata no CCE BASIC. Neste modo, uma instrução deve ser digitada sem o número de linha. Quando a tecla [CR] é apertada, a instrução é imediatamente executada.

Programada (prog) - Instruções usadas na modalidade de execução programada, devem aparecer em uma linha que comece com um número de linha. Quando a tecla [CR] é apertada, o CCE BASIC armazena a linha numerada para utilização posterior. As instruções no modo de execução programado são executadas somente após o comando RUN.

CAPÍTULO 3

COMANDOS ESPECIAIS

As seções desse capítulo são iniciadas pela indicação da modalidade de execução dos comandos apresentados, seguidas da sintaxe dos mesmos.

LOAD E SAVE

LOAD imed e prog
SAVE imed e prog

LOAD
SAVE

A instrucão LOAD carrega um programa de uma fita cassete, isto é, transfere o programa gravado na fita cassete para a memória do computador.

A instrucão SAVE salva um programa em uma fita cassete, isto é, grava em uma fita cassete, um programa da memória do computador.

Não existe qualquer mensagem ou sinal emitido por estas instruções, a não ser um sinal sonoro (bip) para indicar o início ou fim da gravação. Você deve ter o gravador de fita cassete funcionando adequadamente ("play" para a instrucão LOAD e "record" e "play" para a instrucão SAVE), quando a instrucão é executada.

A execução de um programa continua possível após a operacão de gravação, porém, uma operação de carga apaga o programa corrente quando ele começa a ler as novas informacões da fita cassete.

Somente a tecla RESET pode interromper as operacões de carga e gravação.

Se as palavras reservadas LOAD ou SAVE são usadas como primeiros caracteres de um nome de variável, os comandos podem ser executados antes da emissão da mensagem ?SINTAX ERRO. Por exemplo, SAVEIRO = 5 ocasionará a tentativa de gravação do programa corrente. Você pode esperar pelo segundo "bip" (e a mensagem ?SINTAX ERRO) ou apertar RESET.

A declaração LOADAPECA = 47 suspende o sistema, enquando o CCE BASIC apaga o programa corrente, e aguarda indefinidamente por um programado do gravador de fita cassete. Somente pela depressão da tecla RESET você poderá recuperar o controle do computador.

NEW

NEW imed e prog
NEW

Apaga o programa corrente e todas as variáveis.

RUN

RUN imed e prog
RUN [numlinha]

Ao ser emitido, ele limpa todas as variáveis, ponteiros e pilhas, iniciando a seguir a execução do programa a partir de /numlinha/ ou se este não foi definido, a partir da linha de programa de menor número. Se não existir programa na memória, ele retorna imediatamente o controle do computador ao operador.

No modo programado, se /numlinha/é usado, porém não existe aquela linha no programa, ou se numlinha é negativo, então a mensagem ?LINHA INDEFINIDA ERRO aparece. Se/numlinha/ é maior que 63999 a mensagem ?SINTAX ERRO aparece. Você não é informado em qual linha o erro ocorreu. Na modalidade de execução imediata, por outro lado, estas duas mensagens tornam-se ?LINHA INDEFINIDA ERRO EM XXX e ?SINTAX ERRO EM XXX onde XXX pode ser vários números de linha.

Se RUN é usado em execução imediata, toda e qualquer porção subsequente do programa não é executada.

STOP, END, CTRL-C, RESET E CONT

Os modos de execução desses comandos são:

STOP : imediato e programado
END : imediato e programado
CTRL-C: somente imediato
RESET : somente imediato
CONT : imediato e programado

Nenhum deles possui parâmetros.

STOP causa a interrupção da execução de um programa, e retoma o controle do computador ao usuário, apresentando a mensagem:

PAUSA EM numlinha

onde/numlinha/é o número da linha na qual o programa foi interrompido.

END tem o mesmo efeito de STOP, porém não apresenta qualquer mensagem.

CTRL-C tem um efeito equivalente à inserção de uma instrução STOP, imediatamente após a instrução que está sendo executada correntemente. CTRL-C pode ser usado, por exemplo, para interromper uma listagem. Também pode ser usado para interromper uma instrução INPUT, porém, somente se é o primeiro caráter a entrar. A instrução INPUT não é interrompida até que a tecla [CR] seja pressionada.

[RESET] aborta quase qualquer programa ou comando incondicionalmente e imediatamente. O programa não é perdido, porém alguns enderecadores e pilhas são apagados.

Se a execução de um programa foi interrompida por STOP, END ou CTRL-C, o comando CONT causa o prosseguimento da execução do programa na próxima instrução.

Se não houve interrupção na execução, então o comando CONT não tem efeito. Após [RESET] CTRL-C [CR] o programa pode não continuar a execução corretamente, uma vez que alguns endereçadores e pilhas terão sido apagados.

Quando um comando INPUT é interrompido por CTRL-C, uma tentativa para continuar a execução resulta em uma mensagem

?SINTAX ERRO EM numlinha

onde/numlinha/é o número da linha, que contém o comando INPUT.

Caso, após a interrupção do programa, você:

a) modifique ou apague qualquer linha do programa
b) tente qualquer operação que resulte em mensagem de erro.

O uso do comando CONT provocará a mensagem de erro:

?"CONT" ILEGAL ERRO

Contudo, as variáveis do programa podem ser alteradas enquanto não ocorrer mensagens de erro.

Quando a instrução CONT é usada na modalidade programada, a execução do programa é interrompida naquela instrução, mas o controle do computador NÃO volta ao usuário. O usuário pode recuperar o controle do computador pela emissão de um comando CTRL-C, porém uma tentativa de continuar a execução do programa na próxima instrução apenas abandona novamente o controle do programa interrompido.

Se a instrução DEL é usada no modo programado, as linhas especificadas são apagadas e a execução do programa para. Uma tentativa para usar CONT sob estas circunstâncias causará a mensagem

?"CONT" ILEGAL ERRO

Quando a instrução CONT é usada no modo programado, a execução do programa é interrompida naquela instrução, mas o controle do computador NÃO volta ao usuário. O usuário pode recuperar o controle do computador pela emissão de um comando CTRL-C, porém uma tentativa de continuar a execução do programa na próxima instrução apenas abandona novamente o controle do programa interrompido.

TRACE E NOTRACE

TRACE imed prog
NOTRACE imed prog

TRACE
NOTRACE

TRACE determina uma modalidade de depuração de programa que exibe o número de linha de cada instrução à medida que esta é executada. Quando o programa também apresenta dados na tela, os números de linha podem ser exibidos de uma forma inesperada ou ainda superpostos às linhas apresentadas pelo programa. NOTRACE desativa a modalidade de depuração.

Uma vez determinada, a modalidade TRACE não é desativada por RUN, CLEAR, NEW, DEL ou RESET ; RESET CTRL-B desativa TRACE (e elimina qualquer programa armazenado na memória).

PEEK

PEEK imed e prog
PEEK (aexpr)

PEEK obtém o conteúdo do endereço indicado por aexpr, na base decimal. O apêndice J contém exemplos do uso de PEEK.

POKE

POKE imed e prog
POKE aexpr 1, aexpr 2

POKE armazena um byte, o binário equivalente ao valor decimal /aexpr 2/, na localização de memória cujo endereço é dado pela aexpr 1.

O valor de /aexpr 2/ deve ser de Ø a 255; e o de /aexpr 1/ de -65535 até 65535. Os números reais são convertidos em inteiros antes da execução. Valores fora desses intervalos causam a exibição da mensagem

?VALOR ILEGAL ERRO

/aexpr 2/ será armazenada com sucesso somente se o equipamento apropriado de recepção (memória, ou um dispositivo de saída conveniente) existir no endereço especificado pela /aexpr 1/.

/a expr2/ não será armazenada com sucesso em áreas não receptivas tais como EPROM ou dispositivos de entrada/saída não usados. Em geral, isto significa que /aexpr 1/ estará no intervalo de Ø até máximo, onde máximo é determinado pela quantidade de memória no computador.

Muitas áreas de memória contêm informações necessárias ao funcionamento do computador. Um POKE para tais áreas pode danificar a operação do sistema, portanto, muito cuidado ao utilizar tal instrução.

WAIT

WAIT imed e prog
WAIT aexpr 1, aexpr 2 [,aexpr 3]

Permite ao usuário inserir uma pausa condicional dentro de um programa. Somente RESET pode interromper um WAIT.

/aexpr 1/ é o endereço de uma localização de memória, ele deve estar dentro do intervalo de -65535 a 65535 para evitar a mensagem

?VALOR ILEGAL ERRO

Na prática, /aexpr 1/ é usualmente limitada ao intervalo de endereços correspondentes às localizações para as quais existem dispositivos de memória válidos, a partir de Ø até o valor máximo de HIMEM: no seu computador. Veja HIMEM: e POKE para mais detalhes. Endereços positivos e negativos podem ser usados.

/aexpr 2/ e /aexpr 3/ devem estar no intervalo de Ø a 255. decimal. Quanto a instrução WAIT é executada, estes valores são convertidos em números binários no intervalo de Ø a 11111111.

Se somente /aexpr 1/ e /aexpr 2/ são especificadas, cada um dos oito bits do conteúdo binário da localização /aexpr 1/ é substituído pelo resultado da operação de álgebra booleana AND com o correspondente bit do número equivalente da /aexpr 2/. Para cada bit, isto dá um zero, a menos que ambos os bits correspondentes sejam altos (1). Se o resultado deste processo for oito zeros, então o teste é repetido. Se qualquer resultado é não-zero (o que significa que pelo menos um bit alto (1) em /aexpr 2/ foi casado com um bit alto (1) correspondente na localização /aexpr 1/), a instrução WAIT é completada e o programa retoma a execução ou próxima instrução.

WAIT aexpr 1, 7

ocasiona uma pausa no programa até que pelo menos um dos três bits mais à direita na localização /aexpr 1/ seja alto (1).

WAIT aexpr 1, Ø

ocasiona uma pausa de valor indeterminado.

Se todos os três parâmetros foram especificados, então WAIT é executado conforme segue: primeiro, cada bit no conteúdo binário da localização /aexpr 1/ é submetido à operação da álgebra booleana XOR com o correspondente bit no número binário equivalente da /aexpr 3/. Um bit altó (1) em /aexpr 3/ dá um resultado que é o REVERSO do bit correspondente na localização /aexpr 1/ (1 torna-se Ø; Ø torna-se 1/.

Um bit baixo (Ø) em /aexpr 3/ dá um resultado que é o mesmo do bit correspondente na localização /aexpr 1/.

Se a /aexpr 3/ for simplesmente zero, a porção XOR não faz nada.

Segundo, cada resultado é submetido à operação da álgebra boolena AND com o correspondente bit no número binário equivalente da /aexpr 2/.

Se os resultados finais forem oito zeros, o teste é repetido. Se qualquer resultado é diferente, a instrução WAIT é completada, e o programa prossegue na próxima instrução.

Outra maneira de olhar a instrução WAIT: o objeto é testar o conteúdo da localização /aexpr 1/ para ver quando algum certo bit alto (1, ou ligado) ou algum determinado bit baixo (Ø, ou desligado). Cada um dos oito bits no número binário equivalente de /aexpr 2/ indica se você está interessado no bit correspondente na localização /aexpr 1/: 1 significa que você está interessado, Ø significa ignorar aquele bit. Cada um dos oito bits no número equivalente da /aexpr 3/ indica qual estado você está esperando pelo bit correspondente na localização /aexpr 1/: 1 significa que o bit deve ser baixo, Ø significa que o bit deve ser alto.

Se qualquer dos bits nos quais você indicou interesse (por um 1 no bit correspondente de /aexpr 2/) casa com o estado que você especificou para aquele bit (pelo bit correspondente de /aexpr 3/ a instrução WAIT está completada. Se /aexpr 3/é omitida, seu valor de ausência é zero. Por exemplo:

WAIT aexpr 1, 255,Ø significa uma pausa até que pelo menos um dos 8 bits na localização /aexpr 1/ seja alto (1).

WAIT aexpr 1, 255 idêntica ao anterior.

WAIT aexpr 1, 255, 255 significa uma pausa até que pelo menos um dos 8 bits na localização /aexpr 1/ seja baixo (Ø).

WAIT aexpr 1, 1, 1 significa uma pausa até que o bit mais à direita na localização /aexpr 1/ seja baixo, se considerar o estado dos demais bits.

WAIT aexpr 1, 3, 2 significa uma pausa até que ou o bit mais à direita na localização /aexpr1/ seja alto, ou o bit próximo do mais à direita seja baixo, ou ambas as condições.

Este programa espera até que você digite qualquer caráter cujo código ASCII (veja apêndice K) seja IGUAL (=):

```
1ØØ POKE -16384,Ø
1Ø5 REM PAUSA ATE QUE O TECLADO SEJA ACIONADO POR
    QUALQUER TECLA
11Ø WAIT -16384,128: REM ESPERA ATE QUE O BIT ALTO=1
115 REM ESPERA POUCO MAIS ATE QUE A TECLA TOCADA
    SEJA IGUAL
12Ø WAIT -16384,1,1: REM ESPERA ATE QUE O BIT BAIXO
                     SEJA ZERO
13Ø PRINT "IGUAL"
14Ø GOTO 1ØØ
```

CALL

```
CALL imed e prog
CALL aexpr
```

Ocasiona a execução de uma sub-rotina em linguagem de máquina na localização de memória cujo endereço DECIMAL é especificado pela /aexpr/.

/aexpr/ deve estar no intervalo -65535 até 65535 ou será exibida a mensagem

```
? VALOR ILEGAL ERRO
```

Na prática, /aexpr/ é usualmente limitada ao intervalo de endereços válidos de memória, a partir de Ø até o valor máximo para HIMEM: em seu computador. Veja HIMEM: e POKE para mais detalhes.

Exemplos negativos e positivos equivalentes podem ser usados alternadamente. Por exemplo, "CALL -936" e "CALL 64600" são idênticas.

O apêndice J contém exemplos do uso de CALL.

HIMEM:

HIMEM: imed e prog
HIMEM: aexpr

Posiciona o endereço de mais alta localização de memória disponível a um programa CCE BASIC, incluindo variáveis. É usado para proteger a área de memória acima desse endereço para dados, gráficos ou rotinas em linguagem de máquina.

/aexpr/ deve estar no intervalo -65535 até 65535, inclusive para evitar a mensagem

?VALOR ILEGAL ERRO

Entretanto, programas podem não ser executados de formas confiáveis, a menos que exista equipamento adequado de memória nas localizações especificadas por todos os endereços até (inclusive) /aexpr/.

Em geral, o valor máximo de /aexpr/ é determinado pelo total de memória no computador. Por exemplo, com 48K de memória /aexpr/ pode chegar a 49152.

Normalmente, o CCE BASIC posiciona HIMEM: automaticamente, no endereço de memória mais alto disponível no computador do usuário, quando é chamado pela primeira vez.

O valor corrente de HIMEM: está armazenado nas localizações de memória 115 e 116 (decimais). Para ver o valor corrente de HIMEM:, digite

PRINT PEEK (116) * 256 + PEEK (115)

Se HIMEM: posiciona um endereço de memória cujo valor é mais baixo do que aquele posicionado pelo "LOMEM:", ou ainda que não deixa suficiente memória disponível para a execução do programa, é dada a mensagem

?FALTA MEMORIA ERRO

/aexpr/ pode estar no intervalo de Ø incrementado para 65535, ou no intervalo equivalente -65535 incrementado para -1. Valores positivos equivalentes a negativos podem ser usados alternadamente.

HIMEM: não é reposicionado por CLEAR, RUN, NEW, DEL, alteração ou inclusão de uma linha de programa, ou RESET. HIMEM é reposicionado por RESET CTRL-B CR, que também apaga qualquer programa armazenado.

LOMEM:

LOMEM: imed e prog
LOMEM: aexpr

Posiciona o endereço da mais baixa localização de memória disponível a um programa CCE BASIC. Este é usualmente o endereço da localização de memória inicial para a primeira variável.

Normalmente, o CCE BASIC posiciona automaticamente LOMEM: no fim do programa corrente, antes de executá-lo.

/aexpr/ deve estar no intervalo -65535 até 65535, inclusive, para evitar a mensagem

?VALOR ILEGAL ERRO

Contudo, se LOMEM: é posicionado mais alto do que o valor corrente do de HIMEM:, a mensagem:

?FALTA MEMORIA ERRO

é exibida. Isto significa que /aexpr/ deve ser mais baixo do que o valor máximo que pode ser posicionado por HIMEM:.

Se LOMEM: é posicionado mais baixo do que o endereço da mais alta localização de memória ocupada pelo sistema operacional corrente (mais qualquer programa corrente armazenado) a mensagem:

?FALTA MEMORIA ERRO

é novamente exibida. Isto impõe um limite inferior absoluto para /aexpr/ de cerca de 2051 para programas CCE BASIC.

LOMEM: é reposicionado por NEW, DEL, pela adição ou alteração de uma linha de programa, ou por [RESET] CTRL-B [CR] . A última operação apaga qualquer programa armazenado. Ele não é reposicionado por RUN, [RESET] CTRL-C [CR] ou [RESET] ØG [CR] .

O valor corrente de LOMEM: está armazenado nas localizações de memória 1Ø6 e 1Ø5 (decimais). Para ver o valor corrente de LOMEM:, digite

```
PRINT PEEK (1Ø6) * 256 + PEEK (1Ø5)
```

Uma vez posicionado, a menos que ele seja primeiro reposicionado por um dos comandos acima, LOMEM: pode ser posicionado para um novo valor somente se o novo valor é MAIS ALTO (em memória) do que o valor anterior. Uma tentativa para posicionar um LOMEM: mais baixo do que o valor ainda em efeito dá a mensagem

```
?FALTA MEMORIA ERRO
```

A alteração do valor de LOMEM: durante o curso de um programa podem causar a não disponibilidade de certas pilhas ou porções do programa, de modo que o programa não continuará a ser executado corretamente.

Endereços positivos e negativos equivalentes podem ser usados alternadamente.

USR

```
USR imed e prog
USR (aexpr)
```

Esta função passa /aexpr/ para uma sub-rotina em linguagem de máquina.

O argumento /aexpr/ é avaliado e colocado em um acumulador de ponto flutuante (localizações $9D até $A3), e um JSR desvio para localização $ØA é executado.

As localizações $ØA até $ØC devem conter um JMP para a localização inicial da sub-rotina em linguagem de máquina. O valor de retorno para a função é colocado no acumulador de ponto flutuante.

Para obter um inteiro de 2 bytes a partir do valor do acumulador de ponto flutuante, sua sub-rotina deveria fazer um JSR para $E1ØC. No retorno, o valor inteiro estará nas localizações $AØ (byte de mais alta ordem) e $A1 (byte de mais baixa ordem).

Para converter um resultado inteiro para seu equivalente em ponto flutuante, de modo que a função possa retornar aquele valor, coloque o inteiro de 2 bytes em registrador A (byte de alta ordem) e Y (byte de baixa ordem).

Em seguida, faça um JSR para $E2F2. No retorno, o valor em ponto flutuante estará no acumulador de ponto flutuante.

Para voltar ao CCE BASIC, faça um RTS. Aqui está um programa trivial usando a função USR, apenas para mostrar a você o seu formato:

```
] CALL 151
* ØA : 4C ØØ Ø3 [CR]
* Ø3ØØ : 6Ø [CR]
* CTRL C [CR]
] PRINT USR (6) * 3
18
```

Inicialmente, "chamamos" o Monitor através de CALL-151. Na localização $3ØØ (byte de baixa ordem em primeiro, em seguida o byte de alta ordem). Na localização $3ØØ, nós colocamos um RTS (código 6Ø).

De volta ao CCE BASIC, quando USR (6) foi encontrado, o argumento 8 foi colocado no acumulador e o Monitor fez um JSR para a localização $ØA onde ele encontrou um JMP para $3ØØ. Em $3ØØ ele achou um RTS que o enviou de volta ao CCE BASIC. O valor retornado foi exatamente o valor original 6 no acumulador, que foi multiplicado por 3 dando 18.

CAPÍTULO 4

COMANDOS DE EDIÇÃO E FORMATOS RELACIONADOS

Veja também, no capítulo 3, CTRL-C

Cada seção deste capítulo, é iniciada indicando o modo de execução em que o comando atua, seguido pela sintaxe do respectivo comando.

LIST

```
LIST imed e prog
LIST [numlinha 1] [-numlinha 2]
LIST [numlinha 1] [,numlinha 2]
```

Se nem /numlinha 1/ nem /numlinha 2/ estiverem presentes, com ou sem um delimitador, o programa inteiro é exibido na tela. Se /numlinha 1/ estiver presente sem um delimitador, ou se /numlinha 1/ = /numlinha 2/, então apenas a linha numerada pelo /numlinha 1/ é exibida.

Quando /numlinha 1/ e um delimitador estiverem presentes, então o programa é listado a partir da linha numerada por /num linha 1/ até o fim. Se um delimitador e /numlinha 2/ estiverem presentes, então o programa é listado a partir do início até a linha numerada pelo /numlinha 2/. Se /numlinha 1/, um delimitador e /numlinha 2/ estiverem todos presentes, então o programa é listado a partir da linha numerada por /numlinha 1/ até a linha numerada por /numlinha 2/, inclusive.

Quando mais de uma linha deve ser listada, se a linha numerada por /numlinha 1/ no comando LIST não aparece no programa, o comando LIST usará o número da linha imediatamente superior que aparece no programa. Se a linha numerada pelo /numlinha 2/ no comando LIST não aparece no programa, o comando LIST usará o número de linha imediatamente inferior que aparece no programa.

Estes todos listam o programa inteiro:

```
LIST Ø  LIST [,|-]Ø   LIST Ø [,|-] Ø
LIST numlinha, Ø
```

lista a partir da linha numerada pelo numlinha até o fim do programa.

```
LIST , Ø
```

lista o programa inteiro e em seguida dá a mensagem

```
?SINTAX ERRO
```

O CCE BASIC sintetiza suas linhas de programa antes de armazená-las, removendo os espaços desnecessários no processo. Quando estiver listando, as linhas são reconstituídas, adicionando espaços de acordo com suas próprias regras. Por exemplo,

```
10 J=+8/-15:B=-2
```

torna-se

```
10 J = 8 / - 15:B = -2
```

quando listada.

LIST usa uma largura de linha variável. Isto pode ser um problema quando você estiver tentando editar ou copiar uma instrução listada. Para forçar LIST a abandonar a formatação com espaços extras, limpe a tela, e reduza a janela de texto para largura 33 através dos comandos:

```
HOME
POKE 33,33
```

O CCE BASIC trunca uma linha em 239 caracteres, em seguida LIST adiciona espaços liberalmente. Assim, você pode fazer entrar muitos caracteres extras ao deixar de emitir os espaços, quando estiver digitando. LIST os adicionará. Uma tentativa para copiar seu comando expandido a partir da tela resulta no truncamento em 239 caracteres novamente, incluindo os espaços adicionados pelo LIST. Uma listagem pode ser interrompida por CTRL-C.

DEL

```
DEL imed e prog
DEL numlinha 1, numlinha 2
```

DEL apaga o intervalo das linhas do programa numerados por /numlinha 1/ até /numlinha 2/, inclusive. Se /numlinha 1/ não existe no programa, o número de linha imediatamente superior é usada e, se /numlinha 2/ não existe no programa, o número de linha imediatamente inferior é usado.

Caso você não siga o formato usual, o resultado da instrução variará conforme indicado a seguir:

SINTAXE	RESULTADO
DEL	?SINTAX ERRO
DEL,	?SINTAX ERRO
DEL, b	?SINTAX ERRO
DEL -a(,b)	?SINTAX ERRO
DEL Ø,b	Apaga a linha zero, sem considerar o valor de b.
DEL 1, -b	Ignorado, mesmo que o menor número de linha de programa seja zero.
DEL a, -b	?SINTAX ERRO Se A é maior do que o menor número de linha de programa, a menos que o menor número de linha de programa é zero e A é um.
DEL a, -b	Ignorado se A não é zero e a única linha de programa é a linha número zero.
DEL a, -b	Ignorado se A não é zero e se A é menor ou igual ao menor número de linha de programa.
DEL a [,]	Ignorado.
DEL a,b	Ignorado se A não é zero e A é maior do que B.

Quando usada no modo programado, DEL funciona conforme descrito acima, e em seguida interrompe a execução. CONT não funciona nesta situação.

REM

REM imed e prog
REM {caráter|"}

Esta instrução serve para permitir a inserção de texto de qualquer tipo em um programa. Todos os caracteres, incluindo separadores de comandos e brancos podem ser incluídos. Seus significados usuais são ignorados.

Quando as instruções REM são listadas, o CCE BASIC insere um espaço extra após REM, não importando quantos espaços tenham sido digitados após REM pelo usuário.

VTAB

VTAB imed e prog
VTAB aexpr

Move o cursor para a linha que esta /aexpr/ linhas abaixo do topo da tela. A linha do topo é a linha 1; a linha do fundo é a linha 24. Este comando pode envolver a movimentação do cursor para cima ou para baixo, mas nunca para a direita ou esquerda.

Argumentos fora do intervalo de 1 a 24 ocasiona o aparecimento da mensagem

?VALOR ILEGAL ERRO

HTAB

HTAB imed e prog
HTAB aexpr

Presuma que a linha na qual o cursor está localizado tenha 255 posições, de 1 até 255. Independentemente da largura da janela de texto que você tenha posicionado, as posições 1 até 40 estão na linha corrente, posições 41 a 80 estão na próxima linha abaixo, e assim por diante. HTAB move o cursor para a posição que está /aexpr/ posições da extremidade esquerda da linha corrente na tela. Os movimentos de HTAB são relativos à margem esquerda da janela de texto, mas independentes da largura da linha. HTAB pode mover o cursor fora da janela de texto, mas apenas o suficiente para exibir um caráter.

Para colocar o cursor na posição mais à esquerda da linha corrente, use HTAB 1.

HTAB 0 move o cursor para a posição 256.

Se a /aexpr/ é negativa ou maior que 255, é exibida a mensagem

?VALOR ILEGAL ERRO

TAB

TAB imed e prog
TAB (aexpr)

TAB deve ser usada em um comando PRINT, e /aexpr/ deve estar entre parênteses. TAB move o cursor para a posição que está /aexpr/ posições da margem esquerda da janela do texto se /aexpr/ é maior do que o valor atual da posição do cursor em relação à margem esquerda. Se /aexpr/ é menor do que o valor atual da posição do cursor, então o cursor não é movido. TAB nunca move o cursor para a esquerda (use HTAB para isto).

Se TAB indicar uma posição além do limite direito da tela, o cursor é movido para a margem esquerda da linha imediatamente posterior, e o espacejamento continua a partir de lá.

TAB (Ø) coloca o cursor na posição 256. /aexpr/ deve estar no intervalo de Ø a 255, ou é apresentada a mensagem

?VALOR ILEGAL ERRO

TAB é analisada como uma palavra reservada se o próximo caráter diferente de branco é um parêntese esquerdo.

POS

POS imed e prog
POS (aexpr)

Devolve a posição horizontal atual do cursor em relação à margem esquerda da tela.

Na margem esquerda, obtém Ø. Embora /aexpr/ esteja ali apenas para separar os parênteses, ela é avaliada de qualquer forma, de modo que o seu conteúdo é irrelevante, podendo ser um número, uma cadeia ou um nome de variável. Se /aexpr/ é um conjunto de caracteres que não pode ser um nome de variável, os caracteres devem estar entre aspas.

Observe que para HTAB e TAB as posições são numeradas a partir de 1, mas para POS e SPC (que é apresentado adiante) elas são numeradas a partir de Ø. Consequentemente

```
PRINT TAB (23); POS (Ø)
```

causará a exibição de 22, enquanto que

```
PRINT SPC (23); POS (Ø)
```

causará a exibição de 23.

SPC

```
SPC imed e prog
SPC (aexpr)
```

Deve ser usado em um comando PRINT. Tem por função, introduzir /aexpr/ espaços entre o ítem anteriormente exibido (ou a margem esquerda da tela), e o próximo ítem a ser apresentado, se o comando SPC for concatenado com os ítens precedentes e seguintes, por justa posição ou por pontos e vírgulas interpostos. SPC (Ø) não introduz qualquer espaço.

/aexpr/ deve estar no intervalo Ø a 255, inclusive, ou aparecerá a mensagem

```
?VALOR ILEGAL ERRO
```

Contudo, um SPC (aexpr) pode ser concatenado a outro. Por exemplo:

```
PRINT SPC (25Ø) SPC (139) SPC (255)
```

e assim por diante. Dessa forma, pode-se fornecer arbitrariamente grandes espaços positivos.

Observe que enquanto HTAB move o cursor para posição absoluta relativa à margem esquerda da tela, SPC (aexpr) move o cursor um dado número de espaços de distância do ítem previamente exibido.

Um espacejamento além do limite mais à direita da tela ocasiona a movimentação para a margem esquerda da próxima linha.

Quando exibindo em campos tabulados, o espacejamento pode ser dentro do campo tabulado, ou passar de um campo para outro, ou pode ocupar seu próprio campo tabulado.

Se /aexpr/ é um número real, ele é convertido a um número inteiro.

SPC é analisado como uma palavra reservada somente se o próximo caráter diferente de branco é um parêntese esquerdo.

HOME

HOME imed e prog
HOME

Este comando não usa parâmetros. Ele move o cursor para o canto esquerdo superior da janela de texto e limpa completamente a tela. Essa mesma operação pode ser obtida por "CALL - 936".

CLEAR

CLEAR imed e prog
CLEAR

Como no caso anterior, este comando também não usa parâmetros. Ele zera todas as variáveis, matrizes e cadeias. E reposiciona os endereçadores de pilhas.

FRE

FRE imed e prog
FRE (expr)

FRE devolve a quantidade de memória (em bytes) ainda disponível ao usuário. Você pode às vezes ter mais memória do que espera, uma vez que o CCE BASIC armazena cadeias duplicadas uma única vez. Isto é, se A$ = "BOLA" então a cadeia "BOLA" será armazenada somente uma vez.

Se o número de bytes de memória livre excede a 32767, FRE (expr) devolve um número negativo. Somando 65535 a este número teremos o número correto de bytes livres na memória.

FRE (expr) devolve o número de bytes restantes abaixo do espaço de memória para cadeias e acima do espaço de matrizes numéricas e apontadores de matrizes numéricas e cadeias (veja mapa de memória no apêndice I). HIMEM: pode ser posicionado além da localização mais alta de memória RAM porém nesse caso, FRE pode devolver um número sem significado excedendo a capacidade de memória do computador.

Quando o conteúdo de uma cadeia é alterado durante o curso de um programa, (exemplo A$ que correpondia a "GATO" torna-se A$ = "PATO"), o CCE BASIC não elimina "GATO", porém apenas abre um novo arquivo para "PATO". Como resultado, caracteres inúteis lentamente preenchem o espaço a partir do HIMEM: em direção ao espaço das matrizes. O CCE BASIC automaticamente "limpará a casa" quando estes dados velhos chegarem ao espaço de arranjos livre, porém se você está usando algum desses espaços para programas em linguagem de máquina ou como memória intermediária de páginas de Alta Resolução, eles poderão ser danificados.

Usando um comando da forma

```
X= FRE (Ø)
```

periodicamente e seu programa forçará a limpeza de casa para evitar tais eventos.

Embora /expr/ seja usada apenas para separar os parênteses, ela é avaliada e não deve ser ilegal.

FLASH, INVERSE E NORMAL

```
FLASH imed e prog
INVERSE imed e prog
NORMAL imed e prog
FLASH
INVERSE
NORMAL
```

Estes três comandos são usados para posicionar modalidades de saída em vídeo. Eles não usam parâmetros e não afetam a exibição de caracteres conforme você os digita para memória do computador nem caracteres já existentes na tela.

FLASH define a modalidade intermitente de vídeo, de modo que a saída do computador é mostrada alternadamente na tela em branco sobre fundo preto e invertido para preto sobre fundo branco.

INVERSE especifica a modalidade de vídeo de modo que a saída do computador seja apresentada em vídeo reverso, ou seja, em letras pretas sobre um fundo branco.

NORMAL permite em qualquer dos dois casos, voltar à modalidade usual.

SPEED

SPEED imed e prog
SPEED = aexpr

Define a velocidade na qual os caracteres são enviados para a tela ou outros dispositivos de entrada/saída. A menor velocidade é Ø; a maior é 255. Valores fora desse intervalo causarão a exibição da mensagem.

?VALOR ILEGAL ERRO

ESC-A, ESC-B, ESC-C E ESC-D

ESC-A imed
ESC-B imed
ESC-C imed
ESC-D imed

A tecla [ESC], pode ser usada conjuntamente com as teclas de letras A ou B ou C ou D para mover o cursor uma posição, na direção indicada pela tabela a seguir. Para esta operação, digite primeiramente [ESC] e em seguida a letra desejada.

COMANDO	MOVE O CURSOR UM ESPAÇO PARA
ESC A	direita
ESC B	esquerda
ESC C	baixo
ESC D	cima

Este comando "escape" (obtidos através da tecla [ESC]) não afeta os caracteres movidos pelo cursor: os caracteres permanecem na tela e na memória. Por eles mesmos os comandos de "escape" também não afetam a linha do programa que está sendo digitada.

Para alterar uma linha de programa, liste a linha na tela e use os comandos de "escape" para mover o cursor de modo que ele se coloque exatamente sob o primeiro caráter da linha listada. Em seguida use a seta à direita e a tecla [REPT] para recopiar os caracteres a partir da tela, digitando um caráter diferente quando o cursor estiver exatamente sob o caráter que você deseja alterar. Se você não usou o comando LIST para listar a linha, não copie o caráter de "prontidão" (]) que aparece no início da linha. Finalmente, aperte a tecla [CR] para armazenar a linha ou executá-la.

Observação: através de [ESC] e as teclas [I], [J], [K] e [M] você também pode mover o cursor para cima, direita, esquerda e para baixo respectivamente, com a diferença que, em tais casos a tecla [ESC] só precisa ser pressionada a primeira vez. Assim, por exemplo, [ESC] [M] [ESC] [M] [ESC] [M] tem o mesmo efeito que [ESC] [M] [M] [M].

REPETIÇÃO

REPT imed

A tecla de repetição é [REPT]. Se você mantiver apertada a tecla de repetição e apertar uma tecla de caráter, o caráter será repetido (como se fosse digitado várias vezes em seguida).

SETA À DIREITA, SETA À ESQUERDA

Seta à direita imed
Seta à esquerda imed

A tecla seta-a-direita ([->]) move o cursor para a direita. À medida que o cursor se move, cada caráter que ele percorre na tela é copiado na memória do computador, como se você tivesse digitado o caráter. É utilizado, com a tecla de repetição, para poupar a redigitação de uma linha inteira quando somente uma pequena alteração é requerida.

A tecla seta-a-esquerda (←) move o cursor para a esquerda.

Cada vez que o cursor se move para esquerda, um caráter é apagado da linha do programa que você está digitando, sem considerar o que está sob o cursor. A tela é ignorada por este comando permanecendo inalterada.

CTRL—X

CTRL-X imed

Diz ao computador para ignorar a linha que está sendo digitada, sem apagar qualquer linha anterior de mesmo número. Uma barra inclinada para trás é exibida no fim da linha que deve ser ignorada, e o cursor salta para a coluna 0 da linha seguinte. Este comando também pode ser usado durante a resposta a uma instrução INPUT.

CAPíTULO 5

ARRANJOS E CADEIAS

Os comandos apresentados nesse capítulo seguem a mesma forma utilizada no capítulo anterior, indicando logo abaixo do nome de cada seção os modos de execução e sintaxes.

DIM

```
DIM   imed e prog
DIM   var índice [ ,{var índice}]
```

O comando DIM reserva espaço de memória para a matriz com o nome var. Dois bytes na memória são usados para armazenar um nome da variável da matriz, dois para o tamanho da matriz, um para o número de dimensões, e dois para cada dimensão. Conforme abaixo, a quantidade de espaço alocado para os elementos de uma matriz depende de seus índices.

O número de elementos em uma matriz n-dimensional é:

(índice 1+1) * (índice 2+1) *...* (índice n+1).

Por exemplo DIM SHOW (4,5,3) reserva 5x6x4 elementos (120 elementos). Alguns de seus elementos são:

```
SHOW (4,4,1)
SHOW (Ø,Ø,2)
```

e assim por diante.

O número máximo de dimensões para um arranjo é 88, mesmo que cada dimensão possa conter somente um elemento:

DIM A(Ø,Ø,...Ø) onde há 89 zeros dá a mensagem

?FALTA MEMORIA ERRO

mas DIM A (Ø,Ø,...Ø) onde há 88 zeros não dá a mensagem.

Na prática, contudo, o tamanho das matrizes são frequentemente limitados muito mais, pela quantidade de memória disponível. Cada elemento inteiro de matriz ocupa 2 bytes (16 bits) na memória. Cada elemento real de matriz ocupa 5 bytes (40 bits) na memória. Variáveis de matrizes tipo cadeia, usam 3 bytes para cada elemento (um para o comprimento, dois para endereçador de localização), armazenado como uma matriz inteira quando é dimensionado. Como as cadeias, elas mesmas são armazenadas pelo programa, elas ocupam um byte adicional por caráter. Veja o mapa do apêndice I.

Se um elemento da matriz é usado em um programa antes que aquela variável seja dimensionada, é definido o máximo índice de 10 elementos para cada dimensão.

O uso de uma variável, cujo índice é maior do que o máximo designado, ou a qual chama por um número de dimensões diferente daquele especificado em uma declaração DIM, causará o aparecimento da mensagem:

?INDICE ILEGAL ERRO

Caso o nome de uma matriz já dimensionada seja usado novamente com a instrução DIM, será apresentada a mensagem:

?REDIMENSIONAR ERRO

As cadeias individuais em uma matriz tipo cadeia não são dimensionadas, porém aumentam e diminuem conforme necessário. A declaração

CCE$ (5) = ""

libera o espaço atribuído para a cadeia CCE$ (5).

Os elementos das cadeias são zerados quando os comandos RUN ou CLEAR são executados.

LEN

LEN imed e prog
LEN (sexpr)

Esta função devolve o número de caracteres de uma cadeia. Se o argumento é resultante de concatenação de cadeias cujo comprimento combinado é maior do que 255, é dada a mensagem

?"STRING" LONGA ERRO

STR$

STR$ imed e prog
STR$ (aexpr)

Esta função converte /aexpr/ em uma cadeia que representa aquele valor. /aexpr/ é avaliada antes de ser convertida em uma cadeia.

STR$ (100 000 000 000) devolve 1E + 11

Se /aexpr/ excede o limite para os números reais, então é exibida a mensagem

?S/ESPACO ERRO

VAL

VAL imed e prog
VAL (sexpr)

Esta função tenta interpretar uma cadeia como um número real ou um número inteiro, devolvendo o valor daquele número.

O primeiro caráter da cadeia deve ser numérico (espaços no início são aceitáveis), ou 0 é devolvido. Cada caráter seguinte é igualmente examinado, até que o primeiro caráter definitivamente não numérico seja encontrado (brancos, pontos decimais, sinais + e - e E são todos aceitáveis como caracteres numéricos se no contexto correto). O primeiro caráter não numérico, e todos os caracteres sequentes são ignorados, e a cadeia até aquele ponto é avaliada como um número real ou um inteiro.

Se o argumento da função VAL for uma concatenação de cadeias consistindo de mais do que 255 caracteres, é dada a mensagem

?"STRING" LONGA ERRO

Se o valor absoluto do número devolvido é maior do que 1E38, ou se o número contém mais de 38 dígitos (incluindo zeros à direita), é apresentada a mensagem

?S/ESPACO ERRO

CHR$

CHR$ imed e prog
CHR$ (aexpr)

Tal função devolve o caráter ASCII que corresponde ao valor de /aexpr/. /aexpr/ deve estar entre 0 e 255, inclusive, ou aparece a mensagem

?VALOR ILEGAL ERRO

Números reais são convertidos em números inteiros.

ASC

ASC imed e prog
ASC (sexpr)

A função ASC retorna o código ASCII (não necessariamente o número mais baixo) do primeiro caráter de /sexpr/. Os códigos ASCII no intervalo entre 96 a 255 gerarão caracteres no computador o qual os apresenta no intervalo de 0 a 95. Entretanto, embora CHR$ (65) devolva um A e CHR$ (193) também devolva um A, o CCE BASIC não reconhece os dois como o mesmo caráter quando usando operadores lógicos de cadeias.

Se uma cadeia é o argumento, ela deve estar entre aspas, e não deverá haver aspas dentro da cadeia. Se a cadeia for nula, é dada a mensagem

?VALOR ILEGAL ERRO

Uma tentativa para usar a função ASC sobre CTRL @ resulta na mensagem

?SINTAX ERRO

LEFT$

LEFT$ imed e prog
LEFT$ (sexpr, aexpr)

Esta função devolve os primeiros (mais à esquerda) /aexpr/ caracteres de /sexpr/. Exemplo:

```
PRINT LEFT$ ("LILIAN",4)
LILI
```

Nenhuma parte desse comando pode ser omitida.

Se /aexpr/ < 1 ou /aexpr/ > 255, é exibida a mensagem

?VALOR ILEGAL ERRO

Caso /aexpr/ seja um número real, ele é convertido em um número inteiro.

Se /aexpr/ é maior que o comprimento da cadeia, somente os caracteres que consistem a cadeia são devolvidos. Quaisquer posições extras são ignoradas.

Se "$" é omitido do nome do comando, LEFT é tratado como um nome de variável aritmética e é exibida a mensagem

?TIPO INCOMPAT ERRO

RIGHT$

RIGHT$ imed e prog
RIGHT$ (sexpr, aexpr)

Esta função devolve os últimos (mais à direita) /aexpr/ caracteres de /sexpr/:

```
PRINT RIGHT$ ("EDGARD" + "IDO",6)
ARDIDO
```

Nenhuma parte deste comando pode ser omitida.

Se /aexpr/ é maior que o número de caracteres da cadeia, RIGHT$ devolve a cadeia inteira.

Se /aexpr/ < 1 ou /aexpr/ > 255, é exibida a mensagem

?VALOR ILEGAL ERRO

Se o "$" é omitido do nome do comando, RIGHT é tratado como um nome de variável aritmética e é exibida a mensagem

?TIPO INCOMPAT ERRO

MID$

```
MID$    imed e prog
MID$    (sexpr, aexpr 1[, a expr 2])
```

A função MID$ chamada com dois argumentos devolve a subcadeia começada no /aexpr 1/-ésimo caráter de /sexpr/, e finalizada no último caráter de /sexpr/.

```
PRINT MID$ ("ADELIA",4)
LIA
```

A função MID$ chamada com três argumentos devolve /aexpr 2/ caracteres de /sexpr/, começando com o /aexpr 1/-ésimo caráter e continuando para a direita.

```
PRINT MID$ ("ADELIA", 2,4)
DELI
```

Se /aexpr 1/ é maior que o comprimento da cadeia, MID$ devolve uma cadeia nula. Se /aexpr 1/ + /aexpr 2/ excede o comprimento de /sexpr/ (ou 255, o comprimento máximo de qualquer cadeia), os valores excedentes são ignorados. MID$ (A$, 255,255) devolve um caracter se LEN (A$)=255, caso contrário, uma cadeia nula é devolvida.

Se ou /aexpr 1/ ou /aexpr2/ estiverem fora do intervalo de 1 a 255, inclusive, então será exibida a mensagem

```
?VALOR ILEGAL ERRO
```

Se o "$" é omitido do nome do comando, MID é tratado como um nome de variável aritmética, e é exibida a mensagem

```
?TIPO INCOMPAT ERRO
```

STORE E RECALL

```
STORE   imed e prog
RECALL  imed e prog
STORE   avar
RECALL  avar
```

Estes comandos armazenam e chamam de volta matrizes de fita cassete.

Os nomes de matrizes não são armazenados com seus valores, de modo que uma matriz pode ser lida de volta usando um nome diferente daquele que foi usado com o comando STORE.

As dimensões da matriz "chamada" por RECALL devem ser idênticas às dimensões da matriz original conforme ela foi armazenada. Por exemplo, se uma matriz dimensionada por DIM A (5,5,5) fosse armazenada, então poderia ser chamada por uma matriz dimensionada por B (5,5,5). A não observância dessa regra resultará em números misturados na matriz chamada de volta, zeros extras na matriz ou a mensagem

```
?FALTA MEMORIA ERRO
```

Em geral, você receberá a mensagem "?FALTA MEMORIA ERRO", somente quando o número total de elementos reservados para a matriz chamada de volta, foi insuficiente para conter todos os elementos que foram armazenados. Por exemplo:

```
DIM A (5,5,5)
STORE A

salvas na fita cassete 6 * 6 * 6 elementos

DIM B (5,35)
RECALL B

resulta na mensagem

ERR
```

e números misturados na matriz B, porém a execução do programa continua, entretanto:

```
DIM B (5,25)
RECALL B

causará a exibição da mensagem

?FALTA MEMORIA ERRO
```

e a execução do programa cessa. Neste caso, a matriz B contendo 6 * 26 elementos foi insufuciente para conter todos os elementos da matriz A.

Pode-se entretanto chamar uma matriz armazenada em uma nova matriz, onde os elementos não definidos são preenchidos com zeros, através das seguintes regras:

1- somente a última dimensão da matriz chamada de volta pode ser maior do que a última dimensão da matriz armazenada;

2- o número total dos elementos chamados de volta devem ser pelo menos igual ao número dos elementos armazenados.

Se as regras forem seguidas, usando-se dimensões que são comuns a ambas as matrizes (estas devem ser as primeiras dimensões), então alguém pode chamar de volta uma matriz com mais dimensões do que a matriz que foi armazenada. Nesse caso é exibida uma mensagem de erro, mas a execução do programa continua. Por exemplo, se armazenarmos a matriz:

```
DIM A (5,5,5)
STORE A
```

então:

```
DIM B (5,5,5,5)
RECALL B
```

funcionará corretamente (após uma mensagem de erro, e com muitos zeros extras na matriz B) porém

```
DIM B (5,5,3,5)
RECALL B
```

preencherá a matriz B com números misturados (após uma mensagem de erro) e

```
DIM B (5,5,1,1)
RECALL B
```

causará a mensagem

```
?FALTA MEMORIA ERRO
```

porque os 6*6*2*2 elementos na matriz B são menos do que os 6*6*6 elementos armazenados na matriz A.

Somente as matrizes numéricas podem ser armazenadas. As matrizes tipo cadeia precisam ser convertidas para matrizes de números inteiros usando a função ASC a fim de serem armazenadas.

Embora STORE e RECALL refiram-se a variáveis sem utilizarem índice ou dimensão, somente matrizes podem ser armazenadas ou chamadas de volta. O programa:

```
100 A (3) = 63
110 A = 42
120 STORE A
```

armazena em fita os elementos da matriz A (Ø) até A (1Ø) (por "default", a matriz é dimensionada para onze elementos), não a variável A (que vale 27 no programa).

Não há nenhuma mensagem ou qualquer outro sinal emitido pela instrução STORE; o usuário deve ter o gravador funcionando na modalidade de gravação quando a instrução é executada. Um "bip" assinala o início da gravação, e outro "bip" assinala o fim.

O programa

```
3ØØ DIM B (5,13)
31Ø B = 4
32Ø RECALL B
```

lê a partir da fita os 84 (6*14) elementos da matriz B. O valor da variável B não é alterado. Neste caso, também não há mensagem; um "bip" assinala o fim da leitura ou transmissão de dados.

Se STORE ou RECALL contém um nome de matriz não dimensionada anteriormente, é dada a mensagem

```
?FIM DE DATA ERRO
```

STORE ou RECALL podem ser interrompidos somente por RESET .

Se as palavras reservadas STORE ou RECALL são usadas como os primeiros caracteres de qualquer nome de variável, os comandos podem ser executados ANTES de que seja dada qualquer mensagem

```
?SINTAX ERRO
```

Por exemplo, a declaração

```
STORE VALOR = 5
```

causará a mensagem

```
?FIM DE DATA ERRO
```

a menos que tenha sido definido uma matriz cujo nome comece com os caracteres VA. No último caso, o CCE BASIC tentará armazenar a matriz: primeiro você ouvirá um "bip", em seguida, um segundo "bip", finalmente será exibida a mensagem

```
?SINTAX ERRO
```

quando o CCE BASIC tentar analisar o resto da declaração, "=5".

A declaração

```
RECALL OUS = 234
```

causará a exibição da mensagem

```
?FIM DE DATA ERRO
```

a menos que tenha sido definida uma matriz, cujo nome comece com os caracteres OU. No último caso, o CCE BASIC esperará indefinidamente por uma matriz do gravador de fita cassete. A única maneira de recuperar o controle do computador, é apertar a tecla [RESET].

CAPÍTULO 6

COMANDOS DE ENTRADA / SAÍDA

No capítulo 3, veja também LOAD e SAVE
No capítulo 5, veja também STORE e RECALL

101- INPUT

103- GET

104- DATA

106- READ

107- RESTORE

107- PRINT

108- IN#

109- PR#

110- LET

110- DEF FN

INPUT

INPUT prog
INPUT [cadeia;] var [{svar}]

Se a cadeia opcional é omitida, a instrução INPUT exibe um ponto de interrogação, e espera que o usuário digite um número (se var é uma variável aritmética) ou caracteres (se var é uma variável cadeia). O valor deste número ou cadeia é associado a /var/.

Quando a cadeia está presente, ela é exibida exatamente como especificado; nenhum ponto de interrogação, espaços ou outra pontuação é apresentado após a cadeia. Observe que somente uma cadeia opcional pode ser usada. Ela deve aparecer diretamente após INPUT e ser seguida por ponto e vírgula.

INPUT aceitará como entrada numérica um número real ou um número inteiro, não uma expressão aritmética. Os caracteres espaço, +, -, E e o ponto decimal são partes legítimas de uma entrada numérica. INPUT aceitará qualquer destes caracteres no formato correto (exemplo: + E - é correto, + - não é); tais entradas, por si, são avaliadas como zero.

Em entradas numéricas, espaços em qualquer posição são ignorados. Se uma entrada numérica é um inteiro, uma vírgula ou dois-pontos, provocam a exibição da mensagem:

?REENTRE

e a instrução INPUT é reexecutada.

De forma similar, uma entrada atribuida a uma variável tipo cadeia, deve ser uma cadeia simples ou literal, não uma expressão tipo cadeia. Espaços precedendo o primeiro caráter são ignorados. Se a entrada é uma cadeia, então uma aspa em qualquer lugar dentro da cadeia causará a mensagem:

?REENTRE

Contudo, dentro de uma cadeia, todos os caracteres exceto aspas, CTRL-X, CTRL-M são aceitos como caracteres para a cadeia. Isto inclui dois-pontos, vírgula e espaços.

Se a entrada é uma literal, então as aspas são aceitas como caracteres em qualquer parte do literal, exceto o primeiro caráter diferente de espaço.

Espaços seguindo o último caráter são aceitos como parte da literal. Todavia, a vírgula e o dois-pontos (e o CTRL-X e o CTRL-M) não são aceitos como caracteres literais.

Se o usuário simplesmente aperta a tecla [CR] quando uma entrada numérica é esperada, é exibida a mensagem:

?REENTRE

e a instrução INPUT é reexecutada. Se a tecla [CR] sozinha é digitada quando uma entrada cadeia é esperada, a entrada é interpretada como uma cadeia nula e a execução do programa continua.

Variáveis sucessivas obtém valores digitados sucessivamente. Variáveis cadeias e variáveis aritméticas podem ser misturadas no mesmo comando INPUT, mas a entrada do usuário deve ser apropriada para cada tipo. As entradas digitadas podem ser separadas por vírgulas ou [CR]. Como resultado, se um usuário digitar vírgula em uma entrada que não comece com um ponto de interrogação, as vírgulas são interpretadas como separadores de entradas. Isto é verdadeiro, mesmo quando somente uma entrada é esperada.

Se um dois-pontos (:) é digitado em uma entrada a um INPUT que não comece com um ponto de interrogação, todos os caracteres digitados subsequentemente são ignorados. Após um dois-pontos, vírgulas também são ignoradas, assim o início de outra entrada deve ser assinalado por um [CR].

Quando um [CR] é encontrado antes que todas as /var/ tenham tido entradas atribuídas, dois pontos de interrogação são apresentadas para indicar que é esperada uma entrada adicional. Quando um [CR] é encontrado, se a entrada contém mais campos de entradas do que o comando esperado, ou se existe um ponto e vírgula no final da entrada esperada (porém não dentro de uma cadeia), então é apresentada a mensagem:

?EXTRA IGNORADO ERRO

e a execução do programa continua.

Se dois-pontos ou uma vírgula é o primeiro caráter de uma entrada ao INPUT, a entrada é avaliada como zero ou com uma cadeia nula.

Observe que no comando INPUT a cadeia opcional deve ser seguida por um ponto e vírgula, porém as variáveis devem ser separadas por vírgulas.

CTRL-C pode interromper um comando INPUT, mas somente se ele for o primeiro caráter digitado. O programa é interrompido após a digitação de [CR]. Uma tentativa para continuar a execução após tal parada resulta na mensagem

?SINTAX ERRO

CTRL-C é tratado como qualquer outro caráter, se não for o primeiro caráter digitado.

Tentar usar o comando INPUT na modalidade de execução imediata causa a mensagem

?DIRETO ILEGAL ERRO

GET

GET prog
GET var

Aceita um único caráter, a partir do teclado sem exibi-lo na tela, e sem requerer que a tecla [CR] seja pressionada.

Embora o CCE BASIC não tenha sido projetado com a intenção de usar a instrução GET para obter valores para variáveis aritméticas, você pode usar

GET avar

sujeita às seguintes limitações estritas:

- a obtenção de um dois-pontos ou uma vírgula resulta na mensagem

  ?EXTRA IGNORADO ERRO

  seguida pela devolução de um zero como valor digitado;

- o sinal de mais, o sinal de menos, CTRL-C, E, espaço e o ponto, todos devolvem um zero como valor digitado;

- a digitação de um [CR] ou uma entrada não-numérica, causará a exibição da mensagem:

  ?SINTAX ERRO

Com ONERR GOTO...RESUME, dois erros consecutivos na instrução GET deixarão o sistema suspenso até que [RESET] seja apertada.

Se GOTO é substituído por RESUME, tudo corre bem até o 43º erro na instrução GET (em qualquer ordem), quando o programa salta para o Monitor. Para recuperar, use [RESET] CTRL-C [CR].

Por causa destas limitações, recomendamos que os números sejam obtidos através de:

```
GET svar
```

e a cadeia resultante seja transformada em número, usando a função VAL.

DATA

```
DATA prog
DATA [literal|cadeia|real|inteiro][{,[literal|
     cadeia|real|inteiro ]}]
```

Este comando cria uma lista de elementos que podem ser usados pelos comandos READ. Cada comando DATA soma seus elementos a uma lista de elementos construída por comandos DATA anteriores, respeitando-se a numeração das linhas.

O comando DATA não tem que preceder o comando READ, eles podem aparecer em qualquer ponto do programa.

Os elementos do comando DATA que são lidos como variáveis aritméticas, geralmente seguem as mesmas regras das entradas ao INPUT atribuídas às variáveis aritméticas. Entretanto, os dois-pontos não podem ser incluídos como caráter em um elemento numérico de DATA.

Se CTRL-C é um elemento de DATA, ele não para o programa, mesmo que seja o primeiro caráter de um elemento. Com esta exceção, os elementos de dados que são lidos como variáveis tipo cadeia, seguem as mesmas regras das entradas ao INPUT, atribuídas às variáveis cadeia:

- espaços antes do primeiro caráter, e seguintes à cadeia são sempre ignorados;

- qualquer ponto de interrogação que apareça dentro de uma cadeia, causa a mensagem:

?SINTAX ERRO

porém, todos os outros caracteres são aceitos, incluindo o dois-pontos e a vírgula (mas, não incluindo o CTRL-X e o CTRL-M).

Se um elemento é uma literal, então a aspa é aceita como um caráter válido em qualquer lugar exceto como o primeiro caráter diferente de espaço; o dois-pontos, a vírgula, CTRL-X e CTRL-M não são aceitos. Veja INPUT para mais detalhes.

Os elementos de DATA podem ser qualquer mistura de números reais, inteiros, cadeias e literais. Se o comando READ tenta atribuir um elemento de DATA que é uma cadeia ou uma literal para uma variável aritmética, é dada a mensagem:

?SINTAX ERRO

para a linha apropriada de DATA.

Se a lista de elementos em um comando DATA contém um elemento "não-existente", então um zero (numérico) ou a cadeia nula é devolvida para aquele elemento dependendo da variável para a qual o elemento é atribuído. Um elemento "não-existente" ocorre num comando DATA quando qualquer uma das seguintes afirmativas for verdadeira:

1) não há caráter diferente de branco entre DATA e [CR];

2) a vírgula é o primeiro caráter diferente de espaço seguindo DATA;

3) não há caráter diferente de espaço entre duas vírgulas;

4) a vírgula é o último caráter diferente de espaço antes do [CR].

Assim, quando este comando é lido

100 DATA,,

ele pode devolver até três elementos consistindo de zeros ou cadeias nulas.

Quando usado no modo de execução imediata, DATA não causa um SINTAX ERRO, porém, seus elementos de dados não são disponíveis a um comando READ.

READ

READ imed e prog
READ var [{,var}]

Quando o primeiro comando READ é executado em um programa, a primeira variável fica com o valor do primeiro elemento na lista do comando DATA (a lista de dados consiste de todos os elementos de todos os comandos DATA no programa armazenado). A segunda variável (se houver uma), fica com o valor do segundo elemento na lista de dados do comando DATA, e assim por diante. Quando o comando READ termina a execução, ele deixa um indicador de lista de dados após o último elemento de dados usado. O próximo comando READ executado (se houver), começa usando a lista de dados a partir da posição do endereçador. Ou RUN ou RESTORE posiciona o endereçador para o primeiro elemento da lista do comando DATA.

Uma tentativa de ler mais dados do que os contidos na lista, produz a mensagem:

? FIM DE DATA ERRO EM numlinha

onde/numlinha/é o número da linha do comando READ que pediu dados adicionais.

No modo imediato, você só pode ler elementos a partir de comandos DATA, os quais existem como linhas em um programa armazenado correntemente. Os elementos de DATA, em um programa armazenado, podem ser lidos mesmo que o programa armazenado não tenha sido executado. Se nenhum comando DATA foi armazenado, é exibida a mensagem:

? FIM DE DATA ERRO

A execução de um programa no modo imediato não posiciona o indicador da lista de dados para o primeiro elemento da lista do comando DATA.

Dados extras, que não são lidos ficam inalterados.

RESTORE

```
RESTORE imed e prog
RESTORE
```

RESTORE não tem parâmetros ou opções. Este comando simplesmente move o endereçador da lista de dados (veja os comandos READ e DATA) de volta ao início.

PRINT

```
PRINT imed e prog
PRINT [{expr }[{,|; [{expr}]}]][,|;]
PRINT  ;
PRINT {,}
```

O ponto de interrogação (?) pode ser usado como uma abreviação para PRINT: ele é listado como PRINT.

Sem qualquer opção, PRINT causa um pulo de linha na tela. Quando as opções são usadas, os valores da lista das expressões especificadas são exibidos. Se nem uma vírgula, nem um ponto-e-vírgula termina a lista, a alimentação de linha e retorno são executados em seguida ao último ítem apresentado. Se um ítem na lista é seguido por uma vírgula, então o primeiro caráter do próximo ítem a ser exibido aparecerá na primeira posição do próximo campo de tabulação disponível.

O primeiro campo de tabulação compreende as 16 posições mais à esquerda da tela. O segundo campo de tabulação ocupa as próximas 16 posições (17 a 32), e está disponível se a posição 16 não estiver preenchida. O terceiro campo de tabulação consiste nas 8 posições restantes (33 a 40), e está disponível somente se as posições 24 a 32 não foram utilizadas.

O terceiro campo de tabulação PRINT não funciona corretamente se a área de texto da tela é posicionada para menos do que 33 posições de largura; o primeiro caráter pode ser exibido fora da área de texto. HTAB também pode fazer com que o PRINT exiba um primeiro caráter fora da janela (área) de texto.

Se um ítem é seguido por um ponto-e-vírgula, então o próximo ítem é concatenado: ele é exibido imediatamente após o anterior, sem interposição de espaços.

Itens listados sem interposição de vírgulas ou ponto-e-vírgulas, são concatenados se os ítens podem ser analisados sem problemas de síntaxe. Isto é melhor ilustrado pelos exemplos:

```
A=1 : B=2 : C=3 : C(4)=5 : C5=7

PRINT 1/3 (2 * 4) 51, :PRINT 1(A) 2(B) 3C(4) C5
.33333333851                 1122357

PRINT 3.4.5.6,               :PRINT A,"B",C.4
3.4.5.60                       1Ø B.3.4
```

Note que, se PRINT não puder interpretar um ponto como ponto decimal, ele o trata como o número Ø, conforme ilustrado nos exemplos acima.

PRINT seguido por uma lista de ponto-e-vírgulas não faz mais do que um PRINT sozinho, mas isto é legal.

PRINT seguido por uma lista de vírgulas espaceja um campo de tabulação para cada vírgula até o limite de 239 caracteres por instrução. PRINT A$+B$ dá a mensagem:

```
? "STRING" LONGA ERRO
```

se o comprimento das cadeias concatenadas for maior do que 255. Entretanto, você pode exibir a aparente concatenação usando

```
PRINT A$ B$
```

sem se preocupar com o seu comprimento.

IN#

```
IN # imed e prog
IN # aexpor
```

Seleciona a entrada a partir do slot /aexpor/. É usado para especificar qual periférico estará fornecendo entrada para comandos INPUT subsequentes. Os periféricos podem estar nos slots 1 a 7, conforme indicado por /aexpor/.

IN # Ø indica que a entrada subsequente será a partir do teclado em lugar das unidades periféricas. O slot Ø não é endereçável a partir do CCE BASIC para uso com um dispositivo periférico.

Se nenhum periférico está no slot /aexpor/, o sistema será suspenso. Para recuperar, use [RESET] CTRL-C [CR] .

Se /aexpor/ é menor do que Ø ou maior do que 255, é exibida a mensagem

? VALOR ILEGAL ERRO

Se /aexpor/ está no intervalo de 8 a 255, se torna imprevisível o slot a ser usado pelo CCE BASIC.

Para transferência similar à saída, veja o próximo comando.

PR#

PR # imed e prog
PR # aexpor

PR # transfere saída para o slot /aexpor/, onde /aexpor/ deve estar no intervalo 1 a 7, inclusive: PR#Ø retorna à saída para a tela, e não para o slot Ø.

Se nenhum periférico está no slot especificado, o sistema será suspenso. Para recuperar, use [RESET] CTRL-C [CR].

Se /aexpor/ é menor do que Ø ou maior do que 255, será exibida a mensagem:

? VALOR ILEGAL ERRO

Se /aexpor/ está no intervalo de 8 a 255, (A) é imprevisível o slot a ser usado.

Para transferência similar de entrada, veja IN#.

LET

```
[LET] imed e prog
[LET] avar [ índice] = aexpor
[LET] avar [ índice] = aexpor
```

Ao nome da variável à esquerda, é atribuído o valor da cadeia ou expressão da direita. LET é opcional:

```
LET A = 2 e
A = 2
```

são equivalentes.

A mensagem:

```
?TIPO INCOMPAT
```

é exibida se você tenta dar

- um nome de variável cadeia para uma expressão aritmética;

- um nome de variável cadeia para uma literal;

- um nome de variável aritmética para uma expressão cadeia.

Se você tenta dar um nome de variável aritmética para uma literal, CCE BASIC tenta analisar a literal como uma expressão aritmética.

DEF FN

```
DEF prog
FN  imed e prog
DEF FN nome (avar real) = aexpr 1
FN nome (aexpr 2)
```

Permite ao usuário definir funções em um programa. Primeiro, a função /FN nome/ é definida usando DEF. Uma vez que a linha de programa que define a função tenha sido executada, a função pode ser usada na forma FN (argumento), onde o argumento /aexpr 2/ pode ser qualquer expressão aritmética. A /aexpr 1/ da definição pode estar somente em uma linha de programa de comprimento, a FN definida pode ser usada onde quer que funções aritméticas possam ser usadas em CCE BASIC.

Tais funções podem ser redefinidas durante o curso de um programa. As regras para uso de variáveis aritméticas ainda se aplicam, em particular, os dois primeiros caracteres de um nome devem ser únicos. Quando estas linhas

```
1Ø DEF FN ABC (I) = COS (I)
2Ø DEF FN ABT (I) = TAN (I)
```

são executadas, o CCE BASIC reconhece a definição de uma função FN AB na linha 1Ø e na linha 2Ø, a função FN AB é redefinida.

Na instrução DEF /avar real/ é uma variável fictícia.

Quando a função definida pelo usuário /FN nome/ é usada posteriormente, ela é chamada com um argumento aexpr 2. Este argumento é substituído por /avar real/, onde quer que ela apareça na /aexpr 1/ da definição. /aexpr 1/ pode conter qualquer número de variáveis, porém, somente uma corresponde à variável fictícia /avar real/, e consequentemente corresponde à variável argumento.

A /avar real/ da definição não precisa aparecer em /aexpr 1/. Nesse caso, quando a função é usada mais tarde no programa, o argumento da função é ignorado na avaliação de /aexpr 1/. Mesmo neste caso, contudo, o argumento da função é avaliado em si mesmo, assim ele deve ser legal. Por exemplo:

```
1ØØ DEF FN A(W) = 2 * W + W
11Ø PRINT FN A (23)
12Ø DEF FN B (X) = 4+3
13Ø G = FN B (23)
14Ø PRINT G
15Ø DEF FN A (Y) = FN B (Z) + Y
16Ø PRINT FN A (G)


RUN
69          (FN A (23) = 2* 23 + 23)
 7          (FN B (qualquer coisa) = 7)
14          (nova FN A (7) = 7 + 7)
```

Se um comando de modo programado "DEF FN nome" não for executado antes de usar "FN nome", é exibida a mensagem:

```
?LINHA INDEFINIDA ERRO
```

Funções de cadeia definidas pelo usuário não são permitidas. Funções definidas usando um nome de variável inteira não são permitidas.

Quando uma nova função é definida por um comando DEF, são usados 6 bytes na memória para armazenar o ponteiro para a definição.

CAPÍTULO 7

COMANDOS DE DESVIO LÓGICO

GOTO

GOTO imed e prog
GOTO numlinha

Desvia para a linha cujo número é/numlinha/. Se tal linha não existir, ou se/numlinha/estiver ausente do comando GOTO, é exibida a mensagem:

?LINHA INDEFINIDA ERRO EM numlinha

onde /numlinha/é o número da linha do programa contendo a instrução GOTO.

IF...THEN e IF...GOTO

IF imed e prog
IF expr THEN instrução [{:instrução}]
IF expr THEN [GOTO] numlinha
IF expr [THEN] GOTO numlinha

Se /expr/ é uma expressão aritmética cujo valor não é Ø (e cujo valor absoluto é maior do que aproximadamente 2.93873E-39), então deve ser considerado verdadeiro, e quaisquer instruções seguintes ao THEN são executadas.

Se /expr/ é uma expressão aritmética cujo valor é Ø, (ou cujo valor absoluto é menor do que aproximadamente 2.9387E-39), quaisquer instruções seguintes as THEN são ignoradas, e a execução do programa passa para a instrução na próxima linha numerada do programa.

Quando o comando IF ocorre em um programa de execução imediata, se /expr/ é Ø todo restante do programa é ignorado.

Se /expr/ é uma expressão aritmética envolvendo expressões de cadeia e operadores lógicos de cadeia, /expr/ é avaliada pela comparação da ordem alfabética das expressões de cadeia, conforme determinado pelos códigos ASCII para os caracteres envolvidos (veja o apêndice K).

Comandos da forma

IF expr THEN

são válidos: nenhuma mensagem de erro é apresentada.

Um THEN sem um IF correspondente ou um IF sem THEN causará a exibição da mensagem:

?SINTAX ERRO

O CCE BASIC não foi projetado ou pretendeu permitir que as /expr/ dos comandos IF sejam expressões de cadeias, porém, variáveis tipo cadeias e cadeias podem ser usadas como /expr/ sob as seguintes condições estritas: se /expr/ é uma expressão tipo cadeia de qualquer espécie, então /expr/ é diferente de zero, mesmo que /expr/ seja uma variável cadeia à qual não tenha sido atribuída valor algum ou "Ø" ou a cadeia nula, (""). Entretanto, a cadeia nula como em:

```
IF "" THEN...
```

é válida como zero

```
IF cadeia THEN...
```

quando executada mais de duas ou três vezes em um dado programa, causa a exibição da mensagem:

?FORMULA COMPLEXA ERRO

Se /expr/ é uma variável tipo cadeia e a declaração anterior atribuiu cadeia nula para toda e qualquer variável desse tipo, então /expr/ é avaliada como Ø. Por exemplo, o programa

```
12Ø IF A$ THEN PRINT "A$"
13Ø IF B$ THEN PRINT "B$"
14Ø IF X$ THEN PRINT "X$"
```

após RUN exibe

```
A$
B$
X$
```

Entretanto adicionado a linha

```
1ØØ Q$ = ""
```

faz com que todas as 3 cadeias sejam avaliadas como zero e nenhuma saída é apresentada. Apagando a linha 1ØØ ou adicionando qualquer linha 11Ø tal como

```
1ØØ F = 3
```

permite que as 3 cadeias sejam avaliadas como não-nulas.

Antes de THEN a letra A causa problemas de análise:

```
IF PERLA THEN 230
```

é analisada como

```
IF PERL AT HEN 230
```

a qual gera uma mensagem na execução:

```
?SINTAX ERRO
```

As 3 formas são equivalentes:

```
IF A = 3 THEN 160
IF A = 3 GOTO 160
IF A = 3 THEN GOTO 160
```

FOR...TO...STEP

```
FOR imed e prog
FOR avar real = aexpr 1 TO aexpr 2 [STEP aexpr 3]
```

/avar/ é posicionado em /aexpr/, e os comandos seguintes ao FOR são executados até que um comando NEXT avar seja encontrado, onde /avar/ é o mesmo nome que aparece no comando FOR. Em seguida /avar/ é incrementado por /aexpr 3/. Caso /aexpr3/ não seja usado, o incremento valerá 1. A próxima /avar/ é comparada com /aexpr 2/, e se /avar/ > /aexpr 2/, a execução continua com o comando seguinte ao NEXT. Se /avar/ < = /aexpr 2/, a execução continua a partir do comando seguinte ao FOR.

Se /aexpr 3/ < 0 então a operação é ligeiramente diferente depois que /aexpr 3/ é somada a /avar/. Sendo /avar/ < /aexpr 2/ a execução continua com o comando seguinte ao NEXT. Se /avar/ > = /aexpr 2/, então a execução continua a partir do comando seguinte ao FOR.

As expressões aritméticas que formam os parâmetros do laço FOR podem ser números ou variáveis reais, ou números ou variáveis inteiras. Entretanto, /avar real/ deve ser uma variável real. Uma tentativa de usar uma variável inteira para /avar real/ resulta na mensagem:

```
?SINTAX ERRO
```

Como /avar/ é incrementada e comparada a /aexpr 2/, somente no fim do laço FOR...NEXT, a porção do programa dentro do laço é sempre executada pelo menos uma vez.

Os laços FOR...NEXT não se devem cruzar. Se houver cruzamento de tais laços será exibida a mensagem:

?"NEXT" SEM "FOR" ERRO

Se os laços FOR são utilizados com mais do que 1Ø níveis de profundidade, é exibida a mensagem:

?FALTA MEMORIA ERRO

Para executar um laço FOR...NEXT no modo de execução imediata, o comando FOR e o comando NEXT, devem ser incluídos na mesma linha, (uma linha tem até 239 caracteres).

Se a letra A é usada imediatamente antes do TO, não permita espaço entre o T e o O. FOR I = ADELIA TO 56 está correto,porém FOR I = ADELIA T O 56 é analisado como FOR I=ADELIAT 056 e resulta a mensagem:

?SINTAX ERRO

Cada laço FOR...NEXT ativo usa 16 bytes na memória.

NEXT

```
NEXT imed e prog
NEXT [avar]
NEXT avar  [{,avar}]
```

A instrução NEXT forma a parte inferior de um laço FOR...NEXT. Quando um NEXT é encontrado, o programa ou o ignora ou desvia para a instrução seguinte ao FOR correspondente, dependendo das condições explicadas na discussão do comando FOR.

/avars/ múltiplas devem ser especificadas na ordem correta de forma que os laços FOR...NEXT sejam incluídos uns dentro dos outros e não se "cruzem". /avars/ ordenadas incorretamente causarão a impressão da mensagem:

?"NEXT" SEM "FOR" ERRO

Um comando NEXT, no qual nenhum nome de variável é especificado, assume o valor do laço FOR mais recentemente inserido, que ainda está ativo. Se nenhum comando FOR com o mesmo nome de variável está ativo, ou se nenhum comando FOR de qualquer nome está ativo quando um NEXT sem nome é encontrado, é apresentada a mensagem:

?"NEXT" SEM "FOR" ERRO

Uma instrução NEXT sem /avar/ é executada mais rapidamente do que uma NEXT com /avar/.

No modo de execução imediata, o comando FOR e seu comando NEXT correspondente devem ser executados na mesma linha. Se um comando FOR de execução programada ainda está ativo, um comando NEXT de execução imediata pode causar um salto para o programa no lugar apropriado. Entretanto, se o comando FOR foi executado em execução imediata, um comando NEXT em uma linha diferente em execução imediata causará a mènsagem:

?SINTAX ERRO

a menos que não existam linhas interpostas e o ccmanco NEXT esteja só e sem nome, como no exemplo abaixo:

```
]FOR I = 1 TO 5 : PRINT I
1
]NEXT
2
]NEXT
3
]NEXT I
```

?SINTAX ERRO EM XXXX (XXXX é algum número de linha).

GOSUB

GOSUB imed e prog
GOSUB numlinha

O programa é desviado para a linha indicada por /numlinha/. Quando um comando RETURN é encontrado, o programa volta ao comando imediatamente seguinte ao GOSUB executado mais recentemente.

Cada vez que um comando GOSUB é executado, o endereço do comando seguinte é armazenado no topo de uma lista desses endereços, assim o programa pode encontrar depois o seu caminho de volta. Cada vez que um comando RETURN ou um POP é executado, o endereço no topo da lista de endereços de RETURN é removido.

Se o /numlinha/ indicado não corresponde a uma linha de programa existente, é dada a mensagem:

?LINHA INDEFINIDA ERRO EM numlinha

onde /numlinha/ indica a linha de programa que contém o comando GOSUB.

Se os comandos GOSUB são agrupados em mais de 25 niveis é exibida a mensagem:

?FALTA MEMORIA ERRO

Cada GOSUB ativo usa 6 bytes de memória.

RETURN

RETURN imed e prog
RETURN

Esta é uma instrução que desvia para o comando que segue imediatamente o mais recente comando GOSUB executado. O endereço do comando desviado é o mais alto da "lista" de RETURN (Veja GOSUB e POP).

Se um programa encontra comandos RETURN uma vez mais do que encontrou comandos GOSUB, é apresentada a mensagem:

?"RETURN" SEM "GOSUB" ERRO

POP

POP imed e prog
POP

Um POP tem o efeito de um RETURN porém não executa o

desvio. O próximo RETURN encontrado em lugar de desviar para aquele comando além do comando GOSUB mais recentemente executado, desviará para o segundo GOSUB mais recentemente executado.

Se o POP é executado antes que um GOSUB tenha sido encontrado, então é exibida a mensagem:

?"RETURN" SEM "GOSUB" ERRO

porque não há endereços de retorno na lista.

ON...GOTO E ON...GOSUB

ON...GOTO prog
ON...GOSUB prog
ON aexpr GOTO numlinha {[,numlinha]}
ON aexpr GOSUB numlinha {[,numlinha]}

ON...GOTO desvia para o número de linha especificado pelo /aexpr/ ésimo ítem na lista de numlinhas após o GOTO. ON...GOSUB funciona de forma semelhante, mas a execução de um GOSUB é preferível à GOTO.

Se /aexpr/ é Ø ou maior do que os itens listados, mas menor do que 256, então a execução do programa prossegue no próximo comando.

/aexpr 1/ deve estar no intervalo de Ø a 255 para evitar a mensagem:

?VALOR ILEGAL ERRO

ONERR GOTO

ONERR GOTO prog
ONERR GOTO numlinha

Quando ocorre um erro, ONERR GOTO pode ser usado para evitar uma mensagem de erro na tela e a interrupção do programa. O comando aciona um indicador que causa um salto incondicional

(se um erro ocorre posteriormente no programa) para a linha do programa indicada por numlinha. POKE 216,0 reposiciona o indicador de detecção de erro de modo que mensagens de erro normais serão exibidas.

O comando ONERR GOTO deve ser executado antes da ocorrência de um erro para evitar a interrupção do programa.

Quando ocorre um erro em um programa, o código para o tipo de erro é armazenado no endereço decimal de memória 222. Para ver qual erro foi encontrado, utilize a ordem PRINT PEEK (222).

Código	Mensagem de erro
0	"NEXT" SEM "FOR"
16	"SINTAX"
22	"RETURN" SEM "GOSUB"
42	FIM DE DATA
53	VALOR ILEGAL
69	S/ESPACO
77	FALTA MEMORIA
90	COMANDO INDEFINIDO
107	INDICE ILEGAL
120	REDIMENSIONA
133	DIVISAO POR ZERO
163	TIPO COMPAT
176	STRING LONGA
191	FORMULA COMPLEXA
224	FUNCAO INDEF.
254	RESPOSTA ILEGAL AO COMANDO INPUT
255	TENTATIVA INTERROMPIDA CTRL-C

Deve-se tomar cuidado ao manipular erros que ocorrem com laços FOR...NEXT ou entre GOSUB e RETURN, bem como endereçadores e listas de endereços para RETURN desarrumadas. A rotina de manipulação de erros deve recomeçar o laço voltando ao comando FOR ou GOSUB, e não para o comando NEXT ou RETURN. Após a manipulação de um erro, a volta para um NEXT ou um RETURN causará a mensagem apropriada:

```
?"NEXT" SEM "FOR" ERRO ou
?"RETURN" SEM "GOSUB" ERRO
```

Quando ONERR GOTO é usado com RESUME para manipular erros em um comando GET, o programa ficará "suspenso" se existirem dois erros consecutivos GET sem um GET bem sucedido interposto. Para evitar isso, use [RESET] CTRL-C [CR] . Se GOTO termina a rotina de manipulação de erro, tudo funciona bem.

Quando usado na modalidade TRACE ou em um programa contendo um comando PRINT, ONERR causa um salto para o Monitor após terem sido encontrados 43 erros. Onde estes erros são gerados por um comando INPUT, tudo funciona bem se é usado RESUME; mas se GOTO termina a rotina de manipulação de erros, o 37º erro de INPUT ocasiona um salto para o Monitor. Novamente [RESET] CTRL-C [CR] trará você de volta ao CCE BASIC.

Se qualquer dos problemas que acabamos de discutir puder prejudicar seu programa, execute um comando CALL para a seguinte sub-rotina em linguagem de máquina como parte da sua rotina de manipulação de erros:

- no Monitor, digite dados hexadecimais: 68 A8 68 46 DF 9A 48 98 48 60
- em CCE BASIC, digite dados decimais: 104 168 104 166 223 154 72 152 72 96.

Por exemplo, em CCE BASIC você poderia modificar os números decimais nas posições 768 até 777. Então você poderia usar CALL 768 na sua rotina de manipulação de erro.

RESUME

```
RESUME prog
RESUME
```

Quando usado no fim de uma rotina de manipulação de erro, faz com que o programa retorne à execução no começo do comando no qual o erro ocorreu.

Se o RESUME é encontrado antes que um erro ocorra, poderá ser apresentada a mensagem:

```
?SINTAX ERRO EM 65278
```

ou outros eventos estranhos podem acontecer. Usualmente, seu programa será interrompido, ou ficará "suspenso".

Se um erro ocorrer em uma rotina de manipulação de erro, então o uso de RESUME colocará o programa em um laço infinito. Use [RESET] CTRL-C [CR] . No modo imediato, isto pode deixar o sistema "suspenso", ou exibir uma mensagem ?SINTAX ERRO, ou pode começar a executar um programa existente ou até mesmo um programa desativado.

CAPíTULO 8

GRÁFICOS E CONTROLADORES DE JOGOS

TEXT

TEXT imed e prog
TEXT

Volta o vídeo ao modo usual (40 caracteres por linha, 24 linhas) a partir dos modos gráficos. Se apresentado no modo texto, TEXT é equivalente a VTAB 24.

Um comando tal como

175 TEXTIL = 127

causa a execução da palavra reservada TEXT, antes de aparecer a mensagem:

?SINTAX ERRO

Se a área de texto foi definida para qualquer outra modalidade que não seja tela cheia (veja o Apêndice J), TEXT reposiciona para tela cheia.

GR

GR imed e prog
GR

Este comando posiciona a tela para o modo gráfico de Baixa Resolução (40 por 40), reservando 4 linhas para texto na parte de baixo. A tela é limpada, e o cursor é movido para a janela de texto. Pode ser convertida para gráfico de tela cheia (40 por 48) depois da execução GR, com o comando

POKE - 16302,0

ou o comando equivalente

POKE 49234,0

Se GR sucede um comando POKE de tela cheia, a janela de texto é reposicionada.

Depois de um comando GR, COLOR= é posicionado em zero.

Se a palavra reservada GR é usada como os primeiros caracteres de um nome de variável, o GR pode ser executado antes que você obtenha a mensagem:

?SINTAX ERRO

Portanto, a execução do comando

GRAU = 5

deixa você, inesperadamente, com a tela escurecida.

Se GR é emitido enquanto que HGR está ativo, GR se comporta normalmente. Entretanto, se emitido enquanto HGR2 está ativo, GR limpa sua habitual tela cheia da memória, mas deixa ativa a página 2 de gráficos de Baixa-Resolução e texto. Para voltar ao modo normal, digite TEXT. Em programas, use TEXT antes de desviar de HGR2 para GR.

COLOR

COLOR = imed e prog
COLOR = aexpr

Define a cor a ser traçada na modalidade de gráficos de Baixa Resolução. Se /aexpr/ é um real, é convertido em um inteiro. O intervalo de valores para /aexpr/ varia de Ø a 255.

Os nomes de cores e seus números correspondentes são:

Ø preta	8 marrom
1 magenta	9 laranja
2 azul escura	1Ø cinza
3 púrpura	11 rosa
4 verde escura	12 verde clara
5 cinza	13 amarela
6 azul média	14 prata
7 azul clara	15 branca

COLOR é posicionada em zero pelo comando GR.

Para descobrir a cor de um ponto dado na tela, use o comando SCRN.

Quando usado no modo texto, COLOR é um fator na determinação de qual caráter é colocado na tela por uma instrução PLOT. Se usado enquanto se encontra na modalidade de gráficos de Alta Resolução, COLOR é ignorada.

PLOT

PLOT imed e prog
PLOT aexpr 1, aexpr 2

No modo gráfico de Baixa Resolução, este comando coloca um ponto com a coordenada X (aexpr 1) e coordenada Y (aexpr 2). A cor do ponto é determinada pelo comando COLOR= mais recentemente executado (COLOR = Ø se não foi previamente especificado).

/aexpr 1/ deve estar no intervalo de Ø a 39, e /aexpr 2/ deve estar no intervalo de Ø a 47 ou aparece a mensagem:

?VALOR ILEGAL ERRO

Uma tentativa de executar um traçado enquanto o sistema está no modo texto, ou no modo gráfico com janela de texto e /aexpr 2/ no intervalo de 4Ø a 47, resultará na colocação de um caráter onde o ponto colorido teria aparecido (um caráter ocupa o espaço de dois pontos nos gráficos de Baixa Resolução, empilhados verticalmente).

O comando não tem efeito visível quando usado no modo gráfico de Alta Resolução 2(HGR2), mesmo que precedido por um comando GR, pois a tela não está "olhando" a porção de memória de gráficos de Baixa Resolução (página 1).

A origem (Ø,Ø) para todos os gráficos está no canto superior esquerdo da tela.

HLIN

HLIN imed e prog
HLIN aexpr 1, aexpr 2 AT aexpr 3

Usada no modo gráfico de Baixa Resolução, HLIN desenha uma linha a partir de (/aexpr 1/, /aexpr 3/) para (/aexpr 2/, /aexpr 3/). A cor é determinada pelo comando COLOR= mais recentemente executado.

/aexpr 1/ e /aexpr 2/ devem estar no intervalo de 0 a 39, e /aexpr 3/ deve estar no intervalo entre 0 e 47, ou aparece a mensagem:

?VALOR ILEGAL ERRO

/aexpr 1/ pode ser maior, igual ou menor a /aexpr 2/.

Se HLIN é usado quando o sistema está no modo texto, ou no modo gráfico com janela de texto com /aexpr 3/ no intervalo de 40 a 47, uma linha de caracteres será colocada onde a linha de pontos do gráfico deveria ter sido traçada (um caráter ocupa o espaço de dois pontos de Baixa Resolução, empilhados verticalmente).

Essa instrução não tem efeito visível quando usado no modo gráfico de Alta Resolução.

VLIN

VLIN imed e prog
VLIN aexpr 1, aexpr 2 AT aexpr 3

No modo gráfico de Baixa Resolução, esta instrução traça uma linha vertical a partir de (/aexpr 1/,/aexpr 3/) para (/aexpr 2/, /aexpr 3/). A cor é determinada pelo comando COLOR mais recentemente executado.

/aexpr 1/ e /aexpr 2/ devem estar no intervalo de Ø a 47, /aexpr 3/ deve estar no intervalo de Ø a 39, ou é exibida a mensagem:

?VALOR ILEGAL ERRO

/aexpr 1/ pode ser maior, igual ou menor do que /aexpr 2/.

Se o sistema está no modo texto, quando HLIN é usado, ou no modo gráfico com janela de texto com /aexpr 2/ no intervalo de 4Ø a 47, a porção da linha na área de texto aparecerá com uma linha de caracteres, colocada onde os pontos gráficos deveriam ter sido traçados.

Este comando não tem efeitos visíveis quando usado no modo gráfico de Alta Resolução.

SCRN

SCRN imed e prog
SCRN (aexpr 1, aexpr 2)

No modo gráfico de Baixa Resolução, a função SCRN devolve o código de cor do ponto cuja coordenada X é /aexpr 1/ e cuja coordenada Y é /aexpr 2/.

Embora os gráficos de Baixa Resolução tracem pontos nas posições da tela (X,Y) onde X está no intervalo de Ø a 39 e Y está no intervalo de Ø a 47, a função SCRN aceita ambos os valores X e Y no intervalo de Ø a 47. Todavia, se SCRN é usado com um valor X (/aexpr 1/) no intervalo de 4Ø a 47, o número devolvido dá a cor no ponto cuja coordenada X é (/aexpr/-4Ø) e cuja coordenada Y é (/aexpr 2/+16).

Se (/aexpr 2/+16) está no intervalo de 39 a 47, na modalidade gráfica com janela de texto, o número devolvido por SCRN é relacionado ao caráter do texto naquela posição na área de texto abaixo da porção de gráficos da tela. Se (/aexpr 2/+16) está no intervalo de 48 a 63, SCRN devolve um número não relacionado a coisa alguma na tela.

Na modalidade de texto, SCRN devolve números do caráter na posição, no intervalo de Ø a 15, cujo valor é:

- os quatro bits de mais alta ordem, se /aexpr 2/ é impar, ou
- os quatro bits de mais baixa ordem, se /aexpr 2/ é par

No modo gráfico de Alta Resolução, SCRN continua a "olhar para" a área de gráficos de Baixa Resolução, e o número que SCRN devolve não é relacionado à exibição de Alta Resolução.

SCRN é analisado somente se o próximo caráter diferente de espaço é um parêntese esquerdo.

HGR

HGR imed e prog
HGR

Ativa o modo gráfico de Alta Resolução (280 por 160) para a tela, deixando 4 linhas para texto na parte inferior. A tela é limpada e a página 1 da memória (8K - 16K) é exibida. O uso do comando HGR deixa a janela de texto em tela cheia, mas somente as 4 linhas inferiores são visíveis abaixo dos gráficos. O cursor ainda estará na janela de texto, mas pode não ser visível a menos que ele seja movido para uma das quatro linhas inferiores.

A tela pode ser convertida para gráficos de tela cheia (280 por 192) depois de executar HGR com o comando

POKE - 16302,0

ou o uso de seu equivalente:

POKE 49234,0

Se HGR segue um dos comandos POKE acima, a janela de texto é reposicionada.

Se a palavra reservada HGR é usada como os primeiros caracteres de um nome de variável, o comando HGR pode ser executado antes que apareça a mensagem:

?SINTAX ERRO

Assim, a execução do comando

HGRIP = 4

resulta numa inesperada passagem ao modo gráfico de Alta Resolução, a qual pode apagar o seu programa.

Um programa muito longo, o qual se estende acima da posição de memória 8192 pode ser parcialmente apagado quando você executa HGR, ou ele pode "escrever" sobre sua área de gráficos de Alta Resolução página 1. Em particular, dados tipo cadeia, são armazenados no topo da memória. Posicione HIMEM: 8192 para proteger o seu programa e a página 1 dos gráficos de Alta Resolução.

HGR2

HGR2 imed e prog
HGR2

Este comando posiciona o modo gráfico de Alta Resolução, em tela cheia (280 x 192).

A tela é limpada e a página 2 da memória (16K - 24K) é exibida. A memória da tela de texto não é afetada. O uso de HGR2 em lugar de HGR maximiza o espaço de memória disponível para programas.

Se a palavra reservada HGR2 é usada como os primeiros caracteres num nome de variável, a HGR2 pode ser executada antes que seja dada a mensagem:

?SINTAX ERRO

O comando

POKE - 16301,0

permite em qualquer modalidade gráfica o uso da janela de texto, entretanto quando emitido depois de HGR2, as 4 linhas do texto são tomadas da página 2 do texto, que não é facilmente acessível ao usuário.

HCOLOR=

```
HCOLOR= imed e prog
HCOLOR= aexpr
```

Define a cor de gráficos de Alta Resolução para aquela especificada pelo valor de HCOLOR=, que deve estar num intervalo de Ø a 7, inclusive. Os nomes das cores e seus respectivos valores são:

Ø preta	4 preta
1 verde (depende da TV)	5 (depende da TV)
2 azul (depende da TV)	6 (depende da TV)
3 branca	7 branca

Um ponto de Alta Resolução traçado em HCOLOR = 3 (branco) será azul se a coordenada X do ponto é par, verde se a coordenada X é impar e branca somente se ambas (X,Y) e (X+1,Y) são traçadas. Isto é devido ao modo como as TVs domésticas funcionam.

HCOLOR= não é alterado por HGR, HGR2 ou RUN. Até que o primeiro comando HCOLOR seja executado, a cor usada em gráficos de Alta Resolução é indeterminada.

Se usada enquanto estiver no modo gráfico de Baixa Resolução, HCOLOR= não afeta a cor que está sendo exibida.

HPLOT

```
HPLOT imed e prog
HPLOT aexpr 1, aexpr 2
HPLOT TO aexpr 3, aexpr 4
HPLOT aexpr 1, aexpr 2 TO aexpr 3, aexpr 4 [{TO
aexpr, aexpr}]
```

Na primeira opção, HPLOT apresenta um ponto de Alta Resolução, cuja coordenada X é /aexpr 1/ e cuja coordenada Y é /aexpr 2/. A cor do ponto é determinada pelo comando HCOLOR mais recentemente executado. O valor de HCOLOR= é indeterminado se não previamente especificado.

A segunda opção faz com que uma linha seja traçada a partir do último ponto para (/aexpr 3, /aexpr 4/). A cor desta linha é determinada pela cor do último ponto traçado, mesmo que o valor de HCOLOR tenha sido alterado desde o traçado anterior. Se nenhum ponto anterior foi traçado, nenhuma linha é desenhada.

Se a terceira opção é usada, uma linha a partir de (/aexpr 1/,/aexpr 2/) para (/aexpr 3/,/aexpr 4/) é traçada usando a cor especificada pelo mais recente comando HCOLOR. A linha traçada pode ser estendida na mesma instrução, quase indefinidamente. (sujeita aos limites da tela, e ao limite para as instruções de 239 caracteres), pela extensão da instrução com TO aexpr 5, aexpr 6 TO aexpr 7, aexpr 8 e assim por diante. Um único comando

HPLOT Ø,Ø TO 279, Ø TO 279, 159 TO Ø, 159 TO Ø,Ø

pode traçar um retângulo, contornando os 4 lados da tela de Alta Resolução.

HPLOT deve ser precedido de HGR ou HGR2 para evitar danos a porções da memória, incluindo o seu programa e variáveis.

/aexpr 1/ e /aexpr 3/ devem estar no intervalo de Ø a 279.
/aexpr 2/ e /aexpr 4/ devem estar no intervalo de Ø a 191.
/aexpr 1/ pode ser maior, igual ou menor que /aexpr 3/.
/aexpr 2/ pode ser maior, igual ou menor que /aexpr 4/.

Uma tentativa para traçar um ponto cujas coordenadas excedam estes limites causará a mensagem:

?VALOR ILEGAL ERRO

Se a tela está em modo gráfico de Alta Resolução com janela de texto, então tentar traçar pontos com a coordenada Y no intervalo 16Ø a 191, não terá efeito visível.

PDL

PDL imed e prog
PDL (aexpr)

Esta função devolve o valor corrente, de Ø a 255, do controlador de jogo especificado por /aexpr/, se /aexpr/ está no intervalo de Ø a 3. O controlador de jogo é uma resistência variável de Ø a 15Ø ohms.

Se dois controladores de jogos são lidos em instruções PDL consecutivas, a leitura a partir do segundo controlador de jogos pode ser afetada a partir do primeiro. Para obter leituras mais precisas, permita várias linhas de programa, ou coloque um pequeno laço de espera (FOR I = 1 TO 10: NEXT I) entre instruções PDL.

Se /aexpr/ é negativa ou maior que 255 é dada a mensagem:

?VALOR ILEGAL ERRO

Se /aexpr/ está no intervalo de 4 a 255, a função PDL devolve um número um tanto imprevisível de Ø a 255, e pode causar vários efeitos colaterais, alguns dos quais podem perturbar a execução do programa. Por exemplo, se /aexpr/ está no intervalo de 204 a 219, o uso da função PDL é frequentemente e um tanto aleatoriamente acompanhado por um "click" do alto-falante do computador.

Se N está no intervalo 236 a 239, PDL (N) pode resultar em um

POKE - 16540 + N, Ø

de modo que PDL (236) pode posicionar o modo gráfico, PDL (237) pode posicionar modo texto, etc... (veja apêndice J).

Além de ler a colocação das 4 variáveis de controladores de jogos, usando PDL, o CCE BASIC pode ler o estado de 3 botões de jogos (chaves: ligado-desligado) usando vários comandos PEEK e pode ligar e desligar 4 leitores de jogos (chaves TTL), usando vários comandos POKE (veja apêndice J).

CAPíTULO 9

FORMAS EM ALTA RESOLUÇÃO

COMO CRIAR UMA TABELA DE FORMA

O CCE BASIC possui cinco comandos especiais que permitem a você manipular formas em gráficos de Alta Resolução: DRAW, XDRAW, ROT, SCALE e SHLOAD. Antes que estes comandos possam ser usados, uma forma deve ser definida por uma "definição de forma". Esta definição consiste de uma sequência de vetores de traçado, que são armazenados em uma série de bytes na memória do computador. Uma ou mais dessas "definições de forma", com seus índices, fazem uma "tabela de forma" a qual pode ser criada a partir do teclado e salva em disco ou fita cassete para uso futuro.

Cada byte em uma "definição de forma" é dividido em três seções, e cada seção pode especificar um "vetor de traçado". O "vetor de traçado" define se traça ou não um ponto, e também a direção do deslocamento (para cima, para baixo, para a esquerda, ou para a direita). DRAW e XDRAW passam através de cada byte "definição de forma", seção por seção, a partir do primeiro ao último byte. Quando um byte que contém somente zeros é atingido, a definição está completa.

Aqui está como as três seções A, B e C estão arranjadas em um dos bytes de uma definição de forma:

Seção	C		B		A			
Número do Bit	7	6	5	4	3	2	1	0
Especifica	D	D	P	D	D	P	D	D

Cada par de bits DD especifica a direção do movimento, e cada bit P especifica se traça ou não um ponto antes de mover, conforme segue:

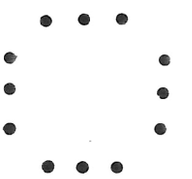
```
Se DD = ØØ move para cima          Se P = Ø não traça
   DD = Ø1 move para a direita        P = 1 traça
   DD = 1Ø move para baixo
   DD = 11 move para esquerda
```

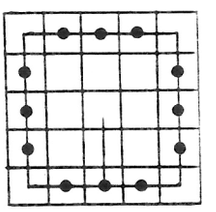
Observe que a seção C (os dois bits de mais alta ordem), não tem um campo P (por "default", P=Ø), assim a seção C somente pode especificar um movimento sem traçado.

Cada byte pode representar até três vetores de traçado, um na seção A, um na seção B, e um terceiro (um movimento somente) na seção C.

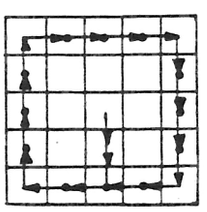
DRAW e XDRAW processam as seções da direita para a esquerda (bit de mais baixa ordem para bit de mais alta ordem: seção A, em seguida B, e finalmente C). Em qualquer seção se as demais seções do byte contém somente zeros, ela deve ser ignorada. Desse modo, o byte não pode terminar com um movimento na seção C de ØØ (um movimento para cima, sem traçado) porque aquela seção, contendo somente zeros, será ignorada. Similarmente, se a seção C for ØØ (ignorada), então a seção B não pode ser um movimento de ØØØ, posto que também será ignorada e, um movimento de ØØØ na seção A, terminará sua definição de forma a menos que exista um bit 1 em algum lugar na seção B ou C.

Suponha que você queira desenhar um feitio como este:





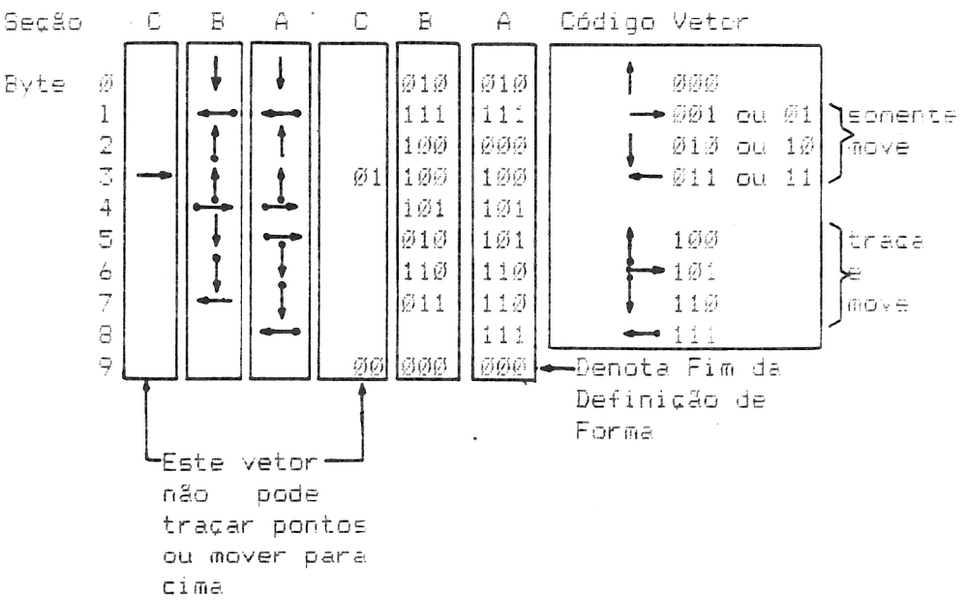
Primeiro, desenhe-o em papel quadriculado, um ponto por quadrado. Em seguida decida onde começar o desenho da forma. Comecemos esta pelo centro. A seguir, desenhe um caminho através de cada ponto no feitio, usando somente ângulos de 9Ø graus nas mudanças de direção:



A seguir, redesenhe a forma como uma série de vetores de traçado, cada um movendo-se uma posição para cima, para baixo, para a direita, ou para a esquerda, e destaque os vetores que traçam um ponto antes de movimentar (uma bolinha marca os vetores que traçam pontos).

Agora "desenrole" aqueles vetores e escreva-os em uma linha reta:

A seguir, desenhe uma tabela como na figura abaixo:



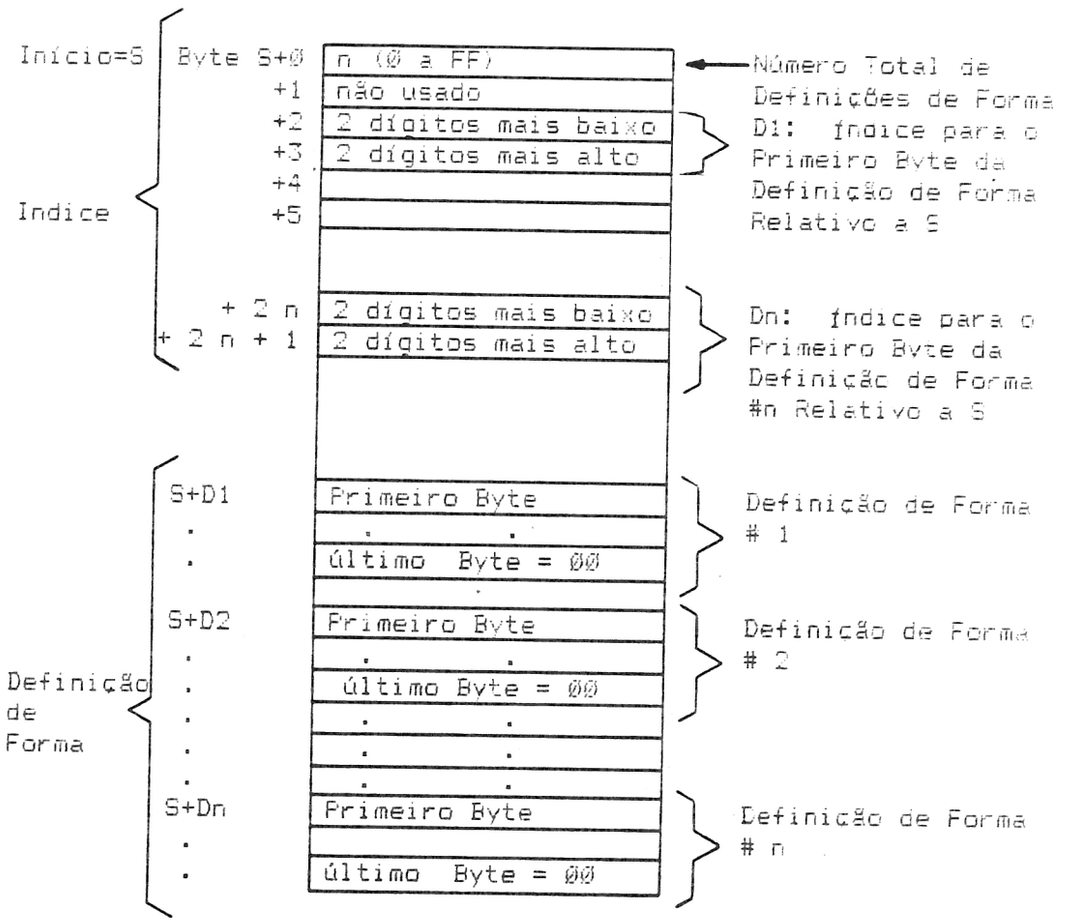
Para cada vetor na linha, determine o código do bit e coloque-o na próxima seção disponível na tabela. Se o código não couber (por exemplo, o vetor na seção C não pode traçar) ou é um ØØ (ou ØØØ) no fim de um byte, então salte aquela seção e vá para a próxima. Quando você tiver terminado de codificar todos os seus vetores, confira o seu trabalho para certificar-se de que ele está correto.

Agora faça outra tabela, conforme indicado a seguir, e recopie os códigos de vetores a partir da primeira tabela. Recodifique a informação de vetores em uma série de bytes hexadecimais, usando os valores apresentados nas tabelas de códigos hexadecimais.

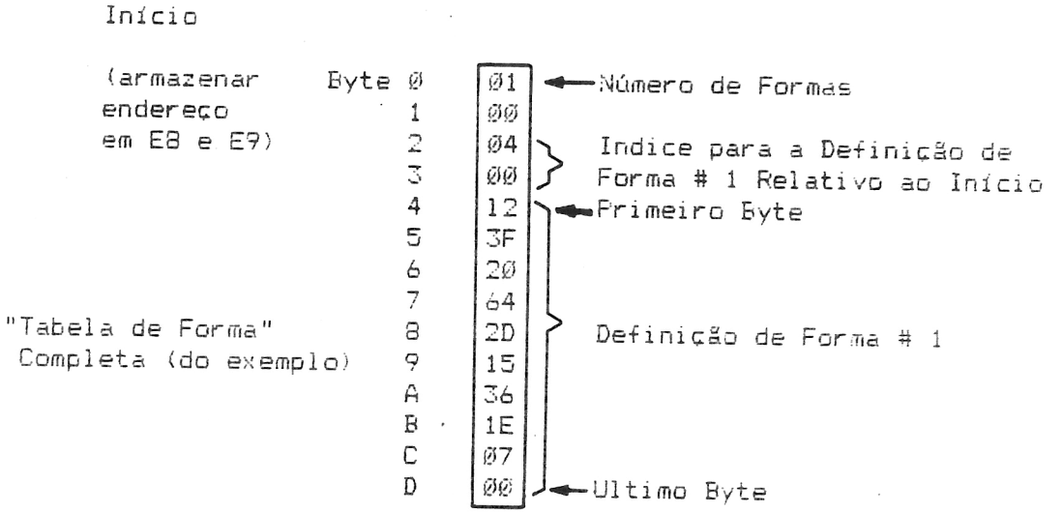
Seção:	C	B	A	Bytes Recodificados em Hexa		códigos Binário	Hexa
Byte Ø	ØØØ1	ØØ1Ø	=	12		ØØØØ	Ø
1	ØØ11	1111	=	3F		ØØØ1	1
2	ØØ1Ø	ØØØØ	=	2Ø		ØØ1Ø	2
3	Ø11Ø	Ø1ØØ	=	64		ØØ11	3
4	ØØ1Ø	11Ø1	=	2D		Ø1ØØ	4
5	ØØØ1	Ø1Ø1	=	15		Ø1Ø1	5
6	ØØ11	Ø11Ø	=	36		Ø11Ø	6
7	ØØØ1	111Ø	=	1E		Ø111	7
8	ØØØØ	Ø111	=	Ø7		1ØØØ	8
9	ØØØØ	ØØØØ	=	ØØ←Denota		1ØØ1	9
				Fim da		1Ø1Ø	A
Hexa:	DíGITO 1	DíGITO 2		Definição		1Ø11	B
				de Forma		11ØØ	C
						11Ø1	D
						111Ø	E
						1111	F

Tabela de
Códigos Hexadecimais

A série de bytes hexadecimais que você obteve acima é a "definição de forma". Existe ainda mais uma pequena informação que você deve fornecer antes de ter a "tabela de forma" completa. A "tabela de forma" completa, com seu índice, é mostrado a seguir. Para este exemplo, seu índice é simples: existe somente uma "definição de forma". A posição dessa tabela, cujo endereço chamamos S, deve conter o número de "definições de forma" (entre Ø e 255) em hexadecimal. Neste caso, aquele número é apenas um. Colocaremos nossa "definição de forma" imediatamente abaixo do índice, por simplicidade, o que significa, neste caso, a "definição de forma" começa no byte S + 4. O endereço da "definição de forma" # 1, relativo a S, é 4 (ØØ Ø4, em hexadecimal). Conseqüentemente, o byte índice S + 2 deve conter o valor Ø4 e o byte índice S + 3 deve conter o valor ØØ. A "tabela de forma" completa para este exemplo, é mostrada na última figura desta seção.



"Tabela de Forma" Completa (genérica)



Agora você está pronto para digitar a "tabela de forma" na memória do computador. Primeiro escolha um endereço de início. Para este exemplo, usaremos o endereço $1DFC (hexadecimal). (Observação: este endereço deve ser menor do que o endereço mais alto disponível em seu sistema, e não em uma área que será limpada quando você usar HGR ou HGR2. A posição $1DFC é imediatamente abaixo da página 1 dos gráficos de Alta Resolução (usada por HGR). Digite CALL-151 para entrar no programa Monitor, e digite o endereço de início para a sua tabela de forma: 1DFC.

Se você apertar a tecla [CR] agora, o computador mostrará o endereço e o conteúdo. É desse modo que você examina um endereço para ver se colocou o valor correto. Se ao invés disso você digitar dois-pontos (:) seguido por um número de 2 dígitos hexadecimais, aquele número será armazenado no endereço especificado quando você aperta a tecla [CR]. Tente isto:

```
* 1DFC:Ø1 [CR]
* 1DFC [CR]
* 1DFC-Ø1
```

O computador agora diz que o valor $Ø1 (hexadecimal) está armazenado na posição cujo endereço é $1DFC. Para armazenar mais números de dois dígitos hexadecimais em bytes na memória, abra apenas o primeiro endereço, e em seguida digite os números, separados por espaços:

```
* 1DFC:Ø1 ØØ Ø4 ØØ 12 3F 2Ø 64 2D 15 36 1E Ø7 ØØ [CR]
```

Você acaba de digitar sua primeira tabela de formas completa. Para conferir a informação em sua tabela de formas, você pode examinar cada byte separadamente ou simplesmente aperte repetidamente a tecla [CR] até que todos os bytes de interesse (e um pouco a mais, provavelmente) tenham sidos exibidos:

```
*1DFC [CR]
1DFC-Ø1
* [CR]
ØØ Ø4 ØØ
* [CR]
1EØØ- 123F 2Ø 64 2D 15 36 1E
* [CR]
1EØ8- Ø7 ØØ DF 1E 23 ØØ ØØ FF
```

Se sua tabela de formas parece correta, tudo que resta é armazenar o endereço inicial da tabela de formas onde o CCE BASIC possa achá-lo (isto é feito automaticamente quando você usa SHLOAD para obter uma tabela a partir de fita cassete). O CCE BASIC procura os 4 dígitos hexadecimais do endereço inicial da tabela nas posições $E8 (dois dígitos mais baixos) e $E9 (dois dígitos mais altos). Para o nosso endereço inicial da tabela de $1DFC, isto poderia ser feito por:

*E8:FC 1D [CR]

(note que a ordem de inserção dos bytes é inversa à original).

Para proteger sua tabela de forma de ser apagada acidentalmente pelo seu programa CCE BASIC, poderia também ser uma boa idéia posicionar HIMEM: (nas posições 73 e 74) para o endereço inicial da tabela:

* 73:FC 1D

Isto também é feito automaticamente quando você usa SHLOAD para obter a tabela a partir de fita cassete.

SALVANDO UMA TABELA DE FORMA

Para salvar sua tabela de forma em fita, você precisa saber três coisas:

1- o endereço inicial da tabela ($1DFC, em nosso exemplo)
2- o último endereço da tabela ($1E09, em nosso exemplo)
3- a diferença entre (2) e (1) ($000D, em nosso exemplo)

A diferença entre o último e o primeiro endereço da tabela, deve ser armazenada nas posições $0 (dois dígitos mais baixos) e$1 (dois dígitos mais altos):

*0: 0D 00 [CR]

Agora você pode "gravar" (armazenar em cassete) primeiro o comprimento da tabela que está armazenado nas posições $0 a $1, em seguida a tabela de forma propriamente, que está armazenada nas posições Endereço Inicial até Último Endereço:

0.1W 1DFC.1E09W

Não aperte a tecla [CR] até que você tenha colocado um cassete em seu gravador; rebobinado e começado a gravar. Agora aperte a tecla [CR] .

Para usar a fita, rebobine-a, comece a tocá-la (aperte a tecla PLAY), e digite SHLOAD [CR] . Você ouvirá um "bip" quando o comprimento da tabela foi lido com sucesso, e outro "bip" no final da leitura da tabela.

USANDO UMA TABELA DE FORMA

Agora você está pronto para escrever um programa CCE BASIC usando os comandos para tratamento de tabelas de forma DRAW, XDRAW, ROT e SCALE.

Aqui está um exemplo de programa CCE BASIC que apresentará nossa forma definida, fará a rotação do mesmo em 16 graus, e em seguida repetirá a operação. Cada repetição maior do que a anterior.

```
10 HGR
20 HCOLOR=3
30 FOR R = 1 TO 50
40 ROT = R
50 SCALE = R
60 DRAW 1 AT 139,79
70 NEXT R
```

Para ver um único "quadrado", adicione a linha 65 END.

Para dar uma pausa e apagar cada quadrado depois que foi desenhado adicione estas linhas:

```
63 FOR I = 0 TO 1000 : NEXT I
65 XDRAW 1 AT 139,79
```

DRAW

```
DRAW imed e prog
DRAW aexpr1 AT aexpr2, aexpr3
DRAW aexpr1
```

DRAW com a primeira opção desenha uma forma em modo de Alta Resolução começando no ponto cuja coordenada X é /aexpr2/ e cuja coordenada Y é /aexpr3/. A forma desenhada é a /aexpr1/ ésima definição de forma na tabela de formas previamente carregada, usando o comando SHLOAD (ou uma tabela de formas pode ser digitada na memória do computador em código hexadecimal, usando o programa Monitor).

/aexpr1/ deve estar no intervalo Ø até n, onde n é o número (de Ø a 255) das definições de formas dados no byte da tabela de forma. /aexpr2/ deve estar no intervalo de Ø até 278. /aexpr3/ no intervalo de Ø a 191. Se qualquer um desses intervalos for excedido, será exibida a mensagem ?VALOR ILEGAL ERRO.

A cor, rotação e escala da forma a ser desenhada, devem ter sido especificadas antes que DRAW seja executada.

A segunda opção é similar à primeira, mas desenha a forma especificada começando no último ponto traçado pelo mais recente comando HPLOT, DRAW, ou XDRAW executado.

NOTA: se emitida quando não houver tabela de forma no computador, poderá causar a suspensão do sistema. Para recuperar, use RESET ou CTRL-C CR . Poderá também desenhar formas aleatórias por toda a área do gráfico de Alta Resolução, possivelmente destruindo seu programa, mesmo que você não esteja no modo gráfico.

XDRAW

```
XDRAW imed e prog
XDRAW aexpr1 (AT aexpr2, aexpr3)
```

Este comando faz o mesmo que DRAW, exceto que a cor usada para desenhar a forma e o complemento da cor já existente em

cada ponto traçado. Estes pares de cores são complementos:

Preto e Branco
Azul e Verde

A finalidade de XDRAW é fornecer um modo fácil para apagar: Se você desenha uma forma com XDRAW, você poderá suprimi-la da tela sem apagar o fundo da figura.

Veja a NOTA em DRAW.

ROT=

ROT= imed e prog
ROT= aexpr

Provoca a rotação angular para a forma a ser desenhada por DRAW ou XDRAW. O valor da rotação é especificado por /aexpr/, o qual deve estar entre Ø e 255.

ROT=Ø faz com que a forma a ser desenhada seja orientada conforme foi definida, ROT=16 causará a rotação da forma a ser desenhada em 9Ø graus no sentido horário, ROT=32 causará a rotação da forma a ser desenhada em 18Ø graus no sentido horário, etc. O processo se repete começando em ROT=64. Para SCALE=1, somente 4 valores de rotação são reconhecidos (Ø,16,32,48); para SCALE=2, 8 valores de rotação são reconhecidos etc. Valores de rotação não reconhecidos causarão o desenho da forma com a orientação de valor igual ao valor reconhecido imediatamente menor (usualmente).

ROT é analisada sintaticamente como uma palavra reservada somente se o próximo caráter diferente de espaço for o sinal de igual (=).

SCALE=

SCALE= imed e prog
SCALE= aexpr

Estipula o tamanho (escala) da forma a ser desenhada por DRAW ou XDRAW para fator de 1 (reprodução ponto por ponto da definição da forma) a 255 (cada vetor ampliado 255 vezes) conforme especificado por /aexpr/.

NOTAS: SCALE=Ø é o tamanho máximo e não um simples ponto. SCALE é analisada sintaticamente como uma palavra reservada somente se o próximo caráter de espaço for o sinal de igual (=).

SHLOAD

SHLOAD imed e prog
SHLOAD

Carrega uma tabela de formas de uma fita cassete. (A tabela de formas é carregada exatamente abaixo do HIMEM:) O HIMEM: é posicionado imediatamente abaixo da tabela de formas para protegê-la. O endereço de início da tabela de formas é dado para as rotinas de desenho do CCE BASIC automaticamente. Se uma segunda tabela é carregada, substituindo a primeira tabela, o HIMEM deve ser reposicionado antes da carga para evitar desperdício de memória. As fitas de tabelas de formas são preparadas usando as instruções no início deste capítulo.

Somente RESET pode interromper SHLOAD. Se a palavra reservada SHLOAD começa um nome de variável, o comando da palavra reservada pode ser executado antes de ser dada a mensagem:

?SINTAX ERRO

A declaração SHLOADER=59 suspende o sistema, enquanto o CCE BASIC espera indefinidamente por uma tabela a partir do gravador de fita cassete. Use RESET CTRL-C para recuperar o controle do computador.

CAPÍTULO 10

ALGUMAS FUNÇÕES MATEMÁTICAS:

147- FUNÇÕES EMBUTIDAS: SIN (seno), COS (co-seno)
TAN (tangente), ATN (arco-tangente),
INT (inteiro), RND (randômica),
SGN (sinal), ABS (absoluto),
SQR (raiz quadrada), EXP (potência),
LOG (logaritmo).

148- FUNÇÕES DERIVADAS

FUNÇÕES EMBUTIDAS

Todas as funções embutidas podem ser aplicadas onde quer que uma expressão do mesmo tipo possa ser usada, quer na modalidade de execução imediata, quer na programada. Aqui estão descrições breves de algumas funções aritméticas do CCE BASIC. Outras funções são descritas em seções que tratam de instruções similares.

SIN (aexpr) - Devolve o seno de /aexpr/ radianos.

COS (aexpr) - Devolve o co-seno de /aexpr/ radianos.

TAN (aexpr) - Devolve a tangente de /aexpr/radianos.

ATN (aexpr) - Devolve a arco-tangente, em radianos, de /aexpr/. O ângulo devolvido está no intervalo de - /2 a + /2 radianos.

INT (aexpr) - Devolve o maior número inteiro menor ou igual a /aexpr/.

RND (aexpr) - Devolve um número real aleatório maior ou igual a 0 e menor que 1.

Se /aexpr/ é maior que zero, RND (aexpr) gera um novo número aleatório cada vez em que é usada.

Se /aexpr/ é menor que zero, RND (aexpr) gera o mesmo número aleatório a cada vez em que é usada com a mesma /aexpr/, como se a partir de uma tabela permanente de números aleatórios, construída dentro do computador. Quando um argumento negativo particular é usado para gerar um número aleatório, então números aleatórios subsequentes gerados com argumentos positivos, seguirão a mesma sequência a cada vez. Uma sequência aleatória diferente é inicializada por cada argumento negativo diferente. A principal razão para usar um argumento negativo para RND é para inicializar uma sequência de números aleatórios respectiva. Isto é particularmente útil na depuração de programas que usem RND.

Sendo /aexpr/ igual a zero RND (aexpr) devolve o mais recente número aleatório gerado anteriormente (CLEAR ou NEW não afetam este procedimento). Às vezes, isto é mais fácil do que atribuir o último número aleatório a uma variável, a fim de salvá-lo.

SGN (aexpr - Devolve "-1" se /aexpr/ <Ø, devolve "Ø" se /aExpr/=Ø e devolve "1" se /aexpr/>Ø.

ABS (aexpr) - Devolve o valor absoluto de /aexpr/, isto é /aexpr/ se /aexpr/ >=Ø e -/aexpr/ se /aexpr/<Ø.

SQR (aexpr Devolve a raiz quadrada positiva. Esta é uma implementação especial que executa mais rapidamente do que elevar um número à potência 1/2 ou ".5".

EXP (aexpr - Eleva "e" (com 6 casas decimais, e=2.718289) à potência indicada, /aexpr/, onde "e" é a base dos logaritmos naturais.

LOG (aexpr) - Devolve o logaritmo natural de /aexpr/.

FUNÇÕES DERIVADAS

As funções seguintes, embora não intrinsecas ao CCE BASIC, podem ser calculadas usando-se as funções apresentadas na seção anterior, e podem ser implementadas facilmente pelo uso da função DEF FN.

```
SECANTE:
 SEC(X) = 1/COS (X)

CO-SECANTE:
 CSC (X) = 1/SEN (X)

CO-TANGENTE:
 COT (X) = 1/TAN (X)

SENO INVERSO:
 ARCSIN (X) = ATN (X/SQR (-X*X+1))

CO-SENO INVERSO:
 ARCCOS (X) = -ATN (X/SQR (X*X+1)) + 1.57Ø8

SECANTE INVERSA:
 ARCSEC (X) = ATN (SQR(X*X-1)) + (SGN(X)-1) * 1.57Ø8

CO-SECANTE INVERSA:
 ARCCSC (X) = ATN (1/SQR(X*X-1)) + (SGN(X)1)* 1.57Ø8
```

```
CO-TANGENTE INVERSA:
 ARCCOT (X) = -ATN (X) + 1.57Ø8

SENO  HIPERBÓLICO :
 SINH (X) = (EXP(X) - EXP(-X))/2

CO-SENO  HIPERBÓLICO :
 COSH (X) = (EXP(X) + EXP(-X))/2

TANGENTE  HIPERBÓLICA :
 TANH (X) = -EXP(-X)/(EXP(X) + EXP(-X)) * 2+1

SECANTE  HIPERBÓLICA :
 SECH (X) = 2/(EXP(X) + EXP (-X))

CO-SECANTE  HIPERBÓLICA :
 CSCH (X) = 2/(EXP(X) - EXP(-X))4

CO-TANGENTE  HIPERBÓLICA :
 COTH (X) = EXP(-X)/EXP(X) - EXP(-X)) * 2+1

SENO  HIPERBÓLICO  INVERSO:
 ARCSINH (X) = LOG (X+SQR (X*X+1))

CO-SENO  HIPERBÓLICO  INVERSO:
 ARCOSH (X) = LOG (X+SQR (X*X-1))

TANGENTE  HIPERBÓLICA  INVERSA:
 ARCTANH (X) = LOG ((1+X)/(1-X))/2

SECANTE  HIPERBÓLICA  INVERSA:
 ARCSECH (X) = LOG ((SQR (-X*X+1)+1)/X

CO-SECANTE  HIPERBÓLICA  INVERSA:
 ARCCSCH (X) = LOG (SGN (X) * SQR ((X*X+1)+1)/X

CO-SECANTE  HIPERBÓLICA  INVERSA:
 ARCCOTH (X) = LOG ((X+1)/(X-1))/2

A MOD B
 MOD(A) = INT ((A/B - INT (A/B))*B+.Ø5) * SGN (A/B)
```

APÊNDICES

APÊNDICE A:

ATIVANDO O CCE BASIC OU INTEGER

A CCE oferece duas versões da linguagem de programação BASIC: o CCE BASIC e o CCE INTEGER. O CCE INTEGER é um BASIC muito rápido adequado para muitas aplicações, especialmente em educação, jogos e gráficos. O CCE BASIC é mais adequado a aplicações comerciais e científicas.

NOTAS: neste manual, [RESET] significa apertar a tecla marcada "RESET"; [CR] significa apertar a tecla marcada "CR" e CTRL-B significa digitar B enquanto segurando a tecla marcada com "CTRL".

Uma das funções do caráter de prontidão, além de indicar que está pronto para você entrar com comandos é identificar em qual linguagem o computador está programado para responder naquele momento. Por exemplo, até agora você viu três caracteres de prontidão:

* para o programa Monitor (quando você emite o comando CALL-151)
> para INTEGER BASIC
] para BASIC com ponto-flutuante (CCE BASIC).

ATIVANDO O CCE BASIC

Existem três situações em que você deverá tomar diferentes atitudes para utilizar o CCE BASIC:

- quando não houver nenhum drive conectado ao seu EXATO ;
- quando houver pelo menos um drive conectado ao seu EXATO porém você não quiser utilizar disco algum;
- se você quiser utilizar o CCE BASIC com o CDOS carregado no computador.

No primeiro caso, quando não houver nenhum drive conectado ao seu EXATO, o simples fato de ligar o computador fará com que o CCE BASIC passe a estar disponível, o que pode ser comprovado através da apresentação do caráter de prontidão desse sistema (]) na tela.

No segundo caso, quando houver um ou mais drives, sem discos, conectados ao seu computador, ao ligar o EXATO, ele procurará "ler" dados do drive "default" (geralmente o drive 1 ligado ao slot 6). Digite então a tecla [RESET].

Quando você quiser carregar o CDOS no EXATO, ligue o computador, introduza o disco SISTEMA MESTRE de forma correta (com a abertura de leitura do disco para dentro do drive e a etiqueta para cima) e feche a porta do drive. Se o computador já tiver sido ligado anteriormente, emita o comando "PR#n" ou "IN#n", onde "n" corresponde ao slot onde estiver ligado o cartão controlador de drive (CCE DISK CARD). Desse modo, normalmente, o comando deve ser:

PR#6 ou IN#6
(se o slot 6 for o padrão)

ATIVANDO O CCE INTEGER

O CCE INTEGER pode ser usado somente após sua carga na memória do EXATO. Para tanto, é recomendável que você disponha da expansão de 16KB (CCE RAM CARD-16). Dispondo desta expansão, após a carga do CDOS, digite:

]INT [CR]

para passar ao CCE INTEGER. A volta ao CCE BASIC se dará através do comando

>FP [CR]

Se você não dispuser da expansão, após a carga do CDOS, digite:

]BRUN CCE INTEGER EM DISCO [CR]

Nesse caso entretanto, lembre-se que você perderá 5KB de memória disponível ao programa e, quando você voltar ao CCE BASIC, todo seu programa CCE INTEGER será perdido da memória do EXATO.

APÊNDICE E:

EDIÇÃO DE PROGRAMAS

A ocorrência de erros durante a digitação de programas é um fato normal. Para facilitar a correção destes "descuidos" a CCE incorporou um conjunto único de dispositivos de edição ao CCE BASIC.

Para fazer uso deles, você precisa em primeiro lugar familiarizar-se com as funções de quatro teclas especiais do teclado do computador. Elas são a tecla "escape", marcada "ESC", a tecla de repetição, marcada ."REPT", e as teclas de seta à esquerda e seta à direita, (← e →).

ESC

A tecla "escape" (ESC) é a tecla mais à esquerda na segunda fileira a partir de cima. Ela é SEMPRE usada com outra tecla (tais como as teclas A, B, C ou D) deste modo: aperte e solte ESC e em seguida aperte e solte A , por exemplo...

Esta operação ou sequência da tecla ESC e em seguida outra tecla, é escrita como "ESC-A". As seguintes funções de escape usadas para edição:

ESC-A (ou ESC-J) move o cursor para a direita

ESC-B (ou ESC-K) move o cursor para a esquerda

ESC-C (ou ESC-M) move o cursor para baixo

ESC-D (ou ESC-I) move o cursor para cima

Usando a tecla "escape" e a tecla desejada, o cursor pode ser movido para qualquer posição na tela, sem afetar o que já esteja em exibição na tela, bem como na memória.

Observação: As funções entre parênteses só necessitam que a tecla ESC seja pressionada na primeira vez. Por exemplo: ESC A ESC A ESC A corresponde a ESC J J J.

TECLA SETA À DIREITA

A tecla seta à direita move o cursor para a direita. É a tecla que mais economiza tempo em todo o teclado, pois não somente move o cursor, mas copia todos os caracteres e símbolos sobre os quais se move para a memória do computador exatamente como se você os tivesse digitado no teclado. A tela não é alterada quando você usa a tecla [->].

TECLA SETA À ESQUERDA

A tecla seta à esquerda move o cursor para a esquerda. A cada vez em que o cursor é movido para a esquerda, um caráter é apagado na memória da linha de programa que você está digitando, sem considerar sobre o que o cursor está se movendo. A tela não é alterada quando você usa a tecla seta à esquerda. Usualmente a tecla seta à esquerda não pode ser usada para mover o cursor para a coluna mais à esquerda, use ESC-B para fazer esta operação.

REPT

A tecla [REPT] é usada com outra tecla de caráter no teclado. Causará a repetição do caráter, enquanto as teclas do caráter e [REPT] forem apertadas simultaneamente.

Agora você está pronto para usar estas funções de edição para poupar tempo ao fazer alterações ou correções em seu programa. Aqui estão alguns exemplos de como usá-las.

Exemplo 1 - Corrigindo Caracteres

Suponha que você fez entrar um programa, digitando-o, e quando você o executa, o computador exibe "?SINTAX ERRO" e para, apresentando o caráter de prontidão e o cursor.

Faça entrar o seguinte programa e execute-o. Observe que "PRIMT" e "PREGRAMA" estão escritas erroneamente de propósito. Abaixo está como ele aparece na tela:

```
10 PRIMT "ISTO E UM PREGRAMA"
20 GOTO 10
RUN
?SINTAX ERRO EM 10
```

Agora digite a palavra LIST e aperte [CR] :

```
LIST
10 PRIMT "ISTO E UM PREGRAMA"
20 GOTO 10
```

Para mover o cursor para cima no início da linha 10, digite [ESC][I] [I] [I] [B].

NOTA: É importante usar ESC-B para colocar o cursor sobre o primeiro dígito do número da linha.

Agora aperte a tecla seta à direita 6 vezes para mover o cursor até a letra M em "PRIMT". Lembre-se, à medida em que o cursor se move sobre um caráter na tela, aquele caráter é copiado na memória do computador, exatamente como se você o tivesse digitado no teclado. Agora digite a letra N para corrigir a grafia de "PRIMT", em seguida copie (usando a tecla seta à direita e a tecla de repetição) até à letra E em "PREGRAMA", e pressione a tecla O . Se você "passou" o cursor demais, use a tecla seta à esquerda para voltar para a letra E. Copie usando a tecla seta à direita até o fim da linha 10. Finalmente, armazene a nova linha na memória do programa apertando a tecla [CR] .

Digite LIST para ver seu programa corrigido:

```
LIST
10 PRINT "ISTO E UM PROGRAMA"
20 GOTO 10
```

Agora execute o programa (use CTRL-C para parar o programa):

```
RUN
ISTO E UM PROGRAMA
ISTO E UM PROGRAMA
ISTO E UM PROGRAMA
ISTO E UM PROGRAMA
ISTO E UM PROGRAMA
ISTO E UM PROGRAMA
PAUSA EM 10
```

Exemplo 2 - Inserindo Texto em uma Linha Existente

No exemplo anterior, suponha que você tivesse desejado inserir um comando TAB(1Ø) após o PRINT na linha 1Ø. Aqui está como você poderia fazê-lo. Primeiramente liste a linha a ser alterada:

```
LIST 1Ø
1Ø PRINT "ISTO E UM PROGRAMA"
```

Digite ESC-D e ESC-B até que o cursor esteja sobre o primeiro caráter da linha a ser alterada; em seguida use as teclas seta à direita e repetição para copiar até a primeira aspa. (lembre-se, um caráter não é copiado para a memória até que você use a tecla seta à direita para mover o cursor a partir daquele caráter para o próximo). Agora, digite outro ESC-D para mover o cursor para a linha vazia exatamente acima da linha corrente. Digite os caracteres a serem inseridos, os quais, neste caso, são TAB(1Ø); sua tela mostrará isto:

```
LIST 1Ø
        TAB (1Ø);
1Ø PRINT "ISTO E UM PROGRAMA"
```

Digite um ESC-C para mover o cursor uma linha para baixo e dê espaços para trás para a primeira aspa usando ESC-B (usando a tecla seta à esquerda aqui apagaria os caracteres que você acabou de digitar). A partir daqui, copie o resto da linha usando a seta à direita e a tecla de repetição até que a tela tenha este aspecto:

```
LIST 1Ø
        TAB (1Ø);
1Ø PRINT "ISTO E UM PROGRAMA"
```

Aperte a tecla [CR] e digite LIST para obter o seguinte:

```
LIST
1Ø PRINT TAB (1Ø); "ISTO E UM PR
   OGRAMA"
2Ø GOTO 1Ø
```

Onde você deseja evitar a cópia de espaços extras que o formato do comando LIST introduz no meio das linhas (tais como aqueles entre o "R" e o "O" de PROGRAMA, no exemplo acima), use ESC-A ou ESC-J.

Estes comandos movem o cursor para a direita sem copiar caracteres. Isto pode ser especialmente útil quando copiando comandos PRINT, INPUT e REM, onde o CCE BASIC não ignora os espaços extras.

Lembre-se que usando a tecla "escape", pode-se copiar e editar textos exibidos em qualquer lugar da tela.

APÊNDICE C:

MENSAGEM DE ERRO

Depois que um erro ocorre, o CCE BASIC volta para o nivel de comando, como indicado pelo caráter "]". Valores de variáveis e o texto do programa permanecem intactos, mas o programa não pode ser continuado, e todos os contadores de laço GOSUB e FOR são posicionados em Ø.

Para evitar a interrupção de um programa que está correndo, o comando ONERR GOTO pode ser usado, em combinação com uma rotina de manipulação de erros.

Quando um erro ocorre num comando de execução imediata, nenhum número de linha é exigido.

Formato de mensagens de erro:

Comando de execução imediata ?XX ERRO
Comando de execução programada ?XX ERRO EM YY

Em ambos os exemplos acima "XX" corresponde ao nome do erro específico. "YY" é o número de linha do comando de execução programada onde o erro ocorreu. Em seguida, os possíveis códigos de erro e seus significados.

?"CONT" ILEGAL

Tentativa de continuar um programa quando não havia nenhum, ou após a ocorrência de um erro, ou depois que uma linha foi apagada ou acrescentada a um programa.

?DIVISAO POR ZERO

Dividir por zero é um erro.

?DIRETO ILEGAL

Você não pode usar um comando INPUT, DEF, FN, GET, ou DATA como um comando de execução imediata.

?VALOR ILEGAL

O parâmetro passado para uma função cadeia ou matemática estava fora de intervalo. Erros do tipo VALOR ILEGAL podem ocorrer devido a:

a) um índice negativo de arranjo (por exemplo, LET A(-1) = Ø)

b) uso do LOG com um argumento negativo ou zero

c) uso do SQR com um argumento negativo

d) A^B com A negativo e B que não seja um inteiro

e) uso de MID$, LEFT$, RIGHT$, WAIT, PEEK, POKE, TAB, SPC, ON...GOTO, ou qualquer função gráfica com um argumento impróprio.

?"NEXT" SEM "FOR"

A variável num comando NEXT não correspondeu à variável num comando FOR que ainda estava em vigor, ou um comando NEXT sem nome não correspondeu a qualquer comando FOR que ainda estava em efeito.

?FIM DE DATA

Um comando READ foi executado, mas todos os comandos DATA no programa já haviam sido lidos. O programa tentou ler dados em demasia, ou foram incluídos dados, insuficientes no programa.

?FALTA MEMORIA

Qualquer das seguintes condições podem causar este erro: programa muito grande; variáveis em demasia; laços FOR agrupados em mais de 1Ø níveis de profundidade; comandos GOSUB agrupados a mais de 24 níveis de profundidade; uma expressão demasiado complicada; parênteses agrupados a mais de 36 níveis de profundidade; tentativa de posicionar LOMEM: muito alto; tentativa de posicionar LOMEM: mais baixo que o valor atual; tentativa de posicionar HIMEM: muito baixo.

?FORMULA COMPLEXA

Foram executados mais de dois comandos do tipo IF "XX" THEN consecutivos.

?S/ESPACO

O resultado de um cálculo foi grande demais para ser representado no formato de número do CCE BASIC. Se ocorre o contrário, o resultado dado é zero, e a execução continua, sem que qualquer mensagem de erro seja apresentado.

?REDIMENSIONAR

Depois que uma matriz foi dimensionada, outro comando de dimensionamento foi encontrado, para a mesma matriz. Este erro ocorre frequentemente se uma matriz foi dada a dimensão 10 por falta de definição. Assim, um comando como A(I)=3 e seguido mais tarde no programa por um DIM A(100). Esta mensagem de erro pode provar sua utilidade se você deseja descobrir em que linha do programa uma certa matriz foi dimensionada: apenas insira um comando DIM para aquele arranjo na primeira linha, execute o programa, e o CCE BASIC lhe indicará onde está o comando DIM original.

?"RETURN" SEM "GOSUB"

Um comando RETURN ou POP foi encontrado sem um comando GOSUB correspondente executado.

?"STRING" LONGA

Foi feita uma tentativa com o uso do operador de concatenação para criar uma sequência com mais de 255 caracteres.

?INDICE ILEGAL

Foi feita uma tentativa de referenciar um elemento de matriz, o qual está fora das dimensões da matriz. Este erro pode ocorrer, se um número errado de dimensões for usado por exemplo: LET A (1,1,1)=Z, quando A foi dimensionado usando DIM A (2,2).

?SINTAX

Perda de parênteses em uma expressão, caráter ilegal em uma linha, pontuação incorreta, etc...

?TIPO INCOMPAT

O lado esquerdo de um comando de atribuição foi uma variável numérica, e o lado direito foi uma cadeia, ou vice-versa; ou a uma função que esperava um argumento cadeia, foi dada uma função numérica ou vice-versa.

?LINHA INDEFINIDA

Foi feita uma tentativa para GOTO, GOSUB ou THEN para um número de linha de comando que não existe.

?FUNCAO INDEF.

Foi feita uma referência a uma função que não foi definida pelo usuário.

APÊNDICE D:

ECONOMIZANDO ESPAÇO

A fim de fazer seu programa ajustar-se em menos espaço (área) de memória, as seguintes sugestões podem ser úteis. Entretanto, os dois primeiros economizadores de espaço devem ser considerados somente quando se enfrenta sérias limitações de espaço. Os programadores mais preocupados com este tipo de problema, frequentemente mantêm duas versões de seus programas: uma expandida e muito bem documentada (com REMs), e outra versão "moída" para usar o mínimo espaço de memória.

1) Use múltiplos comandos por linha. Há uma pequena quantidade de memória gasta (5 bytes) associada a cada linha de programa. Dois desses cinco bytes contém o número de linha em binário. Isto significa que não importa quantos dígitos tenha o seu número de linha (o mínimo é Ø, o máximo é 65529), ele toma o mesmo número de bytes (dois). Colocando o máximo de comandos possíveis em cada linha, diminuirá o número de bytes usados pelo seu programa (uma única linha pode incluir até 239 caracteres).

   NOTA: A combinação de muitos comandos em uma linha torna muito difícil a edição e outras alterações. Isto também faz com que o programa seja difícil para ler e entender, não apenas para outras pessoas como também para você, quando voltar ao programa futuramente.

2) Apague todos os comentários. Cada comando REM usa no mínimo um byte mais o número de bytes no texto do próprio comentário. Por exemplo, o comando 13Ø REM ISTO E UM COMENTARIO usa até 27 bytes de memória. No comando 14Ø X=X+Y: REM ATUALIZA SOMA o REM usa 15 bytes de memória incluindo o dois-pontos antes do REM.

   NOTA: Assim como em programas com múltiplos comandos por linha, um programa sem comentários detalhados é muito difícil de ler e entender, não somente para outras pessoas, como também para você quando voltar ao programa futuramente.

3) Use matrizes de números inteiros em lugar de matrizes de números reais onde quer que seja possível (veja informação para Alocação de Memória, mais adiante neste apêndice).

4) Use variáveis em lugar de constantes. Suponha que você use a constante 3.14159 dez vezes em seu programa. Se você inserir um comando 10 P=3.14159 no programa, e usar P em lugar de 3.14159 a cada vez que for necessário, você economizará 40 bytes. Isto também resultará em melhoria na velocidade.

5) Um programa não necessita terminar com um comando END; assim, um comando END no fim de um programa pode ser apagado.

6) Reutilize as mesmas variáveis. Se você tem uma variável T a qual é usada para conter um resultado temporário em uma parte do programa e você precisa de uma variável temporária posteriormente em seu programa, use-a novamente. Ou, se você pede ao usuário do computador para dar respostas SIM ou NÃO para duas perguntas diferentes em tempos diferentes durante a execução do programa, use a mesma variável A$ para armazenar a resposta.

7) Use GOSUBs para executar seções de comandos de programa que desempenham ações idênticas.

8) Use os elementos das matrizes; por exemplo, A(0), B(0,X).

9) Quando A$="EDGARD" é reatribuida para A$="CERES" a sequência velha "EDGARD" não é apagada da memória. Um comando da forma X=FRE(0) periodicamente em seu programa causará a "limpeza da casa" das velhas sequências do topo da memória.

INFORMAÇÃO PARA ALOCAÇÃO DE MEMÓRIA

Variáveis simples (não matrizes) inteiras, reais ou tipo cadeia tal como V, V% ou V$ usam 7 bytes. Variáveis reais usam 2 bytes para o nome da variável e 5 bytes para o valor (1 para o expoente, 4 para a mantissa). Variáveis inteiras usam 2 bytes para o nome da variável, dois bytes para o valor e tem zeros nos três bytes. Variáveis tipo cadeia usam 2 bytes para o nome da variável, 1 byte para o comprimento da cadeia, 2 bytes para um endereçador para a posição da cadeia na memória, e tem zeros nos 2 bytes restantes. Veja o mapa no apêndice I.

Matrizes de variáveis reais usam 12 bytes no mínimo: 2 bytes para o nome da variável, dois para o tamanho da matriz, um para o número de dimensões, dois para o tamanho de cada dimensão, e cinco bytes para cada elemento da matriz. Matrizes de variáveis inteiras usam somente 2 bytes para cada elemento da matriz. Matrizes de variáveis cadeias usam 3 bytes para cada elemento da cadeia: um para o comprimento, dois para um endereçador.

Variáveis tipo cadeia, simples ou matrizes usam um byte de memória para cada caráter da cadeia. As cadeias são localizadas em ordem de ocorrência no programa, começando em HIMEM:.

Quando uma nova função é definida por um comando DEF, são usados 6 bytes para armazenar o endereçador para a definição.

Palavras reservadas tais como FOR, GOTO ou NOT, e os nomes das funções residentes tais como COS, INT e STR$ ocupam somente um byte da memória do programa. Todos os outros caracteres em programas usam um byte de memória do programa cada um. Quando um programa está sendo executado, o espaço é alocado dinamicamente nas pilhas conforme segue:

1) Cada laço FOR...NEXT ativo usa 16 bytes.

2) Cada GOSUB (a qual não tenha ainda retornado) usa 6 bytes.

3) Cada parênteses encontrado em uma expressão usa 4 bytes, e cada resultado temporário calculado em uma expressão usa 12 bytes.

APÊNDICE E:

ACELERANDO SEU PROGRAMA

As sugestões deste apêndice deverão melhorar o tempo de execução de seu programa CCE BASIC. Observe que algumas destas sugestões são as mesmas que foram dadas para economizar o espaço de memória utilizado pelos seus programas. Isto significa que em muitos casos você pode tornar seu programa mais rápido e ao mesmo tempo melhorar a eficiência no uso de memória.

1) Esta é, provavelmente, a sugestão mais eficiente para o aumento da velocidade do seu programa. Use variáveis em lugar de constantes. Leva mais tempo converter uma constante para sua representação (número real) em ponto flutuante do que ir buscar o valor de uma variável simples ou de uma variável matriz. Isto é especialmente importante em laços FOR...NEXT ou outras execuções repetidas.

2) Variáveis que são encontradas em primeiro durante a execução de um programa BASIC são alocadas no início da tabela de variáveis. Isto significa que um comando tal como:

   5A=Ø : B=A : C=A

   colocará A em primeiro, B em segundo, e C em terceiro na tabela de variáveis (assumindo que 5 é o primeiro comando executado no programa). Mais tarde no programa, quando o CCE BASIC acha uma referência às variáveis, ele pesquisará somente uma entrada na tabela de variáveis para encontrar A, duas entradas para achar B, e três entradas para encontrar C, etc.

3) Use o comando NEXT sem a variável índice. NEXT é um tanto mais rápido do que NEXT I porque nenhuma verificação é feita para conferir a variável especificada no comando FOR mais recente ainda ativo.

4) Durante a execução de um programa, quando o CCE BASIC encontra uma referência a uma nova linha tal como "GOTO 1ØØØ" ele procura o programa do usuário inteiro, começando pela linha mais baixa, até encontrar o número de linha referenciado (1ØØØ, neste exemplo). Consequentemente, linhas frequentemente referenciadas devem ser colocadas o mais perto possível no programa.

APÊNDICE F:

CÓDIGOS PARA PALAVRAS-CHAVE

símbolo decimal	palavra chave	símbolo decimal	palavra chave	símbolo decimal	palavra chave
128	END	164	LOMEM:	200	+
129	FOR	165	ONERR	201	-
130	NEXT	166	RESUME	202	*
131	DATA	167	RECALL	203	/
132	INPUT	168	STORE	204	^
133	DEL	169	SPEED=	205	AND
134	DIM	170	LET	206	OR
135	READ	171	GOTO	207	>
136	GR	172	RUN	208	=
137	TEXT	173	IF	209	<
138	PR#	174	RESTORE	210	SGN
139	IN#	175	&	211	INT
140	CALL	176	GOSUB	212	ABS
141	PLOT	177	RETURN	213	USR
142	HLIN	178	REM	214	FRE
143	VLIN	179	STOP	215	SCRN(
144	HGR2	180	ON	216	PDL
145	HGR	181	WAIT	217	POS
146	HCOLOR=	182	LOAD	218	SQR
147	HPLOT	183	SAVE	219	RND
148	DRAW	184	DEF	220	LOG
149	XDRAW	185	POKE	221	EXP
150	HTAB	186	PRINT	222	COS
151	HOME	187	CONT	223	SIN
152	ROT=	188	LIST	224	TAN
153	SCALE=	189	CLEAR	225	ATN
154	SHLOAD	190	GET	226	PEEK
155	TRACE	191	NEW	227	LEN
156	NOTRACE	192	TAB(	228	STR$
157	NORMAL	193	TO	229	VAL
158	INVERSE	194	FN	230	ASC
159	FLASH	195	SPC(	231	CHR$
160	COLOR=	196	THEN	232	LEFT$
161	POP	197	AT	233	RIGHT$
162	VTAB	198	NOT	234	MID$
163	HIMEM:	199	STEP		

APÊNDICE G:

PALAVRAS RESERVADAS NO CCE BASIC

&

ABS	AND	ASC	AT	ATN			
CALL	CHR$	CLEAR	COLOR=	CONT	COS		
DATA	DEF	DEL	DIM	DRAW			
END	EXP						
FLASH	FN	FOR	FRE				
GET	GOSUB	GOTO	GR				
HCOLOR=	HGR	HGR2	HIMEM:	HLIN	HOME	HPLOT	HTAB
IF	IN#	INPUT	INT	INVERSE			
LEFT$	LEN	LEF	LIST	LOAD	LOG	LOMEM:	
MID$							
NEW	NEXT	NORMAL	NOT	NOTRACE			
ON	ONERR	OR					
PDL	PEEK	PLOT	POKE	POP	POS	PRINT	PR#
READ	RECALL	REM	RESTORE	RESUME	RETURN	RIGHT$	
	RND	ROT=	RUN				
SAVE	SCALE=	SCRN(	SGN	SHLOAD	SIN	SPC(	
	SPEED=	SQR	STEP	STOP	STORE	STR$	
TAB(	TAN	TEXT	TO	TRACE			
USR							
VAL	VLIN	VTAB					
WAIT							
XPLOT	XDRAW						

O CCE BASIC transforma em símbolos estas palavras reservadas: cada palavra ocupa somente um byte na memória do programa. Todos os outros caracteres na memória do programa ocupam um byte cada. Veja o apêndice F para os símbolos das palavras-reservadas.

NOTA: O sinal gráfico (&) destina-se somente ao uso do computador; ele não é propriamente um comando CCE BASIC. Este símbolo, quando executado como uma instrução, causa um desvio incondicional para a posição $3F5. Para recuperar, use [RESET] CTRL-C [CR].

XPLOT é uma palavra reservada que não corresponde a um comando CCE BASIC corrente.

Algumas palavras reservadas são reconhecidas pelo CCE BASIC somente em determinados contextos:

- COLOR, HCOLOR, SCALE, SPEED e ROT são analisadas sintaticamente como palavras reservadas somente se o próximo caráter diferente de espaço for o sinal de igual (=). Isto é de pouco benefício, no caso de COLOR e HCOLOR, visto que a palavra reservada, incluída OR, evita o uso delas em nomes de variáveis de qualquer modo.

- SCRN, SPC e TAB são analisadas sintaticamente como palavras reservadas somente se o próximo caráter diferente de espaço for o parêntese esquerdo (().

- HIMEM precisa ter seus dois-pontos (:) para ser analisada sintaticamente como palavra reservada.

- LOMEM também requer um dois-pontos (:) para ser considerada palavra reservada.

- ATN é analisada como palavra reservada somente se não existir espaço entre o T e o N. Se ocorrer um espaço entre o T e o N, a palavra AT será considerada em lugar de ATN.

- TO é analisada como palavra reservada, a menos que seja precedida por um A e exista um espaço entre o T e o O. Se ocorrer um espaço entre o T e o O, a palavra AT será considerada em lugar de TO.

As vezes, podem ser usados parênteses para manter junto palavras reservadas:

```
100 FOR A = LOFT OR CAT TO 15
```

é listado como

```
100 FOR A = LOF TO RC AT TO 15
```

porém

```
100 FOR A = (LOFT) OR (CAT) TO 15
```

é listado como

```
100 FOR A = (LOFT) OR (CAT) TO 15
```

APÊNDICE H:

CONVERTENDO BASIC EM CCE BASIC

Embora as implementações do BASIC em diferentes computadores sejam semelhantes em vários pontos, há algumas incompatibilidades que você deve esperar se está planejando converter programas BASIC para CCE BASIC.

1) Índice de matrizes: alguns BASICs usam colchetes em lugar de parênteses para indicar índices de matrizes.

2) Cadeias: certos BASICs forçam o dimensionamento (declarar) do comprimento das cadeias antes de usá-las. Você deve remover todos os comandos de dimensão deste tipo de programa. Em alguns desses BASICs, uma declaração da forma DIM A$(I,J) declara uma matriz tipo cadeia de J elementos cada um dos quais tem um comprimento I. Converta comandos DIM deste tipo para o equivalente em CCE BASIC: DIM A$(J).
O CCE BASIC usa "+" para concatenação de cadeias, não "," ou "&".
O CCE BASIC usa LEFT$, RIGHT$ e MID$ para tirar subcadeias de cadeias. Outros BASICs usam A$(I) para acessar o I-ésimo caráter da cadeia A$, e A$(I,J) para tirar uma subcadeia de A$ a partir do caráter na posição I para o caráter na posição J. Observe as conversões, abaixo:

OUTROS BASICs	CCE BASIC
A$(I)	MID$(A$,I,1)
A$(I,J)	MID$(A$,I,J-I+1)

Assim é assumido que a referência a uma subcadeia de A$ está em uma expressão ou está no lado direito de uma atribuição. Se a referência a A$ está no lado esquerdo de uma atribuição, e X$ é a expressão de cadeia usada para recolocar caracteres em A$, converta como segue:

OUTROS BASICs	CCE BASIC
A$(I)=X$	A$=LEFT$(A$,I-1)+X$+MID$(A$,I+1)
A$(I,J)=X$	A$=LEFT$(A$,I,1)+X$+MID$(A$,J+1)

3) Alguns BASICs permitem comandos de "múltipla atribuição" da forma: 500 LET B = C = 0 Este comando posicionaria as variáveis B e C em zero. Em CCE BASIC isto tem um efeito inteiramente diferente. Todos os sinais de igual (=) à direita do primeiro seriam interpretados como operadores de comparação lógica. No nosso caso, esta linha associaria a variável B com -1 se C fosse igual a 0. Se C não fosse igual a 0, B seria associado a 0. O modo mais fácil de converter comandos como este é reescrevê-lo como segue:

```
500 C=0 : B=0
```

4) Alguns BASICs usam "/" em lugar de ":" para separar comandos múltiplos na mesma linha. Troque cada "/" por ":" no programa.

5) Programas que usam as funções MAT disponíveis em alguns BASICS terão que ser reescritos usando laços FOR...NEXT para desempenhar as operações apropriadas.

APÊNDICE I:

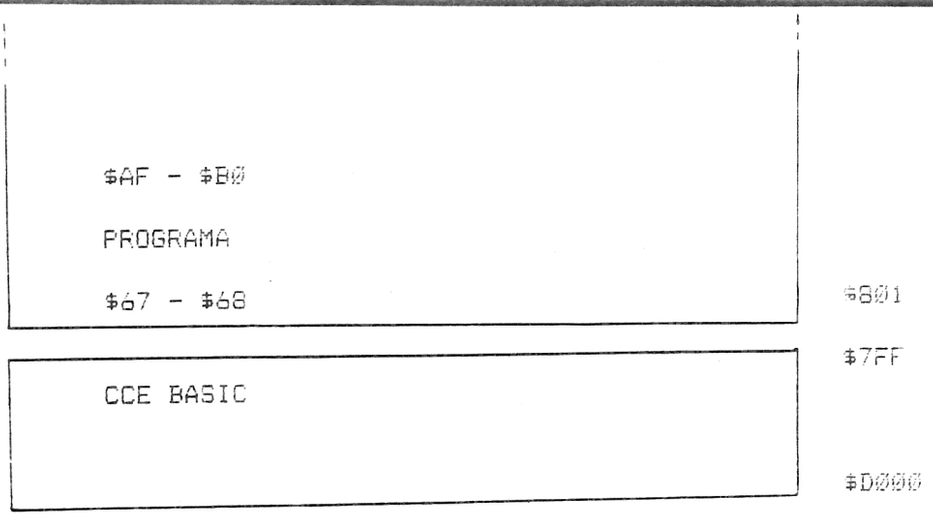
MAPA DA MEMORIA

INTERVALOS DA MEMÓRIA	DESCRIÇÃO
$0.1FF	Espaço de trabalho do programa: não disponível ao usuário.
$200.2FF	Memória auxiliar dos caracteres do teclado.
$300.3FF	Disponível ao usuário para pequenos programas em linguagem de máquina.
$400.7FF	Área de exibição na tela para página 1 de texto ou gráficos em cores.
$800.XXX	Contém o espaço para as variáveis e o programa do usuário, onde XXX é o endereço máximo de memória RAM a ser usada pelo CCE BASIC. Isto é a memória RAM total do sistema, ou menos, se o usuário está reservando parte da memória alta para rotinas em linguagem de máquina, ou para memória auxiliar para a tela de Alta Resolução.
$2000.3FFF	Área de exibição na tela para página 1 de gráficos de Alta Resolução.
$4000.5FFF	Página 2 exibição dos gráficos de Alta Resolução.
$C000.CFFF	Endereços de equipamento de Entrada/Saída.
$D000.FFFF	Memória ROM.
$D000.F7FF	CCE BASIC.
$F800.FFFF	Sistema Monitor.

DIAGRAMA DE MAPA DA MEMÓRIA

Endereçador

Sistema Operacional em Disco (se está sendo usado disco)
$73 - $74 (HIMEM:) HIMEM: é posicionado automaticamente na posição de memória RAM máxima do sistema, a menos que seja posicionada pelo usuário.
CADEIAS $6F - $70
ESPAÇO LIVRE Incluindo as memórias auxiliares dos gráficos de alta resolução. Observação: O espaço para cadeias pode ser preenchido com dados velhos e invadir as áreas de telas de alta resolução ou programas em linguagem de máquina. Para iniciar a limpeza da casa e evitar este problema, insira X=FRE(0) em seu programa.
$6D - $6E ARRANJOS NUMÉRICOS E ARRANJOS DE SEQUÊNCIAS $6B - $6C
VARIÁVEIS SIMPLES $69 - $6A (LOMEM:)


$AF - $B0
PROGRAMA
$67 - $68
$8001
$7FFF
CCE BASIC
$D000

APÊNDICE J:

PEEKs, POKEs E CALLs

Aqui estão algumas características especiais do CCE BASIC que você pode usar por meio dos comandos PEEK, POKE ou CALL. Note que alguns deles duplicam os efeitos de outros comandos.

Ações simples de ligação são usualmente dependentes de endereços. Qualquer comando envolvendo aquele endereço terá o mesmo efeito na chave. Assim, o exemplo pode ser:

```
POKE-16304,0
```

porém você obterá o mesmo efeito com a alteração do conteúdo daquele endereço com qualquer número de 0 a 255 através do comando POKE, ou pelo exame do conteúdo daquele endereço com:

```
X=PEEK (-16304)
```

Isto não se aplica a comandos nos quais você deve alterar o endereço requerido com um valor específico, o qual fixa uma margem ou move o cursor para um lugar específico.

POSICIONANDO A JANELA DE TEXTO

Os primeiros quatro comandos POKE, com exemplos nas linhas números 10, 20, 30 e 40 (ver adiante nesta seção), são usados para posicionar o tamanho da "janela" na qual o texto é mostrado e passado na sua tela. Eles determinam, respectivamente, a margem esquerda, largura de linha, margem superior e margem inferior da janela.

A definição da janela de texto não limpa o resto da tela, e não move o cursor para dentro da janela de texto (use HOME ou HTAB e VTAB). O comando VTAB ignora inteiramente a janela de texto: o texto impresso acima da janela aparece normalmente, enquanto que o texto impresso abaixo da janela aparece todo em uma linha. HTAB também pode mover o cursor para fora da janela, mas somente o bastante para exibir um caráter lá.

Uma alteração na largura da linha entra em ação imediatamente, porém uma alteração na margem esquerda não é detectada até que o cursor tente "voltar" para a margem esquerda.

NOTA: O texto exibido na tela é meramente um mapa especial de uma porção particular da memória do computador (página de texto 1). A tela sempre "olha" para esta mesma porção de memória para o seu texto, e vê o que o computador escreveu lá. Quando você altera a janela de texto, está dizendo ao computador onde, na memória, escrever seu texto, o que funciona perfeitamente enquanto você especifica uma porção de memória que esta dentro da área de texto usual, porém se você posiciona a margem esquerda, digamos, em 255 (o máximo devia ser 40, uma vez que a tela tem 40 posições de largura), você está dizendo ao computador para escrever o texto muito além da área de memória reservada para texto. Esta memória não é mostrada na tela, e pode conter partes do seu programa ou até mesmo informações necessárias ao Sistema Operacional. Para manter o seu programa e o Sistema Operacional protegidos, não defina a janela de texto além dos limites de tela de 40 caracteres por 24 linhas.

10 POKE 32,L

Posiciona a margem esquerda da tela no valor especificado por L, no intervalo de 0 a 39, onde 0 é a posição mais à esquerda. Esta alteração não é afetada até que o cursor tente "voltar" para a margem esquerda.

NOTA: a largura da janela não é alterada por este comando: isto significa que a margem direita será movida pela mesma quantidade que você move a margem esquerda. Para preservar o seu programa e o CCE BASIC, primeiro reduza a largura da janela apropriadamente, em seguida, altere a margem esquerda.

20 POKE 33,W

Posiciona a largura (número de caracteres por linha) da tela para o valor especificado por W, no intervalo de 1 a 40. Não posicione W em zero: POKE 33,0 danifica o CCE BASIC.

NOTA: Se W for menor do que 33, o terceiro campo da tabulação do comando PRINT pode exibir caracteres fora da janela.

30 POKE 34,T

Posiciona a margem superior da tela no valor especificado por T, no intervalo de 0 a 23, onde 0 é a linha no topo da tela.

Um POKE 34,4 não permitirá a exibição de texto nas primeiras quatro linhas da tela. Não posicione a margem superior da janela (T) mais baixo do que a margem inferior (B, abaixo).

40 POKE 35,B

Posiciona a margem inferior da tela no valor especificado por B, no intervalo de 0 a 24, onde 24 é a linha no rodapé da tela. Não posicione a margem inferior da janela (B) mais alto do que a margem superior (T, acima).

OUTROS COMANDOS QUE AFETAM O TEXTO, A JANELA DE TEXTO E O TECLADO

45 CALL-936

Suprime todos os caracteres dentro da janela de texto, e move o cursor para a posição de exibição no topo mais à esquerda da janela. Tem o mesmo efeito que ESC - @ e o comando HOME.

50 CALL-958

Limpa todos os caracteres dentro da janela de texto a partir da posição corrente do cursor até a margem inferior. Os caracteres acima do cursor, e à esquerda do cursor não serão afetados. Isto é o mesmo que ESC F.

Se o cursor estiver acima da janela de texto, limpa a partir do cursor para a direita, margem esquerda e margem inferior como se a margem superior estivesse acima do cursor. Normalmente não se usa esse comando se o cursor está abaixo da margem inferior da janela de texto: usualmente a linha no rodapé da janela é limpa, junto com uma linha da largura de texto na posição do cursor.

60 CALL-868

Limpa a linha corrente a partir do cursor para a margem direita. Tem o mesmo efeito que ESC - E.

70 CALL-922

Pula uma linha. Tem o mesmo efeito que CTRL-J.

```
80 CALL-912
```

Faz o texto correr para cima uma linha; isto é, move cada linha de texto dentro da janela definida uma posição para cima. A linha que estava no topo é perdida; a segunda linha torna-se a primeira, e a última linha da tela é agora brancos. Caracteres definidos fora da janela não são afetados.

```
90 X=PEEK (-16384)
```

Lê o teclado. Se X>127, então uma tecla foi apertada, e X é o valor ASCII da tecla apertada com o bit 7 ligado (=1). Isto é útil em programas longos, nos quais o computador confere para ver se o usuário deseja interromper com novos dados sem parar a execução do programa.

```
100 POKE-16368,0
```

Reposiciona o teclado de modo que o próximo caracter possa ser lido, o que deve ser feito imediatamente após a leitura do teclado.

COMANDOS QUE LIDAM COM O CURSOR

```
110 CH=PEEK (36)
```

Lê a posição horizontal corrente do cursor e associa a variável CH com esse valor. CH estará no intervalo de 0 a 39 e é a posição do cursor relativa à margem esquerda da janela de texto, conforme definida pelo comando POKE 32,L. Assim, se a margem esquerda foi fixada por POKE 32,5, então o caráter mais à esquerda na janela está na sexta posição a partir da extremidade esquerda da tela e se PEEK (36) devolveu o valor 5, então o cursor estava na décima primeira posição a partir da extremidade esquerda da tela, e na sexta posição, a partir da margem esquerda da janela de texto (a posição mais à esquerda é a posição zero, não 1). Obtém-se a mesma função com POS(X). (veja o próximo exemplo).

```
120 POKE 36,CH
```

Move o cursor para uma posição que está CH+1 posições a partir da margem esquerda da janela de texto. (Exemplo: POKE 36,0 causará a exibição do próximo caráter na margem esquerda da janela).

Se a margem esquerda da janela foi fixada em 6 (POKE 32,6) e você queria fornecer um caráter três posições a partir da extremidade da tela, então a margem esquerda da janela deve ser alterada antes de exibir. CH deve ser menor ou igual à largura da janela conforme definida por POKE 22,W e deve ser maior ou igual a zero. À exemplo de HTAB, este comando pode mover o cursor para além da margem direita da janela de texto, mas apenas o suficiente para exibir um caráter.

```
130 CV=PEEK (37)
```

Lê a posição vertical corrente do cursor e associa CV com esse valor. CV é a posição vertical absoluta do cursor e não é referenciada em relação às margens superior e inferior da janela de texto. Assim, CV=0 é a linha de topo da tela, e CV=23 é a linha de rodapé da tela.

```
140 POKE 37,CV
```

Move o cursor para a posição vertical absoluta especificada por CV. Zero é a linha de topo e 23 é a linha de rodapé.

COMANDOS QUE AFETAM OS GRÁFICOS

Com o propósito de exibir textos e gráficos, a memória do computador está dividida em 4 áreas: páginas de texto 1 e 2, e páginas de alta resolução 1 e 2.

1) Página de texto 1: é a área de memória usual para todos os textos e gráficos de Baixa Resolução, usada pelos comandos TEXT e GR.

2 Página de texto 2: encontra-se imediatamente acima da página de texto 1 na memória. Não é facilmente acessível ao usuário. À exemplo da página de texto 1, a informação armazenada na página de texto 2 pode ser interpretada como texto com gráficos de Baixa Resolução, ou ambos.

3) A página 1 dos gráficos de Alta Resolução reside na memória do computador a partir de 8K a 16K. Esta é a área usada pelo comando HGR. Se o texto é mostrado com esta página, ele vem da página de texto 1.

4) A página 2 dos gráficos de Alta Resolução reside na memória de 16K a 24K. Esta é a área usada pelo comando HGR2. Se o texto é mostrado com esta página, ele vem da página de texto 2.

Para usar as modalidades diferentes de gráficos e textos, você pode usar os comandos de textos e gráficos do CCE BASIC, ou pode operar estas 4 chaves diferentes. Como muitas das chaves discutidas aqui, um PEEK ou POKE para um endereço posiciona a chave de uma maneira, e um PEEK ou POKE para um segundo endereço posiciona a chave de outra maneira. Em resumo, estas 4 chaves escolhem entre:

1) Exibição de texto e gráficos,	(POKE-16303,0)
Alta ou Baixa Resolução.	(POKE-16304,0)
2) Página 1 de texto ou Alta Resolução	(POKE-16300,0)
e página 2 de texto ou Alta Resolução	(POKE-16299,0)
3) Página de texto 1 ou 2 para gráficos e pági-	(POKE-16298,0)
na 1 ou 2 para gráficos de Alta Resolução	(POKE-16297,0)
4) Gráficos de Alta ou Baixa Resolução. Tela	(POKE-16302,0)
cheia e mista, gráficos de Alta ou Baixa	(POKE-16301,0)
Resolução mais texto.	

```
150 POKE-16304,0
```

Troca o modo de exibição de texto para gráficos em cores sem limpar a tela de gráficos em preto. Dependendo do posicionamento das outras 3 chaves, a modalidade de gráficos escolhida pode ser de Baixa ou Alta Resolução, a partir da página 1 e 2, e em gráficos mais texto, ou gráficos em tela cheia.

Similar aos comandos CCE BASIC, o comando GR seleciona página 1 Baixa Resolução, tela gráfica mais texto, e limpa a tela de gráficos em preto. O comando HGR seleciona página 1 Alta Resolução, tela gráfica mais texto, e limpa a tela de gráficos em preto. O comando HGR2 seleciona página 2 de Alta Resolução, gráficos em tela cheia, e limpa a tela inteira em preto.

```
160 POKE-16303,0
```

Troca a modalidade de exibição de quaisquer gráficos em cores para a modalidade texto completo sem redefinição da janela de texto.

O comando TEXT seleciona modalidade para texto completo, porém adicionalmente escolhe a página 1 de texto, redefine a janela de texto no máximo e posiciona o cursor no canto inferior esquerdo da tela.

17Ø POKE-16302,Ø

Troca o modo de exibição de tela gráfica mais texto para gráficos em tela cheia.

Dependendo do posicionamento das outras chaves, esta pode aparecer como texto, como gráficos de Baixa Resolução sobre uma grade de 4Ø por 48, ou como gráficos de Alta Resolução sobre uma grade de 278 por 192.

18Ø POKE-16301,Ø

Troca o modo de exibição de gráficos em tela cheia para tela gráfica mais texto, com quatro linhas de 4Ø caracteres de texto no rodapé da tela.

Dependendo do posicionamento das outras chaves, a parte superior da tela pode mostrar texto, gráficos de Baixa Resolução numa grade de 4Ø por 4Ø ou gráficos de Alta Resolução sobre uma grade de 278 por 16Ø. Ambas as porções da tela virão do mesmo número de página (1 ou 2).

184 POKE-16300,Ø

Passa da página 2 para página 1, sem limpar a tela ou mover o cursor. É necessário quando você passa do CCE BASIC para o INTEGER CCE. De outro modo, você pode estar ainda "olhando" para a página 2 da memória.

Dependendo do posicionamento das outras chaves, isto pode causar a troca de gráficos de Alta Resolução página 2, para gráficos de Alta Resolução página 1, de gráficos de Baixa Resolução página 2, para gráficos de Baixa Resolução página 1, ou de textos página 2 para textos página 1.

186 POKE-16299,Ø

Passa da página 1 para a página 2, sem limpar a tela ou mover o cursor.

Dependendo do posicionamento das outras chaves, isto pode causar a troca de gráficos de Alta Resolução página 1 para gráficos de Alta Resolução página 2, de gráficos de Baixa Resolução página 1 para gráficos de Baixa Resolução página 2, ou de textos página 1 para textos página 2.

190 POKE-16298,0

Troca a página para gráficos, de uma página de gráficos de Alta Resolução para a mesma página de texto, sem limpar a tela.

É necessário quando você passa do CCE BASIC para o CCE INTEGER. De outro modo, a instrução GR do CCE INTEGER pode mostrar incorretamente a página de Alta Resolução.

Dependendo do posicionamento das outras chaves, isto pode causar a troca de gráficos de Alta Resolução página 1 para gráficos de Baixa Resolução página 1, de gráficos de Alta Resolução página 2 para gráficos de Baixa Resolução página 2, ou (no modo texto) pode não causar qualquer alteração na exibição.

195 POKE-16297,0

Troca a página para gráficos de uma página de texto para a página correspondente de Alta Resolução, sem limpar a tela.

Dependendo do posicionamento das outras chaves, isto pode causar a troca de gráficos de Baixa Resolução página 1 para gráficos de Alta Resolução página 1, de gráficos de Baixa Resolução página 2 para gráficos de Alta Resolução página 2, ou (no modo texto) pode não causar qualquer alteração na exibição.

200 CALL-1994

Preenche as 20 linhas superiores da página de texto 1 com caracteres de arroba (@) reverso. Se você está no modo gráfico de Baixa Resolução página 1, limpa as 40 linhas mais altas da tela de gráficos em preto. Não tem efeito sobre a página 2, ou sobre gráficos de Alta Resolução.

205 CALL-1998

Limpa a página de texto inteira para sinais @ reversos. Se você está no modo gráficos de Baixa Resolução em tela cheia página 1, limpa a tela inteira em preto. Não tem efeito sobre a página 2 de texto ou sobre gráficos de Alta Resolução.

206 CALL 62450

Limpa a tela de Alta Resolução corrente (o CCE BASIC registra qual tela você usou por último, independentemente do posicionamento das chaves) em preto.

210 CALL 62454

Limpa a tela de Alta Resolução corrente (o CCE BASIC registra qual tela você usou por último, independentemente do posicionamento das chaves), para a cor mais recentemente definida pelo comando HCOLOR= e traçada pelo comando HPLOT. Deve ser precedida por uma operação de traçado.

COMANDOS QUE LIDAM COM CONTROLADORES DE JOGO E ALTO-FALANTES.

220 X=PEEK (-16336)

Aciona o auto-falante uma vez: produz um estalo (clique).

255 X=PEEK (-16352)

Aciona a saída em cassete uma vez: produz um "clique" na gravação do cassete.

230 X=PEEK (-16287)

Lê o botão de pressão no controle de jogo # 0. Se X>127 então o botão está sendo apertado.

240 X=PEEK (-16286)

O mesmo que acima, mas no controlador de jogo # 1.

250 X=PEEK (-16285)

Botão no controlador de jogo # 2.

260 POKE-16296,1

Aciona a saída do "anunciador" do controlador de jogo # 0 (conector E/S de jogo, pino 15) para TTL coletor aberto alto (3,5 volts). Esta é a condição "desligada".

270 POKE-16295,0

Aciona a saída do controlador de jogo # 0 para TTL baixo (0,3 volts). Esta é a condição "ligada" (corrente máxima 1,6 miliamperes).

```
280 POKE-16294,1
```

Posiciona saída do controlador de jogo # 1 (conector E/S de jogo, pino 14) para TTL alto (3,5 volts).

```
290 POKE-16293,0
```

Aciona a saída do controlador de jogo # 1 para TTL baixo (0,3 volts).

```
300 POKE-16292,1
```

Aciona a saída do controlador de jogo # 2 (conector E/S de jogo, pino 13) para TTL alto (3,5 volts).

```
310 POKE-16291,0
```

Aciona a saída do controlador de jogo # 2 para TTL baixo (0,3 volts).

```
320 POKE-16290,1
```

Aciona a saída de controlador de jogo # 3 (conector E/S de jogo, pino 12) para TTL alto (3,5 volts).

```
330 POKE-16289,0
```

Aciona a saída do controlador de jogo # 3 para TTL baixo (0,3 volts).

COMANDOS RELACIONADOS A ERROS

```
340 X=PEEK (218) + PEEK (219)*256
```

Esse comando associa X com o número da linha do comando, onde ocorreu um erro se um comando ONERR GOTO foi executado.

```
350 IF PEEK (216) > 127 THEN GOTO 2000
```

Se o bit 7 na posição de memória 222 (ERREND) foi associado a verdadeiro, então um comando ONERR GOTO foi encontrado.

```
360 POKE 216,0
```

Limpa ERREND de modo que as mensagens de erros normais possam ocorrer.

```
370 Y=PEEK (222)
```

Posiciona a variável Y com um código que descreve o tipo de erro que causou a ocorrência do salto ONERR GOTO. Os tipos de erro são descritos abaixo:

VALOR DE Y	TIPO DE ERRO ENCONTRADO
0	"NEXT" SEM "FOR"
16	SINTAX
22	"RETURN" SEM "GOSUB"
42	FIM DE DATA
53	VALOR ILEGAL
69	S/ESPACO
77	FALTA MEMORIA
90	LINHA INDEFINIDA
107	INDICE ILEGAL
120	REDIMENSIONAR
133	DIVISAO POR ZERO
163	TIPO INCOMPAT
176	"STRING" LONGA
191	FORMULA COMPLEXA
224	FUNCAO INDEF.
254	RESPOSTA ILEGAL A UM COMANDO INPUT
255	TENTADA UMA INTERRUPCAO COM CTRL-C

```
380 POKE 768,104: POKE 769,168: POKE 770,104: POKE 771,166:
    POKE 772,223: POKE 773,154: POKE 774,72 : POKE 775,152:
    POKE 776,72 : POKE 777,96
```

Estabelece uma sub-rotina em liguagem de máquina na posição 768, a qual pode ser usada em uma rotina de manipulação de erro. Resolve alguns problemas com ONNER GOTO com PRINT e mensagens ?FALTA MEMORIA ERRO. Use o comando CALL-768 na rotina de manipulação de erro.

MAPA DAS VARIAVEIS CCE BASIC

VARIAVEIS SIMPLES

ENDERECO $69-$6A	REAL	INTEIRA	APONT. DE CADEIA
	NOME(pos) 17 byte (pos) 27 byte	NOME(reg) 17 byte (reg) 27 byte	NOME(reg) 17 byte (pos) 27 byte
	expoente 1 byte mantissa M.S.byte mantissa mantissa mantissa 1.S.byte	byte alto byte baixo Ø Ø Ø	tamanho 1 byte ender byte baixo ender byte alto Ø Ø

VARIÁVEIS TIPO MATRIZ

ENDEREÇO	REAL	INTEIRA	APONT. DE CADEIA
$6B-$6C	NOME(pos) 17 byte (pos) 27 byte	NOME(reg) 17 byte (reg) 27 byte	NOME(reg) 17 byte (pos) 27 byte
	DESVIA o endereçador para o próximo endereço deste nome variável. byte baixo byte alto	DESVIA o endereçador para o próximo endereço deste nome variável. byte baixo byte alto	DESVIA o endereçador para o próximo endereço deste nome variável. byte baixo byte alto
	Nº DE DIMENSÕES 1 byte	Nº DE DIMENSÕES 1 byte	Nº DE DIMENSÕES 1 byte
	TAM. DA N-ESIMA DIMENSÃO byte alto byte baixo	TAM. DA N-ESIMA DIMENSÃO byte alto byte baixo	TAM. DA N-ESIMA DIMENSÃO byte alto byte baixo
	TAM. DA 1ª DIMENSÃO byte alto byte baixo	TAM. DA 1ª DIMENSÃO byte alto byte baixo	TAM. DA 1ª DIMENSÃO byte alto byte baixo
	REAL(Ø,Ø,...,Ø) expoente 1 byte mantissa MSbyte mantissa mantissa mantissa 1Sbyte	INT%(Ø,Ø,...,Ø) byte alto byte baixo	CAD$(Ø,Ø,...,Ø) compto 1 byte end byte baixo end byte alto
$6D-$6E	REAL(N,N,...,N) expoente 1 byte mantissa MSByte mantissa mantissa $6D-$6E mantissa 1Sbyte	INT%(N,N,...,N) byte alto byte baixo	CAD$(N,N,...,N) compto 1 byte end byte baixo end byte alto

As cadeias são armazenadas em ordem de entrada, a partir do HIMEM: para baixo. A tabela de cadeia aponta para o primeiro caráter de cada cadeia, no final da cadeia na memória. Conforme as cadeias são alteradas, novos endereços são escritos; quando a memória disponível está esgotada, uma arrumação de casa apaga todas as cadeias abandonadas (a arrumação de casa é forçada por um comando FRE(X)).

Todas as matrizes são armazenadas com o índice mais à direita subindo o mais lento; por exemplo, os números na matriz A%(1,1) onde A%(0,0)=0, A%(1,0)=1, A%(0,1)=2, A%(1,1)=3 seriam encontrados na memória na sequência apropriada.

APÊNDICE K:

CÓDIGOS DE CARACTERES ASCII

DEC = código decimal ASCII

HEX = código hexadecimal ASCII

CARAT = nome do caráter ASCII

n/a = não acessível diretamente a partir do teclado do computador.

DEC	HEX	CARAT	O QUE DIGITAR
Ø	ØØ	NULL	CTRL @
1	Ø1	SOH	CTRL A
2	Ø2	STX	CTRL B
3	Ø3	ETX	CTRL C
4	Ø4	ET	CTRL D
5	Ø5	ENQ	CTRL E
6	Ø6	ACK	CTRL F
7	Ø7	BEL	CTRL G
8	Ø8	BS	CTRL H ou <-
9	Ø9	HT	CTRL I
1Ø	ØA	LF	CTRL J
11	ØB	VT	CTRL K
12	ØC	FF	CTRL L
13	ØD	CR	CTRL M ou CR
14	ØE	SO	CTRL N
15	ØF	SI	CTRL O
16	1Ø	DLE	CTRL P
17	11	DC1	CTRL Q
18	12	DC2	CTRL R
19	13	DC3	CTRL S
2Ø	14	DC4	CTRL T
21	15	NAK	CTRL U ou ->
22	16	SYN	CTRL V
23	17	ETB	CTRL W
24	18	CAN	CTRL X
25	19	EM	CTRL Y
26	1A	SUB	CTRL Z
27	1B	ESCAPE	ESC
28	1C	FS	n/a
29	1D	GS	CRTL shift-M
3Ø	1E	RS	CTRL ^
31	1F	US	n/a

DEC	HEX	CARAC	O QUE DIGITAR
32	2Ø	ESPACO	ESPACO
33	21	!	!
34	22	"	"
35	23	#	#
36	24	$	$
37	25	%	%
38	26	&	&
39	27	'	'
4Ø	28	(	(
41	29	)	)
42	2A	*	*
43	2B	+	+
44	2C	,	,
45	2D	-	-
46	2E	.	.
47	2F	/	/
48	3Ø	Ø	Ø
49	31	1	1
5Ø	32	2	2
51	33	3	3
52	34	4	4
53	35	5	5
54	36	6	6
55	37	7	7
56	38	8	8
57	39	9	9
58	3A	:	:
59	3B	;	;
6Ø	3C	<	<
61	3D	=	=
62	3E	>	>
63	EF	?	?
64	4Ø	@	@
65	41	A	A
66	42	B	B
67	43	C	C
68	44	D	D
69	45	E	E
7Ø	46	F	F
71	47	G	G
72	48	H	H
73	49	I	I
74	5Ø	J	J
75	4B	K	K
76	4C	L	L
77	4D	M	M
78	4E	N	N

DEC	HEX	CARAC	O QUE DIGITAR
79	4F	O	O
80	50	P	P
81	51	Q	Q
82	52	R	R
83	53	S	S
84	54	T	T
85	55	U	U
86	56	V	V
87	57	W	W
88	58	X	X
89	59	Y	Y
90	5A	Z	Z
91	5B	[	(shift-B)
92	5C	\	(shift-V)
93	5D	]	(shift-M)
94	5E	^	^
95	5F	_	(shift-C)

Os códigos ASCII no intervalo de 96 a 255 gerarão caracteres no computador que repetem a lista acima (primeiramente aqueles na coluna 2, em seguida a série completa novamente). Embora a função CHR$(65) devolva um A, e a função CHR$(193) também devolva um A, o CCE BASIC não reconhece os dois como o mesmo caráter quando usando operadores lógicos de cadeias, e uma impressora conectada ao seu computador iria imprimi-los de maneira diferente.

APÊNDICE L

USO DA PÁGINA ZERO DO CCE BASIC

POSIÇÕES (em hexa)	USO
$Ø-$5	Instruções de salto para continuar em CCE BASIC ([RESET] ØG [CR] para CCE BASIC é equivalente a RESET CTRL-C [CR] para CCE INTEGER).
$A-$C	Posições para instruções de salto de funções USR. Veja a descrição da função USR.
$D-$17	Contadores/flags de uso geral para o CCE BASIC.
$2Ø-$4F	Posições reservadas ao Sistema Monitor do EXATO.
$5Ø-$61	Endereçadores de uso geral para CCE BASIC.
$62-$66	Resultado da última multiplicação/divisão.
$67-$68	Endereçador para o início do programa.
$69-$6A	Endereçador para o início da área de variáveis simples. Também aponta para o fim do programa mais 1 ou 2, a menos que seja alterado com o comando LOMEM:.
$6B-$6C	Endereçador para o começo da área de matrizes.
$6D-$6E	Endereçador para o fim da área de armazenamento numérico em uso.
$6F-$7Ø	Endereçador para o início da área de armazenamento de cadeias. As cadeias são armazenadas a partir dessa posição até o fim da memória.
$71-$72	Endereçador geral.
$73-$74	Posição de memória mais alta disponível ao CCE BASIC mais um. A partir da entrada inicial para o CCE BASIC, é posicionado na posição mais alta de memória RAM disponível.
$75-$76	Número de linha corrente que está sendo executada.

POSIÇÕES (em hexa)	USO
$77-$78	"Número velho" de linha. Posicionado por um CTRL-C, STOP ou um comando END. Dá o número de linha na qual a execução foi interrompida.
$79-$7A	"Endereçador de texto velho". Aponta para a posição de memória que contém a próxima instrução a ser executada.
$7B-$7C	Número de linha corrente a partir da qual estão sendo lidos de um comando DATA os dados pela instrução READ.
$7D-$7E	Aponta para a posição de memória absoluta a partir da qual estão sendo lidos de um comando DATA os dados pela instrução READ.
$7F-$80	Endereçador da fonte corrente de INPUT. Posicionado em $201 durante um comando INPUT. Durante um comando READ e posicionado no comando DATA do qual seus dados estão sendo lidos.
$81-$82	Segura o nome da última variável usada.
$83-$84	Endereçador para o valor da última variável usada.
$85-$9C	Uso geral.
$9D-$A3	Acumulador principal de ponto flutuante.
$A4	Uso geral em rotinas matemáticas de ponto flutuante.
$A5-$AB	Acumulador secundário de ponto flutuante.
$AC-$AE	Endereçadores/flags de uso geral.
$AF-$B0	Endereçador para fim de programa (não alterado por LOMEM:)
$B1-$C8	Rotina CHRGET. O CCE BASIC "chama" esta localização aqui sempre que quer outro caráter.
$B8-$B9	Aponta para o último caráter obtido através da rotina CHRGET.

POSIÇÕES (em hexa)	USO
$C9-$CD	Número aleatório.
$D0-$D5	Endereçadores de gráficos de Alta Resolução.
$D8-$DF	Endereçadores ONERR.
$E0-$E2	Coordenadas X e Y dos gráficos de Alta Resolução.
$E4	Byte de cor dos gráficos de Alta Resolução.
$E5-$E7	Uso geral para gráficos de Alta Resolução.
$E8-$E9	Endereçador para início da "tabela de forma".
$EA	Contador de colisão para gráficos de Alta Resolução.
$F0-$F3	Flags de uso geral.
$F4-$F8	Endereçadores ONERR.

APÊNDICE M:

DIFERENÇAS ENTRE O CCE BASIC E O CCE INTEGER

Estes comandos são disponíveis em CCE BASIC, porém não em CCE INTEGER:

ATN						
CHR$	COS					
DATA	DEF FN	DRAW				
EXP						
FLASH	FN	FRE				
GET						
HCOLOR	HGR	HGR2	HIMEM:	HOME	HPLOT	
INT	INVERSE					
LEFT$	LOG	LOMEM:	.			
MID$						
NORMAL						
ON...GOSUB		ON...GOTO		ONERR GOTO		
POS						
READ	RECALL	RESTORE	RESUME	RIGHT$	ROT	
SCALE	SHLOAD	SIN	SPC	SPEED	SQR	STOP
	STORE	STR$				
TAN						
USR						
VAL						
WAIT						
XDRAW						

Estes comandos são disponíveis em CCE INTEGER, porém não em CCE BASIC:

AUTO	
DSP	
MAN	MOD

Estes comandos têm nomes diferentes nas linguagens:

CCE INTEGER	CCE BASIC
CLR	CLEAR
CON	CONT
TAB	HTAB (obs: O CCE BASIC também tem um TAB)
GOTO X*10+100	ON X GOTO 100, 110, 120
GOSUB X*100+1000	ON X GOSUB 1000, 1100, 1200
CALL-936	HOME (ou CALL-936)
POKE 50,127	INVERSE
POKE 50,255	NORMAL
X	X% (% indica variável inteira)
#	<> ou ><

OUTRAS DIFERENÇAS

Em CCE INTEGER, a correção da sintaxe de um comando é conferida quando o comando é armazenado na memória do computador (quando você aperta a tecla [CR]). Em CCE BASIC tal conferência é feita quando um comando é executado.

GOTO ou GOSUB devem ser seguidos por um número de linha em CCE BASIC. O CCE INTEGER permite uma variável aritmética ou expressão.

Variáveis reais e constantes (número de "ponto flutuante", com pontos decimais e/ou expoentes) são permitidas em CCE BASIC, porém não em CCE INTEGER.

Em CCE BASIC, somente os dois primeiros caracteres num nome de variável são significativos. Em CCE INTEGER, todos os caracteres num nome de variável são significativos.

Operações de cadeia são definidas de maneira diferente nas duas linguagens. Tanto as cadeias como as matrizes devem ser dimensionados em CCE INTEGER. No CCE BASIC, somente as matrizes devem ser dimensionadas.

Em CCE BASIC as matrizes podem ser multi-dimensionais, ao passo que no CCE INTEGER, as matrizes são limitadas a uma dimensão.

O CCE BASIC zera todos os elementos da matriz, ao executar RUN, CLEAR, RESET CTRL-B CR. Em CCE INTEGER, o programa do usuário é que deve gerar os elementos da matriz.

Quando a asserção em CCE INTEGER do comando IF...THEN avalia como zero (falso), somente a porção THEN do comando é ignorada. Em CCE BASIC, todos os comandos seguintes ao THEN, ou na mesma linha, são ignorados quando a asserção IF avalia como zero, (falso): a execução do programa salta para a próxima linha numerada do programa.

Em CCE BASIC, o comando TRACE exibe o número da linha de cada instrução individual em uma linha de programa com múltiplas instruções, não apenas a da primeira instrução, como no CCE INTEGER.

Em CCE BASIC, os comandos CALL, PEEK e POKE podem usar o verdadeiro intervalo de endereços das posições de memória (0 a 65535). Em CCE INTEGER, as posições com endereços maiores do que 32767 devem ser referidas pelos seus valores negativos do complemento de dois (posição 32768 é chamada -32767-1; 32769 é chamada -32767; 32770 é chamada -32766 etc).

O comando END em um programa, que para no número de linha mais alto, é opcional em CCE BASIC, porém é requerido em todos os casos para evitar mensagem de erro em CCE INTEGER.

O comando NEXT deve ser seguido por um nome de variável em CCE INTEGER. Em CCE BASIC o nome de variável é opcional.

Em CCE INTEGER, a sintaxe do comando INPUT é:

INPUT [cadeia,] {var}

Se /var/ é uma /avar/, então INPUT exibe um ponto de interrogação (?) com ou sem a cadeia opcional. Se /var/ é um /svar/, então nenhum ponto de interrogação é exibido, se a cadeia opcional estiver ou não presente.

APÊNDICE N:

GLOSSÁRIO DE DEFINIÇÕES SINTÁTICAS E ABREVIAÇÕES

Neste apêndice, tais definições e abreviações são apresentadas na ordem alfabética. Veja o Capítulo 2 para uma apresentação lógica destas definições. O símbolo := significa "é, pelo menos, parcialmente definido como".

CADEIA

:= "[{caráter}]"
Uma cadeia ocupa 1 byte (8 bits) para seu comprimento, 2 bytes para seu endereço e 1 byte para cada caráter de cadeia.

CARÁTER

:= letra|dígito símbolos especiais

CARÁTER ALFANUMÉRICO

:= letra|dígito

CONTROL

:= mantenha comprimida a tecla [CTRL] enquanto a tecla nomeada em seguida é pressionada.

DELIMITADOR

:= (|)|=|-|+|^|>|<|/|*|,|;|:
Um nome não tem que ser separado de uma precedente ou seguinte palavra reservada por qualquer destes delimitadores.

DÍGITO

:= 1|2|3|4|5|6|7|8|9|0

ESCAPE

:= uma depressão da tecla [ESC]

EXPRESSÃO

:= expr

expr
:= aexpr|sexpr

caráter de prontidão
:=]

EXPRESSÃO ARITMÉTICA

:= aexpr

aexpr

:= avar real inteiro

:= avar real índice

:= (aexpr)

Se os parênteses estiverem agrupados em mais de 36 níveis de profundidade, ocorrerá a mensagem: ?FALTA MEMORIA ERRO

:= (+ - NOT) aexpr

NOT aparece somente aqui ao lado de um único + e -

:= aexpr op aexpr

EXPRESSÃO CADEIA

:= sexpr

ÍNDICE

:= (aexpr [{,aexpr}])

O número máximo de dimensões é 89, embora na prática exista a limitação de memória disponível. /aexpr/ deve ser positiva e em uso é convertida para um inteiro.

avar

:= avar índice

Em matrizes, os reais ocupam 5 bytes e os inteiros 2 bytes.

aexpr := avar índice

INTEIRO

:= [+|-] {dígito}

Inteiros devem estar dentro do intervalo de -32767 a 32767. Quando convertendo não inteiros para inteiros, o CCE BASIC geralmente trunca o não-inteiro para o próximo menor inteiro. Contudo, isto não é verdadeiro no limite à medida que o não-inteiro aproxima-se do próximo maior inteiro. Por Exemplo:

```
A% = 123.999 999 959 999   B% = 123.999 999 969
PRINT A%                   PRINT B%
  123                        124
```

```
C% = 12345.999 995 999        D% = 12345.999 996 999
PRINT C%                      PRINT D%
  12345                         12346
```

Uma matriz inteiro ocupa 2 bytes (16 bits) na memória.

LETRA

:=A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R|S|T|U|V|W|X|Y|Z

LINHA

:= numlinha [{instrução:}]
Uma linha pode ter até 239 caracteres, o que inclui todos os espaços digitados pelo usuário, porém não inclui os espaços adicionados pelo CCE BASIC na formatação da linha.

LITERAL

:= [{caráter}]

METANOME

:= meta-símbolo [dígito]

META-SÍMBOLO

:= são caracteres usados, neste manual, para indicar diversas estruturas ou relações do CCE BASIC, mas que não fazem parte da linguagem propriamente dita.

:= letra minúscula

:= |[|]|{|}|\|

:= um simples dígito concatenado com um metanome.

NOME

:= letra {letra|dígito}
Um nome pode ter até 238 caracteres de extensão. Quando distinguido um nome de outro, o CCE BASIC ignora qualquer caráter alfanumérico após os dois primeiros. Entretanto, até mesmo a porção ignorada de um nome não deve conter um caráter especial, uma aspa (") ou qualquer das "palavras reservadas". (Veja o apêndice G para a lista destas palavras reservadas e comentários sobre as exceções a esta regra).

NOME DE VARIAVEL CADEIA
:= nome$

NOME DE VARIAVEL INTEIRA
:= nome%
Um número real pode ser armazenado como uma variável inteira, sendo então convertido para inteiro.

NÚMERO DA LINHA
:= numlinha

numlinha
:= dígito [{dígito}]
Os números de linha devem estar dentro do intervalo Ø a 63999 ou resultará a mensagem ?SINTAX ERRO.

OPERADOR
:= op

op
:= aop alop

OPERADOR ARITMÉTICO
:= aop

aop
:=|+|-|*|/|^

OPERADOR DE CADEIA
:= sop

sop
:= +

OPERADOR LÓGICO ARITMÉTICO
:= alop

alop
:= AND|OR|=|>|<|<>|><|>=|=>|<=|=<

NOT não está incluído aqui propositadamente.

OPERADOR LÓGICO DE CADEIA
:= slop

slop
:= =|>|>=|=>|<|<=|=<|<>|><

aexpr
:= sexpr slop sexpr

REAL

:= [+|-] {dígito} [.{dígito}][E[+|-]{dígito}[dígito]]
:=[+|-] [{dígito}].[{dígito}][E[+|-]{dígito}[dígito]]

A letra E, como usada na notação de número real (na forma de "notação científica"), vale por "expoente". É a forma rápida de escrever * 10^. Dez é elevado à potência do número à direita do E, e o número à esquerda do E é multiplicado pelo resultado. No CCE BASIC os números reais devem estar no intervalo -1E38 a 1E38, ou você se arrisca a uma mensagem de erro do tipo:

?S/ESPACO ERRO

Usando adição ou subtração, pode-se gerar números tão grandes como 1.7E38 sem receber esta mensagem.

Um número real cujo valor absoluto é menor do que 2.9388E-39 será convertido a zero.

O CCE BASIC reconhece os seguintes números reais quando apresentados por si mesmos e os avalia em zero:

. +. -. .E +.E -.E

.E+ .E- +.E- +.E+ -.E+ -.E-

Consequentemente, o elemento da matriz M(.) é o mesmo que M(Ø).

Em adição aos reais abreviados listados acima, os seguintes são reconhecidos como reais e avaliados em zero quando usados como respostas numéricas para INPUT ou como elementos numéricos de DATA:

+ - E +E espaço

E+ E- +E+ +E- -E-

A instrução GET avalia todos os reais de um caráter nas listas acima como zero.

Quando exibido um número real, o COE BASIC mostrará até o máximo de nove dígitos (veja exceção, abaixo), excluindo o expoente (se houver). Quaisquer dígitos posteriores serão cortados.

À esquerda do ponto decimal, quaisquer zeros precedendo o dígito não-zero mais à esquerda, não são exibidos. À direita do ponto decimal, quaisquer zeros seguintes ao dígito não-zero mais à direita, não serão exibidos. Se não existirem dígitos não-zero à direita do ponto decimal, o ponto decimal não é apresentado.

NOTA: O arredondamento pode ser curioso:

```
PRINT 99999999.9
99999999.9

PRINT 99999999.99
100000000

PRINT 11.11111145000
11.1111115

PRINT 11.1111114519
11.1111114
```

Se o valor absoluto de um número real é maior ou igual a .01 e menor do que 999999999.2, o número real é apresentado em notação de ponto fixo, que significa que nenhum expoente é exibido.

No intervalo 0.1000000005 a .999999999 os reais são exibidos com até dez dígitos, incluindo o zero imediatamente à direita do ponto decimal. Esta é a única exceção ao limite de nove dígitos apresentados, excluindo o expoente.

Se você tentar usar um número com mais de 36 dígitos, tal como
211.111111111111111111111111111111111
então a mensagem:

?S/ESPACO ERRO

é exibida, mesmo que o número esteja claramente dentro do intervalo -1E38 a 1E38. Isto é verdadeiro mesmo que a maioria dos dígitos sejam zeros não significativos, como no número 211.00000000000000000000000000000000000000. Zeros à esquerda, contudo, são ignorados. Se o primeiro dígito é um "1", e o segundo dígito é menor ou igual a "6", números com 39 dígitos podem ser usados sem receber mensagem de erro. Um número real ocupa 5 bytes (40 bits) na memória.

REINICIALIZA (REPOSICIONA)
:= uma depressão da tecla [RESET]

RETORNO (ENTRA)
:= uma depressão da tecla [CR]

SÍMBOLO ESPECIAL
:= !|#|$|%|&|'|(|)|*|:|=|-|@|+|;|?|/|>|.|<|, |^|"|\
_|]|[
Caracteres de controle (caracteres que são digitados com a tecla [CTRL] comprimida) e o caráter nulo também são especiais.
O CCE BASIC usa o colchete direito (]) somente como caráter de prontidão neste documento.

VARIÁVEL ARITMÉTICA
:= avar

avar
:= nome nome%

Todas as variáveis simples ocupam 7 bytes na memória, 2 bytes para o nome e 5 bytes para o valor real ou inteiro.

VARIÁVEL TIPO CADEIA
:= svar

svar
:= nome$ nome$ índice

O endereço e o nome da variável ocupam 2 bytes cada na memória. O comprimento e cada caráter ocupa 1 byte cada.

APÊNDICE O:

SUMÁRIO DOS COMANDOS

ABS (-3.451)

Devolve o valor absoluto do argumento. O exemplo devolve 3.451.

TECLAS DE SETAS

As teclas marcadas com as setas direita (→) e esquerda (←), são usadas para editar programas. A tecla seta direita move o cursor para a direita. À medida que ela faz isto, cada caráter que ela percorre na tela, é inserido como se você o houvesse digitado. A tecla de seta esquerda move o cursor para a esquerda; conforme ele se move, um caráter é apagado em memória da linha de programa, a qual você está digitando, sem considerar sobre o que o cursor está se movendo.

ASC ("QUEEN")

Devolve o código decimal ASCII para o primeiro caráter no argumento. No exemplo, 81 (ASCII para Q) será devolvido.

ATN (2)

Devolve o arco-tangente, em radianos, do argumento. No exemplo, 1.1Ø714872 (radianos) será devolvido.

CALL-922

Causa a execução de uma rotina em linguagem de máquina, na posição de memória cujo endereço decimal é especificado. O exemplo causará um salto de linha.

CHR$(65)

Devolve o caráter ASCII, que corresponde ao valor do argumento, o qual deve estar entre Ø e 255. O exemplo devolve a letra A.

CLEAR

Zera todas as variáveis e torna nulas todas as cadeias.

COLOR=12

Define a cor a ser usada no modo gráfico de Baixa Resolução. No exemplo, a cor definida é verde.

A cor é definida em Ø por GR. Os nomes das cores e seus respectivos números são:

Ø preta	4 verde escura	8 marrom	12 verde
1 magenta	5 cinza	9 laranja	13 amarela
2 azul escura	6 azul média	1Ø cinza	14 prata
3 violeta	7 azul clara	11 rosa	15 branca

Para descobrir a cor de um dado ponto na tela, use o comando SCRN.

CONT

Se a execução do programa foi interrompida por STOP, END, CTRL-C [RESET] ou ØG [CR], o comando CONT faz com que a execução continue na próxima instrução, não na próxima linha numerada. Nada é apagado. Após [RESET], ØG[CR], o programa pode não continuar corretamente porque alguns endereçadores de programa e pilhas são apagadas. CONT não pode ser usado se você:

a) modificou, adicionou ou apagou uma linha de programa;

b) recebeu uma mensagem de erro desde a interrupção da execução.

COS (2)

Devolve o co-seno do argumento, o qual deve estar em radianos. No exemplo, -.41514683Ø é devolvido.

CTRL-C

Pode ser usado para interromper a execução do programa ou uma listagem. Ele pode, também, ser usado para interromper uma instrução INPUT, se este for o primeiro caráter inserido. A instrução INPUT não é interrompida até que a tecla [CR] seja pressionada.

CTRL-X

Informa ao computador para ignorar a linha que está sendo digitada, sem apagar nenhuma linha anterior com o mesmo número de linha. Uma barra inclinada (/) é exibida no fim da linha que deve ser ignorada.

DATA LILIAN, "CODE 32", 23.45, -6

Cria uma lista de elementos os quais podem ser usados pelos comandos READ. No exemplo, o primeiro elemento é o literal LILIAN, o segundo elemento é a cadeia "CODE 32", o terceiro elemento é o número real 23.45, e o quarto elemento é o inteiro -6.

DEF FN A (W) = 2* W+W

Permite ao usuário definir funções de uma linha em um programa. Primeiro, a função deve ser definida usando DEF; depois, no programa, a função anteriormente definida pode ser usada. O exemplo ilustra como definir uma função FN A(W); ela pode ser usada posteriormente no programa, na forma FN A(23) ou FN A(-7*0+1), e assim por diante. FN A(23) fará com que o W seja substituído por 23 em 2*W+W: a função será avaliada em 2*23+23 ou 69. Assuma 0 = 2; então FN A(-7*0+1) é equivalente a FN A(-7*2+1) ou FN A(-13): a função será avaliada em 2*(-13) +(-13) ou -26 -13 ou -39.

DEL 23,36

Remove o intervalo de linhas especificado de um programa. No exemplo, as linhas 23 até 56 serão removidas do programa. Para apagar uma única linha, digamos a linha 350, use a fórmula 350,350 ou simplesmente digite o número da linha, e então aperte a tecla CR.

DIM IDADE(20,3), NOME$(50)

Quando uma declaração é executada, ela reserva espaço para as matrizes especificadas com índices variando de 0 até o índice dado. No exemplo, NOME$(50) alocará 50+1 cadeias de qualquer comprimento, a matriz IDADE (20,3) alocará (20+1)*(3+1) ou 21*4 ou o número real 84 elementos. Se um elemento da matriz é usado em um programa, antes que ele tenha sido dimensionado, um índice máximo de 100 é fixado para cada dimensão. Elementos de matrizes são zerados quando RUN ou CLEAR são executados.

DRAW 4 AT 50,100

Desenha definição de forma número 4 a partir de uma "tabela de formas" previamente carregada em gráficos de Alta Resolução, começando em X=50, Y=100. A cor, rotação, e escala da forma a ser desenhada, deve ter sido especificada antes que DRAW seja executada.

END

Causa a interrupção da execução de um programa, e devolve o controle ao usuário. Nenhuma mensagem é exibida.

ESC-A ou ESC-B ou ESC-C ou ESC-D

A tecla "escape" pode ser usada em conjunto com as teclas das letras A, B, C ou D para mover o cursor sem afetar os caracteres sob o mesmo. Para mover o cursor um espaço, primeiro aperte a tecla "escape", então solte a tecla "escape", e aperte a tecla adequada de letra.

COMANDO	MOVE O CURSOR UM ESPAÇO
ESC-A (ou ESC-K)	para a direita
ESC-B (ou ESC-J)	para a esquerda
ESC-C (ou ESC-M)	para baixo
ESC-D (ou ESC-F)	para cima

EXP (2)

Devolve o valor de "e" elevado à potência indicada pelo argumento. Para 6 casas decimais, e = 2.718289, assim, no exemplo, 7.389Ø561 será devolvido.

FLASH

Aciona a modalidade de vídeo piscante, de modo que a saída do computador é alternadamente mostrada na tela, em caracteres brancos sobre preto, em seguida invertido para caracteres preto sobre fundo branco. Use NORMAL para voltar a exibição usual de letras brancas sobre fundo preto.

```
FOR W = 1 TO 2Ø...NEXT W
FOR Q = 5 TO -3 STEP -2...NEXT Q
FOR Z = 5 TO -4 STEP  3...NEXT
```

Permite escrever um "laço" para executar um número especificado de vezes, quaisquer instruções entre o comando FOR (no início de "laço"), e o comando NEXT (ao fim do "laço"). No primeiro exemplo, a variável W conta quantas vezes fazer as instruções; as instruções dentro do laço serão executadas para W=1,2,3,...2Ø, então o laço termina (com W=21), e a instrução após o NEXT W é executada. O segundo exemplo ilustra como indicar que o tamanho do STEP, conforme você conta, deve ser diferente de 1. A verificação tem lugar no fim do laço, assim no terceiro exemplo, as instruções dentro do laço são executadas uma vez.

FRE (Ø)

Devolve a quantidade de memória, em bytes, ainda disponíveis ao usuário. O que você põe dentro dos parênteses é irrelevante, embora seja avaliada pelo CCE BASIC.

GET ANS$

Obtém um único caráter do teclado sem mostrá-lo na tela, e sem requerer que a tecla CR seja apertada. No exemplo, o caráter digitado é armazenado na variável ANS$.

GOSUB 25Ø

Ocasiona o desvio de um programa para a linha indicada (25Ø no exemplo). Quando um comando RETURN é executado, o programa desvia para o comando imediatamente seguinte ao mais recentemente GOSUB executado.

GOTO 25Ø

Ocasiona o desvio incondicional do programa para a linha indicada (25Ø no exemplo).

GR

Aciona o modo gráfico de Baixa Resolução (4Ø por 4Ø) para a tela, deixando 4 linhas para texto na parte inferior. A tela é limpada em preto, o cursor é movido para a janela do texto, e COLOR= é posicionada em Ø (preto).

HCOLOR=4

Define a cor do gráfico de Alta Resolução na cor especificada por HCOLOR=. Os nomes das cores e seus respectivos valores são:

Ø preta	4 preta 2
1 verde (depende do vídeo)	5 (depende do vídeo)
2 azul (depende do vídeo)	6 (depende do vídeo)
3 branca	7 branca 2

HGR

Aciona o modo gráfico de Alta Resolução (28Ø por 16Ø) para a tela, deixando 4 linhas para texto. A tela é limpada em preto, e a página 1 da memória é exibida. Nem HCOLOR= nem a memória do texto da tela são afetados quando HGR é executado. O cursor é movido para a janela de texto.

HGR2

Aciona o modo gráfico de Alta Resolução tela-completa (280 por 192). A tela é limpada em preto, e a página 2 da memória é exibida. A memória do texto da tela não é afetada.

HIMEM: 16384

Especifica o endereço da mais alta localização de memória disponível a um programa CCE BASIC, incluindo variáveis. Ele é usado para proteger uma área de memória para dados, telas de Alta Resolução de rotinas em linguagem de máquina. HIMEM: não é afetado por CLEAR, RUN, NEW, DEL, alteração ou adição de uma linha de programa, ou RESET .

HLIN 10,20 AT 30

Usado para traçar linhas horizontais no modo gráfico de Baixa Resolução, utilizando a cor mais recentemente especificada pelo COLOR=. A origem (X=0 e Y=0) para o sistema é o ponto superior mais à esquerda da tela. No exemplo, a linha é desenhada de X=10 para X=20 no Y=30, ou seja, a linha é traçada do ponto (10,30) até o ponto (20,30).

HOME

Move o cursor para o canto superior esquerdo na janela de texto, e limpa todo o texto da tela.

```
HPLOT 10,20
HPLOT 30,40 TO 50,60
HPLOT 70,80
```

Traça pontos e linhas no modo gráfico de Alta Resolução, geralmente usando a cor mais recentemente especificada por HCOLOR=. A origem é o topo da tela (X=0, Y=0). O primeiro exemplo traça um ponto de Alta Resolução na posição (X=10, Y=20). O segundo exemplo traça uma linha de Alta Resolução do ponto em (X=30, Y=40) para o ponto (X=50, Y=60). O terceiro exemplo traça uma linha a partir do último ponto traçado para o ponto na posição (X=70, Y=80), usando a cor do último ponto traçado, e não necessariamente a mais recente HCOLOR=.

HTAB 23

Move o cursor à esquerda ou à direita, para a coluna especificada (1 a 40) na tela. No exemplo, o cursor será posicionado na coluna 23.

```
IF IDADE < 18 THEN A = 0 : B = 1 : C = 2
IF ANS$ = "SIM" THEN GOTO 100
IF N < MAX THEN 25
IF N < MAX GOTO 25
```

Se a expressão seguinte à palavra IF é avaliada como verdadeira (isto é, não zero), então a(s) instrução (ões) seguinte(s) à palavra THEN, na mesma linha, será (ão) executada(s). Caso contrário, todas as instruções seguintes ao THEN são ignoradas e a execução passa para a próxima instrução do programa. Expressões de cadeias são avaliadas por ordem alfabética. Os exemplos 2, 3 e 4 comportam-se de forma semelhante, a despeito das diferentes maneiras de escrever.

```
INPUT A%
INPUT "DIGITE IDADE, VIRGULA E NOME"; B, C$
```

No primeiro exemplo, o INPUT apresenta um ponto de interrogação, e espera que o usuário digite um número, o qual será associado à variável inteira A%.

No segundo exemplo, o INPUT exibe a cadeia opcional exatamente conforme mostrado, em seguida que o usuário digite um número (o qual será associado à variável real B), seguido de uma vírgula e uma cadeia (a qual será associada à variável tipo cadeia C$). Múltiplas entradas para INPUT podem ser separadas por vírgulas ou "CRs".

```
INT (NUM)
```

Devolve o maior inteiro menor ou igual a um dado argumento. No exemplo, se NUM é 2.389, então será devolvido 2; se NUM é -45.123345, então será devolvido -46.

```
INVERSE
```

Aciona a modalidade de vídeo reverso (letras pretas sobre fundo branco). Use NORMAL para voltar para letras brancas sobre fundo preto.

```
IN#4
```

Especifica o slot (de 1 até 7) das unidades periféricas que fornecerá entradas subsequentes para o computador. IN#0 restabelece entrada a partir do teclado, ao invés das unidades periféricas.

```
LEFT$("EXATO",3)
```

Devolve o número especificado dos caracteres mais à esquerda da cadeia. No exemplo, EXA (os 3 caracteres mais à esquerda) serão devolvidos.

```
LEN ("CCE BASIC")
```

Devolve o número de caracteres em uma cadeia, entre Ø e 255. No exemplo, será devolvido, 9.

```
LET A=23.567
A$="RICARDO"
```

O nome da variável à esquerda do sinal de igualdade (=) é associado ao valor da cadeia ou expressão à direita do sinal "=". O LET é opcional.

```
LIST
LIST 2ØØ-3ØØØ
LIST 2ØØ,3ØØØ
```

O primeiro exemplo faz com que o programa inteiro seja exibido na tela; o segundo exemplo exibirá as linhas de programa 2ØØ até 3ØØØ inclusive. Para listar um programa do início até a linha 2ØØ, use LIST-2ØØ; para listar a partir da linha 2ØØ até o fim do programa, use LIST 2ØØ-. O terceiro exemplo comporta-se de maneira idêntica ao segundo. A listagem é interrompida pelo CTRL-C.

```
LOAD
```

Lê um programa CCE BASIC da fita cassete para a memória do computador. Nenhum aviso é dado: o usuário deve rebobinar a fita e apertar a tecla "play" no gravador, antes de carregar. Um sinal sonoro (bip) é ouvido quando a informação é encontrada na fita que está sendo carregada. Quando a carga é completada com sucesso, um segundo "bip" soa, e o caráter de prontidão (]) retorna. Somente RESET pode interromper a carga de um programa.

```
LOG (2)
```

Devolve o logarítmo natural da expressão aritmética especificada. No exemplo, .693147181 é devolvido.

```
LOMEM: 2Ø6Ø
```

Especifica o endereço da mais baixa localização de memória disponível a um programa CCE BASIC, o que permite a proteção de variáveis de gráficos de Alta Resolução.

```
MID $ ("NEYDE C.", 3)
MID $ ("NEYDE C.", 3,4)
```

Devolve a subcadeia especificada. No primeiro exemplo, os últimos caracteres da cadeia, a partir do terceiro caráter, serão devolvidos "YDE C.". No segundo exemplo, os 4 caracteres, começando com o terceiro caráter na cadeia serão devolvidos "YDE ".

```
NEW
```

Apaga o programa corrente e todas as variáveis.

```
NEXT
```

Veja a discussão de FOR... TO... STEP

```
NORMAL
```

Posiciona a modalidade de vídeo usual (letras brancas sobre fundo preto), para entradas e saídas.

```
NOTRACE
```

Desativa a modalidade TRACE. Veja TRACE.

```
ON ID GOSUB 100, 200, 23, 4005, 500
```

Execute um GOSUB para o número de linha indicado pelo valor da expressão aritmética que segue ON. No exemplo, se ID é 1, GOSUB 100 é executada; se ID é 2, GOSUB 200 é executada, e assim por diante. Se o valor da expressão é 0 ou maior do que os números de linhas listados, a execução do programa continua na próxima linha.

```
ON ID GOTO 100, 200, 23, 4005, 500
```

Semelhante a ON ID GOSUB (veja acima), porém executa um desvio GOTO para o número de linha indicado pelo valor da expressão aritmética que segue ON.

```
ONERR GOTO 500
```

Usada para evitar uma mensagem de erro que interrompe a execução do programa, quando ocorre um erro. Quando executada, ONERR GOTO aciona um "flag" que causa um desvio incondicional para o número de linha indicado (500, no exemplo), se qualquer erro for encontrado.

PDL (3)

Devolve o valor corrente, um número de Ø a 255, da alavanca de controle do jogo indicado. Os números de Ø a 3 são válidos para as alavancas de jogos.

PEEK (37)

Devolve o conteúdo, em decimal, do byte, no endereço especificado em decimal (no exemplo, 37).

PLOT 1Ø,2Ø

No modo gráfico de Baixa Resolução, coloca um ponto na localização especificada. No exemplo, o ponto estará em (X=1Ø, Y=2Ø). A cor do ponto é determinada pelo valor mais recente de COLOR, o qual é Ø (preto), se não especificado previamente.

POKE -163Ø2,Ø

Armazena o equivalente binário do segundo argumento (Ø, no exemplo) na localização de memória, cujo endereço decimal é dado pelo primeiro argumento (-163Ø2, no exemplo).

POP

Este comando retira o último endereço da pilha (stack) de endereços para o comando RETURN. Após executado este comando, o próximo RETURN encontrado no programa desviará a execução para a linha seguinte do penúltimo GOSUB efetuado.

POS (Ø)

Devolve a posição horizontal corrente do cursor. Este é um número de Ø (na margem esquerda) a 39 (na margem direita). O que você põe dentro dos parênteses é irrelevante, não obstante seja avaliada pelo CCE BASIC.

```
PRINT
PRINT A$; "X="; X
```

O primeiro exemplo causa uma salto de linha. No segundo exemplo, ítens de uma lista a ser apresentada devem estar separados por vírgulas, se cada ítem deve ser exibido em um campo de tabulação separado. Os ítens devem ser separados por ponto-e-vírgulas (;) caso se deseje que sejam exibidos em seguida, sem qualquer interposição de espaços. Se A$ contém "COPA" e X vale 3, o segundo exemplo causará a exibição de:

COPAX=3

PR#2

Transfere as saídas do computador para o slot especificado 1 a 7. PR#Ø devolve a saída para a tela.

READ A, B%, C$

Quando executada, atribuiu às variáveis na declaração READ os valores sucessivos, a partir dos elementos nas declarações DATA do programa. No exemplo, os primeiros dois elementos, nas declarações DATA, devem ser números, e o terceiro uma cadeia. Eles serão atribuídos às variáveis A, B% e C$, respectivamente.

RECALL MX

Recupera uma matriz de números inteiros ou reais que tenha sido armazenada em fita cassete. Uma matriz pode ser chamada de volta com um nome diferente daquele que foi usado quando ela foi armazenado na fita. Quando chamado de volta, MX deve ter sido dimensionado pelo programa. Os índices não são usadas com STORE ou RECALL: no exemplo, a matriz cujos elementos são MX (Ø), MX (1),... serão recuperados; a variável sem índice não será afetada. Nenhum sinal é dado: você deve apertar a tecla "play" no gravador, quando o RECALL é executado; "bips" assinalam o início e o fim do arranjo gravado. Somente RESET pode interromper um RECALL.

REM ESTE E UM COMENTARIO

Permite a inserção de texto dentro de um programa, como comentário.

REPETIÇÃO

Se você mantiver apertada a tecla REPT , enquanto pressionando qualquer tecla de caráter, este será repetido.

RESTORE

Reposiciona o endereçador da lista de DATA para o primeiro elemento. Faz com que a próxima instrução READ encontrada releia as declarações DATA a partir do primeiro elemento.

RESUME

Ao fim de uma rotina de manipulação de erro (veja ONERR GOTO), causa a retomada do programa à instrução na qual o erro ocorreu.

RETURN

Desvia para a instrução imediatamente seguinte ao GOSUB mais recentemente executado.

RIGHT$ ("EXATO",4)

Devolve o número especificado de caracteres mais à direita da cadeia. No exemplo, XATO (os 3 caracteres mais à direita) são devolvidos.

RND (5)

Devolve um número real aleatório entre Ø e 1. RND (Ø) devolve o número aleatório mais recentemente gerado. Cada argumento negativo gera um número aleatório particular, que é o mesmo todas as vezes que RND é usado com aquele argumento, e RNDs subsequentes, com argumentos positivos, sempre seguirão uma seqüência particular que se repete. Todas as vezes que RND é usado com qualquer argumento positivo, um novo número aleatório de Ø a 1 é gerado, a menos que ele seja parte de uma seqüência de números aleatórios, iniciada por um argumento negativo.

ROT=16

Fixa o ângulo de rotação de um desenho para os comandos DRAW ou XDRAW. O comando ROT=Ø faz com que o desenho seja apresentado no ângulo em que foi definido, enquanto ROT=16 faz com que ele rode sobre o seu eixo central 9Ø graus no sentido horário. O mesmo processo será feito a partir de ROT=64.

RUN 5ØØ

Limpa todas as variáveis, enderecadores e pilhas, e começa a execução no número de linha indicado (5ØØ, no exemplo). Se nenhum número de linha é especificado, a execução começa na linha de menor número do programa.

SAVE

Armazena um programa em fita cassete. Nenhum sinal é dado; o usuário deve apertar as teclas "record" e "play" no gravador, antes que a instrução seja executada. O SAVE não confere se os botões adequados estão pressionados. "Bips" assinalam o início e o fim de uma gravação.

SCALE = 5Ø

Fixa a escala desejada nos comandos DRAW ou XDRAW. O comando SCALE=1 reproduz o desenho do mesmo tamanho que foi definido na tabela de forma. O comando SCALE=255 aumentará cada ponto 255 vezes.

NOTA: SCALE=Ø é a escala máxima e não uma redução como você poderia estar imaginando.

SCRN (1Ø,2Ø)

No modo gráfico de Baixa Resolução, devolve o código de cor do ponto especificado. No exemplo, a cor do ponto em X=1Ø, Y=2Ø é devolvida.

SGN (NUM)

Devolve -1, se o argumento é negativo; Ø, se o argumento é Ø e 1, se o argumento é positivo.

SHLOAD

Carrega uma "tabela de formas" de uma fita cassete. A tabela é carregada imediatamente abaixo do HIMEM: e em seguida, HIMEM é posicionado imediatamente abaixo da tabela de forma, para protegê-la.

SIN (2)

Devolve o seno do argumento, o qual deve estar em radianos. No exemplo, .9Ø9297427 é devolvido.

SPC (8)

Deve ser usado em um comando PRINT. Introduz o número especificado de espaços (8, no exemplo) entre o último item exibido e o próximo, se ponto-e-vírgulas (;) precedem e seguem o comando SPC.

SPEED = 5Ø

Define a velocidade em que os caracteres devem ser enviados para a tela, ou outros dispositivos de entrada/saída. A velocidade mais baixa é Ø; a maior é 255.

SQR (2)

Devolve a raíz quadrada positiva do argumento; no exemplo, 1.41421356 é devolvida. SQR executa mais rapidamente do que ^.5 (isto é: elevar ao expoente 1/2).

STOP

Causa a interrupção da execução de um programa, e exibe uma mensagem informando em que número de linha contém o STOP. O controle do computador é devolvido ao usuário.

STORE MX

Grava uma matriz de números inteiros ou reais em fita cassete. Nenhuma mensagem ou sinal é fornecido: o usuário deve apertar as teclas "record" e "play" no gravador quando STORE é executado. "Bips" assinalam o início e o fim da gravação. O índice da matriz não é indicada quando STORE é usado. No exemplo, os elementos MX (Ø), MX (1), MX (2),... são salvos na fita; a variável MX não é afetada. Veja RECALL.

STR$ (12.45)

Devolve uma cadeia que representa o valor do argumento. No exemplo, a cadeia "12.45" é devolvida.

TAB (23)

Deve ser usado um comando PRINT. O argumento entre parênteses deve estar entre Ø e 255. Para argumentos de 1 até 255, se o argumento é maior que o valor da posição corrente do cursor, então TAB move o cursor para a posição especificada de exibição, contando a partir da extremidade esquerda da linha corrente. Se o argumento é menor do que o valor da posição corrente do cursor, então o cursor não é movido. TAB (Ø) põe o cursor na posição 256.

TAN (2)

Devolve a tangente do argumento, o qual deve estar em radianos. No exemplo, -2.185Ø3987 é devolvido.

TEXT

Especifica a tela para a modalidade usual de textos com 4Ø caracteres por linha e 24 linhas. Reposiciona, também, a janela de texto para a tela completa.

TRACE

Exibe na tela o número da linha de cada comando, à medida que eles são executados. TRACE não é desativado por RUN, CLEAR, NEW, DEL ou RESET. NOTRACE desativa TRACE.

USR (3)

Esta função passa seu argumento para uma sub-rotina em linguagem de máquina. O argumento é avaliado e colocado em um acumulador de ponto flutuante (localizações $9D até $A3), e uma JSR para a localização $ØA é executada. As localizações $ØA até $ØC devem conter uma JMP para a localização inicial da sub-rotina em linguagem de máquina. O valor de retorno para a função, é colocado no acumulador de ponto flutuante. Para voltar ao CCE BASIC, faça um RTS.

VAL ("-3.7E4A5CCE")

Tenta interpretar uma cadeia até o primeiro caráter não numérico, como um número real ou um inteiro, e devolve o valor daquele número. Se nenhum número ocorre antes do primeiro caráter não numérico, um Ø é devolvido. No exemplo, -37ØØØ é devolvido.

VLIN 1Ø,2Ø AT 3Ø

No modo de gráfico de Baixa Resolução, traça uma linha vertical na cor indicada pelo mais recente comando COLOR=. A linha é desenhada na coluna indicada pelo terceiro argumento. No exemplo, a linha é traçada de Y=1Ø para Y=2Ø em X=3Ø.

VTAB (15)

Move o cursor na tela para a linha especificada pelo argumento. A linha superior é a linha 1; a linha inferior é a 24.

WAIT 16ØØØ,255
WAIT 16ØØØ,255,Ø

Permite a inserção de uma pausa condicional em um programa. O primeiro argumento é o endereço decimal da posição de memória a ser testada para ver quando certos bits são altos (1, ou ligado) e certos bits são baixos (Ø, ou desligado). Cada bit no número binário equivalente ao decimal do segundo argumento, indica se você está interessado no bit correspondente, na posição de memória: 1 significa que você está interessado, Ø significa ignorar aquele bit. Cada bit no número binário equivalente ao

decimal do terceiro argumento, indica em qual estado você espera que esteja o bit correspondente na posição de memória: 1 significa que o bit deve ser baixo; Ø significa que o bit deve ser alto. Se não existe terceiro argumento, o Ø é assumido. Se qualquer um dos bits indicados por um bit 1 no segundo argumento estiver de acordo com o estado daquele bit indicado pelo bit correspondente no terceiro argumento, a instrução WAIT acaber.

XDRAW 3 AT 18Ø,12Ø

Desenha a "definição de forma" número 3, a partir de uma "tabela de forma" carregada anteriormente, em gráficos de Alta Resolução, começando em (X=18Ø, Y=12Ø). Para cada ponto traçado, a cor é a complementar daquela já existente naquele ponto. A instrução fornece uma maneira fácil para apagar: se você utiliza XDRAW para desenhar uma forma, um novo XDRAW apagará a mesma, sem alterar a cor de fundo da tela.

Sugestões para o aperfeiçoamento do manual, bem como indicações de possíveis falhas encontradas no decorrer deste são bem vindas.

Favor enviá-las à:

CCE - Indústria e Comércio de Componentes Eletrônicos S/A DIVISÃO DE INFORMÁTICA Av. Otaviano Alves de Lima, 2724 CEP 02501 - São Paulo - SP.

NOME :_____

PROFISSÃO :_____

CARGO :_____

TELEFONE :_____

MANUAL :_____

SUGESTÃO :_____

_____

_____

_____

_____

_____

_____

_____

_____